ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

На правах рукописи

Полухин Павел Валерьевич

МЕТОДЫ ОРГАНИЗАЦИИ ПРОЦЕССА ФАЗЗИНГ-ТЕСТИРОВАНИЯ И АНАЛИЗА ВЕБ-ПРИЛОЖЕНИЙ НА ОСНОВЕ МОДЕЛЕЙ ДИНАМИЧЕСКИХ БАЙЕСОВСКИХ СЕТЕЙ

Специальность 2.3.8. Информатика и информационные процессы

Диссертация на соискание ученой степени

доктора технических наук

Научный консультант:
доктор технических наук, профессор
Азарнова Татьяна Васильевна

Воронеж – 2024

Оглавление

Введение

Актуальность темы исследования. Современный этап в разработке информационных систем характеризуется переходом от локальных информационных систем к веб-ориентированным информационным системам. Переход к веб-ориентированным информационным системам требует обеспечения более высокого уровня безопасности разработок, который достигается применением современных эффективных технологий тестирования. Тестирование сопровождает каждый этап разработки, требования к уровню тестирования различных функциональных блоков веб-приложений формируются в зависимости от критичности ошибок их функционирования. Несмотря на большое количество исследований, проблемы, связанные с организацией тестирования, нельзя считать полностью решенными. Необходимо развивать новые подходы тестирования в плане разработки общих методологий, оптимизирующих качество, затраты ресурсов и надежность, а также создавать новые частные алгоритмы, направленные на избирательное тестирование классов ошибок, выделенных в соответствии с классификациями OWASP и MITRE. Эффективной технологией тестирования веб-приложений является фаззинг, заключающийся в подаче веб-приложению в качестве запросов или команд управления некорректных или случайных данных. Классический фаззинг белого, серого и черного ящика достаточно успешно решает проблемы обнаружения некоторых ошибок функционирования веб-приложений. Но постоянное усложнение структуры веб-приложений, применение широко спектра библиотек и программных компонентов, реализующих механизмы сетевого, межпрограммного взаимодействия приводит к тому, что классические подходы на основе эвристического анализа становятся ресурснозатратными и недостаточно эффективными. Требуется разработка новых подходов и методов организации фаззинга, в частности, учитывающих временную природу возникновения ошибок на различных этапах функционирования веб-приложений.

Тестирование методом фаззинга одного блока ошибок или цепочки блоков, как правило, представляет собой сложноорганизованную иерархическую

последовательность этапов, связанных между собой стохастическими информационными потоками и предикативными условиями. Процессы тестирования можно рассматривать как стохастические процессы, для повышения эффективности их организации и извлечения полезной информации для данных процессов можно использовать моделирование, которое способно обучаться, адаптироваться к ретроспективной информации и хотя бы частично прогнозировать развитие процессов анализа. Подобным требованиям к аппарату моделирования хорошо отвечает аппарат динамических байесовских сетей.

В случае представления процессов тестирования в виде динамических байесовских сетей, возникает ряд трудностей, связанных с расчетом начальных и транзитивных распределений вероятностей, а также с достижением требуемого уровня согласованности выборок, формируемых в задачах вероятностного вывода. Для решения описанных выше проблем необходима оптимизация существующих инструментов работы с динамическими байесовскими сетями и разработка новых методов и гибридных алгоритмов.

Несмотря на достаточно глубокие исследования в данной сфере, есть целый ряд актуальных направлений оптимизации алгоритмов формирования структуры, настройки параметров вершин и реализации процедур вероятностного вывода, в частности, ряд алгоритмов разработан только для статических байесовских сетей и их адаптация к условиям динамических байесовских сетей это отдельная задача, имеющая широкий спектр приложений. Оптимизировать существующие подходы обучения и вероятностного вывода можно за счет применения методов математической статистики, позволяющих сократить область формирования выборок и повысить степень их согласованности, а также использовать распределенные алгоритмы генерации выборок, матричные и графовые вычисления, базирующиеся на оптимальной синхронизации распределенных систем обработки данных с использованием моделей массового обслуживания.

Степень разработанности. Исследованиям вопросов тестирования посвящено большое число научных работ, среди которых можно выделить масштабные исследования А.И. Аветисяна, А.А. Браницкого, И.В. Котенко, Т.И. Булдаковой,

П.Д. Зегжды, А.А. Сироты, С.А. Арустамова, А.А. Белецкого, А.К. Петренко, М. Залевского, М. Саттона, А. Грина, П. Годфройда, Б. Миллера, Дж. Де Мотта, З. Питерсона, М. Пределя, Л. Чжана и др. Вопросы применения математического моделирования, интелектуального анализа данных и машинного обучения для моделирования процессов тестирования и обнаружения аномального поведения программных компонентов информационной системы рассмотрены в работах А.А. Браницкого, И.В. Котенко, Т.И. Булдаковой, П.Д. Зегжды, А.А. Сироты. Фундаментальные подходы фаззинг-тестирования заложены в работах А.И. Аветисяна, П. Годфройда, Б. Миллера, Дж. Де Мотта, Л. Чжана. Анализ современных исследований в области тестирования показывает достаточно быстрое и успешное развитие данной сферы, основными трендами которой являются: ускорение процессов тестирования за счет автоматизации рутинных процедур регрессионного тестирования, применение методов интеллектуального анализа данных и машинного обучения для прогнозирования ошибок функционирования программного обеспечения в автоматизированных системах тестирования, создание глобальной стратегии оценки качества программ. Предложенное в данной работе применение вероятностных графических байесовских моделей для организации процессов тестирования является новым направлением применения инструментов интеллектуального анализа данных, данное направление в полной мере отвечает тенденциям развития методов и алгоритмов тестирования. В работе исследуется не только концепция применения байесовских сетей в данной сфере, но и вопросы повышения ресурсной и временной эффективности вычислительных методов и алгоритмов для динамических байесовских. Используемые в работе технологии повышения эффективности алгоритмов обучения и вероятностного вывода для динамических байесовских сетей, являются развитием подходов, изложенных в работах В.И. Городецкого, А.В. Сироткина, А.Л. Тулупьева, К.В. Воронцова, В.Б. Сулимова, С.И. Николенко, И. Тсамардиноса, С.Ф. Алефиса, К. Мерфи, С. Рассела, П. Норвига, Д. Перла, К. Бишопа, Д. Барбера, Б. Корба, Н. Фридмана, Д. Хекермана, Д. Чикерина, А. Дусета, П.Д. Морала, С. Сарка, Д.Б. Рубина и др.

В диссертационном исследовании рассмотрен комплексный подход к моделированию процессов тестирования методом фаззинга с помощью динамических байесовских сетей, включающий разработку новых методов и гибридных алгоритмов обучения и вероятностного вывода, позволяющий произвести оценку динамики возникновения ошибок с течением времени и выработать набор тестовых генераций, локализующих потенциально возможные (аномальные) ошибки веб-приложений и дать возможность сформировать прогноз на несколько состояний в будущем.

Актуальность тематики диссертационного исследования обусловлена необходимостью дальнейшего расширения теоретической и практической базы инструментов моделирования, анализа и синтеза процессов тестирования в условиях непрерывного совершенствования и развития программной экосистемы веб-приложений. В диссертации решается крупная научная проблема, связанная с разработкой перспективных инструментов тестирования программных продуктов на базе применения современного аппарата математического моделирования, эффективных технологий обработки данных и организации вычислительного процесса. Решение данной проблемы имеет важное прикладное и хозяйственное значение.

Тематика работы соответствует «Стратегии научно-технологического развития РФ» от 28 февраля 2024 г., одним из направлений которой являются новые современные технологии проектирования, которые базируются на внедрении интеллектуальных производственных решений, вычислительных систем высокой производительности и современных методов искусственного интеллекта. Диссертационное исследование выполнено в рамках научной темы ФГБОУ ВО Воронежский государственный университет «Математическое моделирование, программное и информационное обеспечение, методы вычислительной и прикладной математики и их применение к фундаментальным исследованиям в естественных науках».

Цель работы. Заключается в разработке с единых методологических позиций направлений повышения эффективности организации информационных процессов

фаззинг-тестирования и анализа веб-приложений на основе моделирования данных процессов с помощью динамических байесовских сетей, способных аккумулировать информацию о различных этапах тестирования и формировать прогнозы методами вероятностного вывода, и в рамках развития математического инструментария исследования, разработать направления по временной и ресурсной оптимизации алгоритмов обучения и вероятностного вывода для динамических байесовских сетей, базирующихся на методах статистического оценивания и нелинейной фильтрации.

Для достижения цели исследования решались следующие теоретические и практические задачи:

1. С позиции системной методологии провести анализ существующих подходов к моделированию процессов тестирования ошибок функционирования веб-приложений с целью выявления направлений совершенствования методов обработки информации, генерируемой в процессе тестирования, и прогнозирование результатов тестирования, предложить и обосновать подход моделирования процессов тестирования с помощью динамических байесовских сетей.

2. Построить модели динамических байесовских сетей для процессов тестирования основных классов ошибок функционирования веб-приложений методом фаззинга, отражающие динамику изменения состояния надежности различных аспектов функционирования приложений во времени.

3. Разработать комплекс методов, реализующих гибридные технологии обучения структуры и параметров динамических байесовских сетей, включающий обучение топологии, направленности связей между отдельными узлами сети и вероятностных параметров вершин сети, базирующихся на методах статистического оценивания и структурной оптимизации, а также адаптированных для решения задач обучения в условиях неполных данных.

4. Разработать метод повышения эффективности процедуры вероятностного вывода на основе многочастичного фильтра для динамических байесовских сетей, базирующийся на применении метода Метрополиса-Гастингса на этапе повторной генерации выборок, деревьев сочленения и теории достаточных статистик.

5. Разработать метод синхронизации большого объема данных, передаваемых в распределенной вычислительной системе в процессе решения задач обучения и вероятностного вывода для динамических байесовских сетей тестирования веб-приложений методом фаззинга.

6. Разработать программное обеспечение, предназначенное для апробации полученных в диссертации результатов и провести на базе данного программного обеспечения комплексный вычислительный эксперимент по оценке эффективности применения предложенных в исследовании инструментов для организации процесса тестирования основных групп ошибок функционирования веб-приложений по классификации OWASP и MITRE.

Методы исследования. В рамках диссертационного исследования для моделирования процессов тестирования веб-приложений методом фаззинга использовались методы: теории систем, теории тестирования, теории графов, теории вероятностей и математической статистики, теории случайных процессов, теории байесовских сетей. В рамках решения задач обучения структуры и параметров динамических байесовских сетей применялись методы: теории графов, статистического оценивания, теории оптимизации. В процессе создания программно-ориентированных инструментов тестирования использовались технологии параллельного программирования, облачных вычислений и распределенной обработки данных.

Тематика работы соответствует следующим пунктам паспорта специальности 2.3.8. Информатика и информационные процессы: п. 1. Разработка компьютерных методов и моделей описания, оценки и оптимизации информационных процессов и ресурсов, а также средств анализа и выявления закономерностей на основе обмена информацией пользователями и возможностей используемого программно-аппаратного обеспечения; п. 7. Разработка методов обработки, группировки и аннотирования информации, в том числе, извлеченной из сети интернет, для систем поддержки принятия решений, интеллектуального поиска, анализа; п. 17. Разработка методов обеспечения надежной обработки информации и обеспечения помехоустойчивости информационных коммуникаций

для целей передачи, хранения и защиты информации; разработка основ теории надежности и безопасности использования информационных технологий.

Научная новизна работы. В работе получены следующие результаты, характеризующиеся научной новизной.

1. Разработан новый подход к организации процесса тестирования веб-приложений по методологии фаззинга, отличающийся моделированием данного стохастического процесса с помощью комплекса динамических байесовских сетей, позволяющий отражать динамику шагов тестирования, рассматривать уже определенные элементы тестирования в виде переменных свидетельств на определенных временных срезах сетей, прогнозировать значения целевых переменных запроса, планируемых в будущем, и нереализованных ранее элементов тестирования на основе процедур вероятностного вывода.

2. Построены оригинальные динамические байесовские модели фаззинг-тестирования OWASP-групп ошибок функционирования веб-приложений, позволяющие учитывать временные и вероятностные связи между различными элементами тестирования.

3. Разработан метод обучения структуры графа динамических байесовских сетей на основе формирования марковского покрытия, учитывающий при обучении транзитные связи между соседними временными срезами сетей, отличающийся использованием гибридных технологий обучения, базирующихся на статистических подходах оценки топологии байесовских сетей и применении оптимизационной процедуры имитация отжига для определения направленности связей между отдельными узлами сетей.

4. Разработан адаптированный к условиям неполных данных метод обучения вероятностных параметров динамических байесовских сетей, отличающийся применением стохастического алгоритма ожидания максимизации с использованием семплирования по методу Монте-Карло с применением аппарата цепей Маркова.

5. Разработаны адаптированные к динамическим байесовским сетям методы вероятностного вывода, базирующиеся на многочастичном фильтре,

отличающиеся применением алгоритма Метрополиса-Гастингса на этапе повторной генерации выборок и теории достаточных статистик для сокращения количества выборок и оптимизации расчета весов генерируемых выборок, теоремы Лемана и Шеффе и технологии снижения сложности структуры динамических байесовских сетей на основе построения деревьев сочленений.

6. Разработан метод синхронизации большого объема данных, передаваемых в распределенной вычислительной системе в процессе решения задач обучения и реализации вероятностного вывода, базирующийся на методах Сузуки-Касами и теории массового обслуживания, отличающийся применением инструментов вложенных цепей Маркова и статистической оптимизации для настройки параметров синхронизации.

7. Предложена структура параллельных вычислений в алгоритмах обучения и вероятностного вывода для динамических байесовских сетей, отличающихся применением разработанных в рамках исследования инструментов повышения эффективности процедур обучения и реализации вероятностного вывода.

8. Предложена структура вычислительного эксперимента, реализующего комплексный подход к организации на основе аппарата байесовских моделей процесса тестирования, выделенных по классификации OWASP и MITRE групп ошибок функционирования веб-приложений, направленный на оценку качества, временной и ресурсной эффективности разработанных в исследовании новых методов, моделей и гибридных алгоритмов обучения и вероятностного вывода.

Теоретическая и практическая значимость. Теоретическая значимость исследования заключается в разработке основополагающих аспектов применения динамических байесовских вероятностных моделей для организации и анализа информационных процессов тестирования веб-приложений методом фаззинга, развитии методов обучения структуры графов и параметров вершин данных моделей, реализации вычислительных процедур вероятностного вывода и обосновании эффективности использования предложенных методов для обеспечения информационной безопасности и качества функционирования веб-приложений.

Разработанные алгоритмы являются универсальными и могут быть использованы не только для динамических байесовских сетей тестирования веб-приложений методом фаззинга, но и для байесовских сетей, моделирующих другие временные стохастические процессы. Предложенные методы обучения структуры и параметров динамических байесовских сетей построены на основе существующих методов обучения структуры и параметров и направлены на достижение эффекта ресурсной оптимизации процессов установления вероятностных связей между переменными байесовских сетей. Разработанные методы вероятностного вывода базируются на гибридной комбинации существующих методов вероятностного вывода, направлены на оптимизацию процедур генерации выборок, оценки их значимости и распространение алгоритмов для статических байесовских сетей на динамические байесовские сети. Теоретической значимостью характеризуются созданные в рамках исследования методы распараллеливания вычислений в предложенных алгоритмах обучения структуры и вероятностного вывода для динамических байесовских сетей, использующие специальные подходы синхронизации вычислений на основе инструментов теории массового обслуживания. Практическая значимость исследования заключается в возможности использования результатов работы для повышения эффективности и качества реализации процесса тестирования веб-приложений. Применение данных инструментов позволяет прогнозировать результаты определенных элементов процесса тестирования по протоколам предшествующих элементов тестирования, вырабатывать механизмы поиска аномальных ошибок и повышать оперативность их локализации.

Результаты диссертационного исследования позволяют раскрыть потенциал динамических байесовских сетей для оптимизации информационных процессов тестирования веб-приложений методом фаззинга. Обоснованы возможности использования динамических байесовских сетей как инструментов анализа, выявления закономерностей, разработки методов обработки, распределения и аннотирования информации, разработки методов обеспечения помехоустойчивости информационных и телекоммуникационных систем для

решения задач передачи, хранения и защиты информации, циркулирующей внутри данных систем.

Положения, выносимые на защиту.

1. Подход к организации процедуры тестирования веб-приложений методом фаззинга на основе моделирования данного стохастического процесса с помощью комплекса динамических байесовских сетей, позволяющий прогнозировать наличие определенных ошибок функционирования на следующих временных срезах, а также результаты нереализованных ранее фрагментов тестирования на предыдущих временных срезах на основе решения фильтрации и сглаживания в рамках вероятностного вывода.

2. Динамические байесовские сети тестирования методом фаззинга основных OWASP-групп ошибок функционирования веб-приложений, позволяющие комплексно моделировать временные и вероятностные связи между различными элементами тестирования.

3. Метод обучения структуры графа динамических байесовских сетей, позволяющий оптимизировать процедуру восстановления топологии и направленности внутрисрезовых и транзитивных связей, базирующийся на статистических подходах оценки топологии байесовских сетей и применении оптимизационной процедуры имитация отжига для определения направленности связей.

4. Метод обучения вероятностных параметров динамических байесовских сетей, позволяющий повысить эффективность обучения в условиях неполных данных за счет применения стохастического алгоритма ожидания максимизации и семплирования по методу Монте-Карло с использованием аппарата цепей Маркова.

5. Методы реализации вероятностного вывода в динамических байесовских сетях на основе многочастичного фильтра, позволяющие сократить количество выборок, оптимизировать расчет весов генерируемых выборок и упростить процесс вычислений путем преобразования структуры сети.

6. Метод синхронизации большого объема данных, передаваемых в распределенной вычислительной системе в процессе решения задач обучения и вероятностного вывода для динамических байесовских сетей, позволяющий оптимизировать настройку параметров синхронизации.

7. Структура организации параллельных вычислений в алгоритмах обучения и вероятностного вывода для динамических байесовских сетей с применением предложенных в исследовании инструментов повышения эффективности процедур обучения и вероятностного вывода.

8. Структура программного комплекса для проведения вычислительного эксперимента, позволяющего оценить качество, временную и ресурсную эффективность разработанных в исследовании новых методов и гибридных алгоритмов обучения и вероятностного вывода.

Реализация и внедрение результатов работы.

Результаты диссертационного исследования используются в ЦСИИР АО НТЦ «Радиоэлектронной борьбы» в рамках организации процесса тестирования специализированных средств и технологий, применяемых в процессе разработки инструментов радиоэлектронной борьбы, и в IT-компании АО НПП «РЕЛЭКС» в системе комплексного автоматизированного и ручного тестирования разрабатываемых IT-решений для государственной и коммерческой сферы.

Основные результаты работы внедрены в учебный процесс факультета Прикладной математики информатики и механики Воронежского государственного университета и Военного учебно-научного центра Военно-воздушных сил «Военно-воздушная академия имени профессора Н.Е. Жуковского и Ю.А. Гагарина».

Степень достоверности и апробация результатов.

Достоверность защищаемых положений работы, работоспособность и результативность предлагаемых решений подтверждается:

1. приведенным в диссертации обоснованием постановок задач и используемых допущений, комплексным анализом существующих методов их решения, непосредственным сопоставлением полученных результатов с

фактическими данными, корректным применением методов моделирования, методов организации вычислительного процесса и других инструментов исследования;

2. результатами проведенного вычислительного эксперимента;

3. практическим внедрением результатов диссертации в деятельность высокотехнологичных IT-компаний и в учебном процессе.

Основные положения диссертационной работы докладывались и обсуждались на следующих конференциях: Международной научно-практической конференции «Технические науки – основа современной инновационной системы» (Йошкар-Ола, 2014), Международной научной конференции «Наука в современном обществе» (Ставрополь, 2014), Межвузовской научно-практической конференции «Молодежные чтения памяти Ю.А. Гагарина» (Воронеж, 2014), Международной научной конференции «Informative and communicative space and a person» (Prague, 2014), Международной научно-практической конференции «Перспективы развития информационных технологий (Новосибирск, 2014-2015), Международной научно-практической конференции «Научная дискуссия: вопросы технических наук» (Москва, 2013, 2014, 2015), Международной научной конференции «Математические методы распознавания образов» (Москва, 2019), Международной конференции «Цифровая индустрия: состояния и перспективы развития» (Челябинск, 2018), Международной конференции Динамические системы и компьютерные науки: теория и приложения (Иркутск 2022, 2023), Всероссийской конференции «Актуальные проблемы прикладной математики и механики», посвященной памяти академика А.Ф. Сидорова (Джанхот 2023), Международной научной конференции «Математические методы в технике и технологиях» (Минск 2022), Международной научно-практической конференции «Новые информационные технологии и системы» (Пенза 2022), Международной конференции «SUMMA» (Липецк, 2023), Международной научно-практической конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2017, 2018, 2020, 2021), а также на ежегодных научных сессиях Воронежского государственного университета.

Публикации. По теме исследования опубликовано 55 работ, отражающих основные положения исследования, в том числе 21 статья в изданиях, рекомендованных ВАК РФ, 7 статей в изданиях, индексируемых в базе данных Scopus и Web of Science, 16 научных статьи в других изданиях, получено 11 свидетельств о государственной регистрации программы.

Личный вклад соискателя. В работах, опубликованных в соавторстве и приведенных в конце автореферата, лично соискателю принадлежат: [65, 67] – описание процессов управления тестированиям веб-приложений методом фаззинга; [69] – стохастические методы обучения динамических байесовских сетей на основе алгоритма Метрополиса-Гастингса, [63] – алгоритмы вероятностного вывода для реализации процедур тестирования веб-приложений, [64] – решения задачи фильтрации динамических байесовских сетей на основе построения дерева сочленения, [66, 68] – обоснование расчета параметров модели синхронизации распределенных вычислительных систем для решения задач обучения, [165] – алгоритмы обучения структуры и параметров динамических байесовских сетей для решения задач тестирования веб-приложений, [174] – разработка оптимальной модели динамической байесовской сети тестирования SQL-инъекций, [180, 184] – процедура представления процесса тестирования в виде дискретного стохастического процесса с конечным числом состояний, [165, 179] – численные методы оптимизации алгоритма обучения динамической байесовской сети, [168, 182] – статистические методы повышения эффективности вероятностного вывода во временных моделях тестирования веб-приложений.

Структура и объем работы. Диссертация состоит из введения, пяти глав, заключения, списка использованной литературы, включающего 218 наименований и 4 приложения. Работа изложена на 361 странице машинописного текста, содержит 39 рисунков, 8 таблиц.

Глава 1. Динамические байесовские сети как инструмент организации процесса тестирования веб-приложений методом фаззинга

1.1. Фаззинг–тестирование веб-приложений

Веб-приложения представляют собой широко используемый онлайн-инструмент, функционирующий в рамках глобальной сети интернет и предназначенный для организации бизнес-процессов, продажи товаров и услуг, распространения информации и общения пользователей. Веб-приложения могут выступать в виде прикладных сервисов, отвечающих за взаимодействие с информационными системами во всем мире, а также предоставлять функционал пользовательского интерфейса для администрирования и доступа к данным пользователями внутри сети интернет. Инструменты проектирования и разработки информационных систем на базе веб-приложений позволяют упростить процесс создания сложных веб-приложений. Однако разработка качественного и надежного приложения является сложным и многогранным процессом, требующим углубленного понимания принципов взаимодействия не только функциональных, но и программных компонентов, отвечающих за определенный набор действий внутри программы. Неправильное понимание этого процесса неминуемо ведет к появлению различных ошибок, представляющих угрозу для целостности данных, и способствующих потери информации конфиденциального характера. Для решения данных задач используется стратегия тестирование приложений на предмет наличия программных ошибок.

Для анализа и оценки надежности и качества программного кода приложения, а также механизмов взаимодействия отдельных компонентов, функционирующих внутри приложений, используются различные инструменты тестирования. В общем случае эти инструменты можно разделить на две основные категории: встроенные, используются непосредственно в процессе построения приложений, и внешние, выступающие как дополнение к встроенным инструментам (используются после построения и разработки приложений). Тем не

менее, понятие программной ошибки тесно связано с устойчивостью приложения. Предполагается, что программа находится в состоянии устойчивости до тех пор, пока не наступило событие сбоя или ошибки. В рамках формального представления процесса исследования приложений следует, что существующие методы тестирования не дают полной гарантии закрытия программных ошибок, а поведение анализируемой программы может быть некорректным и непредсказуемым. Для решения данных проблем ученые и исследователи-практики, занимающиеся вопросами изучения надежности веб-приложений, разработали стратегию тестирования на основе технологии фаззинга. В научном сообществе понятие фаззинга тесно связано с понятием граничных значений, которое характеризуется некоторым фиксированным интервалом допустимых значений набора тестовых данных. Анализ граничных значений позволяет оценить эффективность применения метода исключений для удаления неэффективных наборов тестовых данных. Более того, фаззинг позволяет произвести анализ не только граничных, но и промежуточных значений, что способствует повышению охвата и покрытия входных данных, которые могут повлиять на состояние устойчивости функционирования приложения.

Впервые, точное определение фаззинга было сформулировано М. Саттоном и П. Амини [200] и на текущий момент является общепринятым и устоявшимся понятием. Фаззинг представляет собой технологию выявления ошибок в программных приложениях и компонентах. В основе фаззинга лежит возможность передачи на вход анализируемого объекта заведомо некорректных данных, формируемых для его аварийного завершения и последующий анализ данного события. В общей формулировке фаззинг не позволяет прогнозировать сам факт наличия программных ошибок в приложении в будущем, однако дает достаточно полную оценку надежности приложения в текущий момент времени. Фаззинг позволяет выстроить процесс тестирования в хронологическом порядке, что дает возможность выявить выборки тестовых данных, которые привели к возникновению события сбоя или ошибки. Структурно процесс фаззинга можно декомпозировать на следующие этапы: порождение тестовых данных и внедрение

их в целевое приложение, контроль состояния объекта, анализ и сохранение результатов тестирования. Исследователи выделяют две основные разновидности фаззинга, применяемые в процессе формирования тестовых выборок: генеративный и мутационный. Рассмотрим сущность каждого из них.

Генеративный фаззинг основан на автоматическом формировании случайных наборов входных данных с использованием заранее установленной грамматики тестируемого приложения. В качестве грамматики могут выступать, межпроцессные и сетевые протоколы, форматы файлов и документов, обрабатываемых приложениями. Моделирование процессов порождающего фаззинга представляет собой прозрачный процесс, основная особенность которого сводится к детальному пониманию грамматики и принципов функционирования приложений. Существующие методики тестирования на основе данного подхода обладают высокой производительностью, в то время как показатели качества обнаружения ошибок находятся на достаточно низком уровне и вероятность ложных срабатываний достигает своего максимума. В связи с этим использование порождающего фаззинга целесообразно лишь для тестирования простых приложений, если качеством тестирования можно пренебречь по сравнению со скоростью тестирования. Для решения задач повышения качества детектирования ошибок, учеными и исследователями была выдвинута концепции тестирования на основе мутационного фаззинга.

Мутационный фаззинг основывается на вероятностном изменении тестовых данных с учетом принципов функционирования приложения. При этом набор грамматик, лежащий в основе приложения, может являться статическим и динамическим. Реализация процессов моделирования с использованием этого подхода представляет собой достаточно сложную задачу, так как требует от исследователя четкого и однозначного понимания внутренней структуры приложения для реализации механизмов преобразования тестов на основе мутации. В тоже время процесс тестирования должен быть выстроен в хронологическом порядке относительно некоторого временного интервала, характеризующего этапы тестирования нескольких приложений. Но именно эти особенности позволяют

рассматривать фаззинга в виде адаптивного и предметно-ориентированного подхода к тестированию, направленного на анализ не только наиболее известных ошибок, но и на выявление аномальных и ранее неизвестных ошибок устойчивости.

Таким образом, использование комбинированного подхода на основе объединения представленных выше функциональных характеристик дает возможность повысить качество детектирования аномального поведения внутри приложений и установить требуемые критерии надежности программ и их компонентов [127, 157]. Комбинированный подход позволяет реализовать концепцию интеллектуального фаззинга и служит фундаментом для построения сложных систем тестирования. В научных трудах принято выделять следующие виды тестирования, направленные на выявление ошибок методом фаззинга и представленные на рисунке 1.1

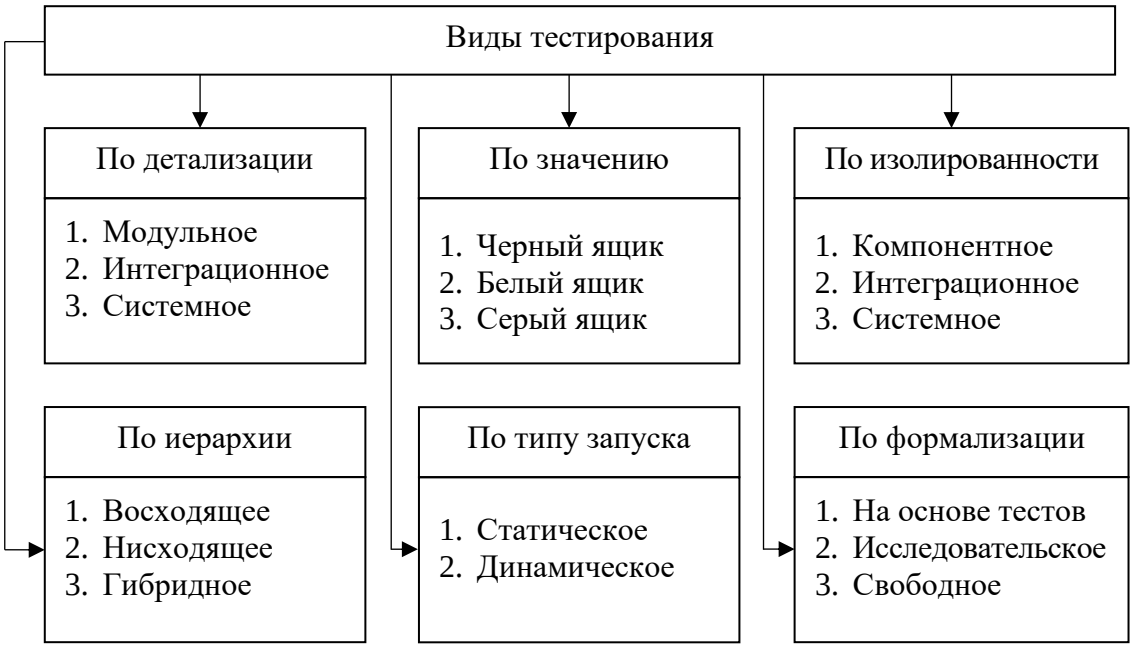

Рисунок 1.1. Виды тестирования

Рассматривая фаззинг с позиции тестирования, все методы анализа программ могут быть представлены в виде отдельных компонентов фаззинга. В связи с этим выделяют следующие основные группы методов тестирования: белого, черного и серого ящика [84]. Каждая группа обладает своими отличительными признаками. Рассмотрим каждый из методов по отдельности.

Метод белого ящика. Основная направленность белого ящика – тестирование исходного кода приложений посредством эвристического анализа.

Применение данного подхода позволяет выполнять обнаружение ошибок, учитывая внутреннее строение рассматриваемой программы. В свою очередь, метод белого ящика позволяет реализовать статический анализ программного кода, что дает возможность выявить фрагменты, в которых возможно появление ошибок. Особую роль в использование данного метода занимает анализ логики работы программы, вызовов функций и обработка данных, поступающих на вход приложения. Метод белого ящика находит широкое применение не только в системах тестирования, но и в инструментах интегрированной разработки программ. Данный метод выявления программных ошибок тесно связан с понятием модульного тестирования, включающего инструменты выявления ошибок программ на функциональном уровне и отвечающего за обнаружение ошибок уровнях компиляции и выполнения определенного модуля. Данный подход иногда называют «прозрачным» тестированием, поскольку он позволяет получить доступ к исходному коду приложения, последовательно восстановить логику выполнения программы и выстроить хронологию появления программных ошибок для определенного типа приложений.

Метод черного ящика. Основа концепции данного тестирования состоит в том, что сведения о внутренних механизмах и структуре приложения неизвестны и не учитываются на этапе анализа. При этом от тестировщика требуется лишь базовые знания архитектуры приложения без непосредственного доступа к программному коду. Данная стратегия может быть применима к удаленным веб-сервисам, реализующим прикладной программный интерфейс для организации абстрактного доступа к данным и ресурсам информационных систем посредством сетевого взаимодействия. Это позволяет обнаружить недокументированные возможности приложений с позиции анализа надежности и выявления наиболее неустойчивых компонентов данной системы. На сегодняшний день можно выделить следующие основные разновидности тестирования черного ящика. Рассмотрим их сущность и содержание.

Основные виды тестирования методом черного ящика [103].

Эквивалентное разбиение. Позволяет уменьшить объем тестовых выборок за

счет декомпозиции входных элементов на отдельные фрагменты. В общем случае данный подход позволяет разделить входные и выходные данных на конечное число классов эквивалентности и выбрать соответствующее значение для каждого класса. При этом результаты тестирования одного значения определенного класса полагаются эквивалентными другим значения того же класса. Если в процессе тестирования определенного набора тестовых данных появляется ошибка, то другие эквивалентные тесты будут обнаруживать туже самую ошибку. Важной особенностью данного подхода является то, что он позволяет придерживаться выборочной стратегии, дает возможность сократить объем комбинаторных выборок, необходимых для проведения тестирования, и определить логические критерии отбора наиболее корректных выборок из всего набора тестовых генераций.

Анализ граничных значений. Данный вид тестирования направлен на реализацию функционального тестирования программ путем определения верхних и нижних границ входных параметров. Выход за пределы данных границ является индикатором наличия программной ошибки. Существует три основных типа границ: корректные, некорректные и пограничные. Используя граничные значения, можно задать область значения параметра, а также определить максимальное и минимальное значения внутри данной области. В рамках превентивного тестирования применяются репрезентативные данные, область значения которых выходит за пределы ранее установленного диапазона.

Данный механизм можно применять как на функциональном, так и на структурном уровне тестирования, что позволяет повысить обнаружение ошибок, проявляемых во входных параметрах и связанных с их выходом за пределы установленного диапазона. Выделяют три типа границ входных данных: правильные, неправильные и граничные. Данный подход использует значения, лежащие внутри, на границах (крайние точки) и минимальные (максимальные) значения. При тестировании за пределами допустимого диапазона используются репрезентативные данные, которые находятся за пределами диапазона, тем самым заведомо принимают ошибочные значения.

Отладка переходных состояний. Основывается на анализе графа состояний, а также навигацию и ориентирование внутри графа пользовательского интерфейса приложения для определения всех возможных комбинаций переходов из одного фиксированного состояния в другое. В этом случае анализируемое приложение может быть представлено в виде конечного автомата с фиксированным набором переходов. Диаграмма переходов представляет собой графическое отображение конечного автомата. Основная задача данной диаграммы сводится к отображению всех возможных состояний, которые может достичь система и ее элементы. При этом более глубокий анализ позволяет определить события или предпосылки к переходу в то или иное состояние [95, 96]. Использование сложных и композиция более простых диаграмм позволяет определить процессы трансформации составных компонентов системы, принципы взаимодействия между объектами, а также логику выполнения.

Функциональные диаграммы. Методика основана на создании графа состояний и установлении связей между действиями и причинами выполнения этих действий: тождественность, отрицание, логические «и» и «или». Использование данного подхода позволяет представить тестовую систему в виде совокупности причинно-следственных связей. В качестве элементов данной системы могут выступать как сами наборы тестовых данных, выступающие в качестве причины возникновения сбоев и ошибок, так и отдельные компоненты программных продуктов, реагирующие на ошибки и порождающие определенные результаты, выступающие как следствие.

Тестирование всех пар значений. Реализует методику порождения наборов тестовых данных, включающих в себя все возможные дискретные комбинации каждой пары параметров. Данный подход позволяет установить степень корреляции результатов тестирования нескольких параметров внутри одной программной функции или модуля. При этом предполагается, что обработка параметров является одновременной и приводит к разным результатам в зависимости от различных комбинаций тестовых выборок, передаваемых в тело каждого входного параметра.

Метод серого ящика. Выступает объединением методов черного и серого ящика. Использование данного метода наиболее характерно в тех случаях, когда не известна структура самого приложения, а лишь понятна логика обработки параметров некоторыми модулями. В качестве таких модулей могут выступать компоненты, отвечающие за сетевое взаимодействие, обработку данных, межпроцессное и межмодульное взаимодействие. Процедуры тестирования серого ящика можно представить в виде черного ящика с замкнутой на вход обратной связь, которая используется для реализации механизмов корректировки и своевременного обновления выборок. Это способствует выработке тех тестовых сценариев, которые с максимальной вероятностью способны привести к обнаружению определенного типа программной ошибки или групп ошибок. Принципы тестирования, лежащие в основе данного метода, находят широкое применение в анализе надежности приложений, разрабатываемых для глобальной сети интернет.

Для оценки эффективности каждого из методов в работе систематизированы критерии, позволяющие дать сравнительную характеристику методов тестирования программного обеспечения.

Таблица 1.1 Сравнительная характеристика методов тестирования

Критерий	Черный ящик	Белый ящик	Серый ящик
Охват кода	охват входных параметров	полный	охват входных параметров
Управление потоком выполнения	частичный	полный	частичный
Направленность	нет	полная	на основе статистики
Воспроизводимость	максимальная	частичная	максимальная
Доступность	локальный и удаленный доступ	исходный код	любой тип доступа
Сложность	зависит от числа параметров	зависит от объема кода	зависит от числа параметров
Граничность	частичная	по согласованию	полная

В классической интерпретации фаззинг больше ассоциируется с понятием черного ящика, в то время как мутационный, соотносится с механизмами серого ящика. В рамках научного исследования основным подходом тестирования на основе фаззинга будет методология серого ящика, объединяющая воедино

стратегию черного и белого ящика, и позволяющая анализировать приложения при наличии или отсутствии исходного кода посредством локального или удаленного тестирования. Именно это доказывает обоснованность концепции применения фаззинга в качестве основной технологии тестирования веб-приложений.

Еще одним важным понятием, связанным с тестированием, является понятие покрытие тестами. Данное понятие устанавливает численную оценку между общим числом функциональных фрагментов программы и точным числом тестов [129], необходимых для их полного тестирования. Для анализа тестовых наборов используются различные метрики покрытия, позволяющие установить минимальное пороговое значение функции покрытия, необходимое для обеспечения максимального охвата программных компонентов тестами. Одним из таких критериев является критерий полноты тестового покрытия. Допустим, если P – множество программных компонентов, T – множество тестов, Σ – множество тестовых наборов, такое что $\Sigma \subset T$, тогда задача порождения тестовых данных может быть сформулировано с точки зрения покрытия. Для некоторой системы тестирования $S \in P$ необходимо построить тестовый набор $\sigma \in \Sigma$, удовлетворяющий критерию полноты тестирования $F: P \times S$. При этом F выступает в качестве критерия для тестовой системы S и определяет элементы покрытия Q_S^F. В рамках диссертационной работы под элементами тестового покрытия будем понимать набор событий, возникающих в процессе тестирования целевой программы. Смежным понятием по отношению к критерию полноты покрытия выступает метрика тестового покрытия [127]. Данная метрика представляет собой функцию $M: P \times S \to R$. При этом данная функция представляет собой числовую оценку, показывающую насколько полно тестовый набор σ, покрывает тестовую систему S. Данный критерий может быть представлен в виде следующего выражения:

$$M(S, \sigma) \geq a_S. \tag{1.1}$$

где a_S – минимальное значение пороговой метрики M для тестовой системы S.

Далее рассмотрим структуру процесса тестирования веб-приложений методом фаззинга. Данная структура устанавливает не только последовательность

действий, реализуемых в процессе тестирования, но и определяет критерии завершения при достижении требуемого охвата программных компонентов, для которых выполняется тестирование. Данное условие достигается за счет комбинирования тестирования и сканирования. Этап сканирования подразумевает оправку сгенерированных наборов тестовых данных в целевое приложение, а также обработку выходных результатов на предмет наличия программной ошибки. Сам процесс является циклическим, а коррекция тестовых данных производится в том случае, если в процессе сканирования не были выявлены ошибки. Схема процесса тестирования приведена на рисунке 1.2

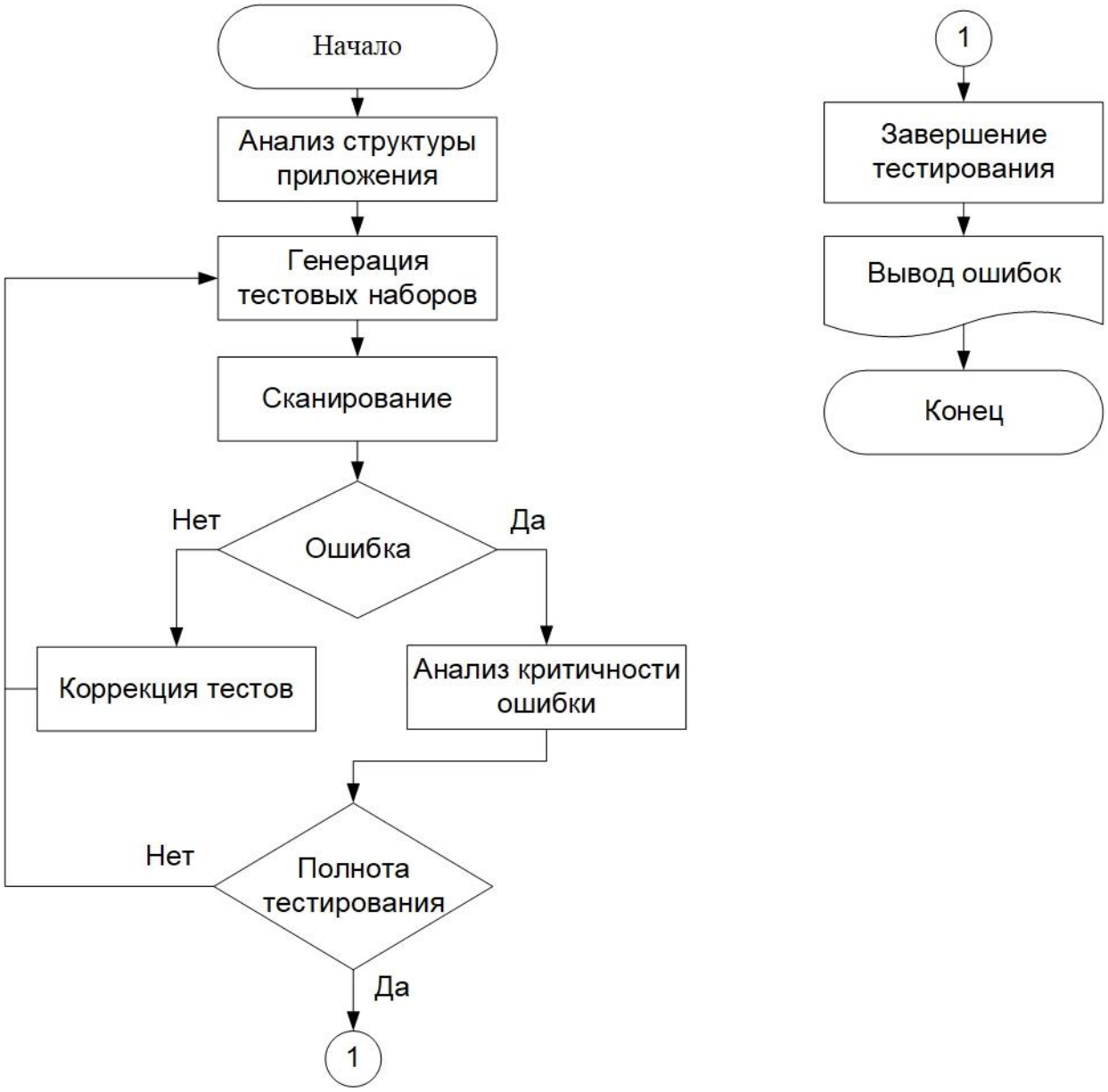

Рисунок 1.2. Структура процесса тестирования методом фаззинга

Из рисунка 1.2 видно, что структура процесса адаптирована к тестированию

серого ящика за счет применения механизмов первичного анализа функциональных особенностей рассматриваемого приложения, что позволяет производить генерацию тестовых выборок более осмысленно. Использование данного подхода дает возможность адаптироваться и локализовать уже известные ошибки и выявить ранее неизвестные ошибки (аномалии).

Программные ошибки, возникающие внутри веб-приложений, имеют определенного рода направленность. Они способны извлекать данные посредством различных механизмов эксплуатации, обходить встроенные фильтры, правила и блокировки, выполнять динамический код внутри клиентских приложений и т.д. Именно это определяет спецификацию ошибок, а также дает возможность объединить ошибки в определенные группы, что позволяет более детально понять природу возникновения ошибок и выработать ряд направлений для их локализации. Все больше и больше компаний по всему миру пытаются стандартизировать данную классификацию, используя опыт как своих, так и сторонних компаний, ведущих разработку веб-приложений. Однако постоянная трансформация программных ошибок затрудняет данный процесс, так как не всегда возможно своевременно сопоставить ошибку определенной группе и выработать стратегию для ее локализации. На сегодняшний день наиболее распространены исследования надежности программ следующих зарубежных и отечественных компаний: MITRE, Aspect Security, Offensive Security, Rapid 7, Disa, Positive Technologies, Bi.Zone, Group-IB, InfoWatch, Kaspersky Industrial Cybersecurity и т.д. Преимущественно деятельность данных компаний связана с выработкой критериев надежности и устойчивости функционирования информационных систем за счет локализации программных ошибок. На начальных этапах распространения сети интернет и роста популярности веб-приложений существовало несколько ведущих компаний, специализирующихся на разработке интернет-сервисов, однако сейчас каждая компания может использовать концепцию построения и использования веб-приложений в рамках реализации своих бизнес-процессов. Разнородность и несогласованность требований к инструментам тестирования и механизмам детектирования ошибок приводит к их

постоянному росту. Для решения данных задач группа ученых во главе с М. Керфи, Д. Гровс впервые разработали и представили проект, предназначенный для обеспечения надежности и отказоустойчивости веб-приложений (OWASP). К данному проекту присоединились сотни компаний по всему миру, а также отдельные исследователи, численность которых постоянно растет. Это позволило выработать и постоянно поддерживать в актуальном состоянии основные направления исследования проекта, реализованные за счет выработки критериев и предложений по обеспечению надежности и устойчивости функционирования веб-приложений. Систематизируя аналитические данные по типам ошибок и способам их локализации, конструктивно проект был разделен на несколько подпроектов, каждый из которых предназначен для решения определенного круга задач [126].

Проект OWASP Top 10. Содержит детальную классификацию десяти наиболее критичных программных ошибок веб-приложений. Между тем в рамках данного проекта представлены возможные ситуативные факторы и модели появления программных ошибок. Представлен расширенный анализ последствия воздействий на компоненты программ и информационную систему в целом при эксплуатации данных ошибок устойчивости.

Руководство по разработке кода OWASP (Code Review). Представляет собой подробное руководство и набор сценариев проведения тестирования с целью обнаружения программных ошибок, а также рассматривается вопрос интеграции тестирования в рамках цикла разработки программного обеспечения. Целью данного руководства является обучить разработчиков создавать надежный и качественный код, а также предлагается ряд готовых решений для закрытия программных ошибок из классификации *OWASP Top 10*.

Руководство OWASP (Guide Project). Данное руководство предоставляет большое число полезных материалов для разработчиков, ключевыми элементами которых являются: архитектура, дизайн, построение и конфигурация приложений. Это позволяет разработчикам и архитекторам программного обеспечения еще на ранних стадиях формирования проекта отказаться от тех функциональных

элементов, которые будут заведомо содержать ошибки. Такой подход позволит повысить устойчивость разрабатываемой программы к возникновению ошибок.

Инструменты тестирования OWASP (Tool). Включают в себя программные инструменты, разрабатываемые в рамках проекта OWASP и предназначенные для тестирования программ на предмет наличия ошибок и использования отдельных модулей в рамках построения комплексных систем тестирования программных систем, построенных на базе веб-приложений.

Концепции, предложенные OWASP, находят широкое применение в проектах компаний по всему миру, способствуют формированию требований к отказоустойчивости и надежности программных продуктов, а также выработке практических навыков разработчиков, направленных на определение комплексных механизмов поиска программных ошибок. Классификация и характеристика ошибок согласно OWASP представлена в таблице 1.2

Таблица 1.2 Классификация основных ошибок веб-приложений

Название ошибки	Характеристика
Нарушение контроля доступа	Ограничения функционала приложения, доступное только авторизованным пользователям спроектировано некорректно. Нелегитимный пользователь может выполнить доступ к данным веб-приложения без должных прав и привилегий.
Криптографические ошибки	Большинство приложений хранят и обрабатывают данные в открытом виде. Это дает возможность сторонним пользователям получать, модифицировать и использовать данные по своему усмотрению. Сохранность данных требует использования вспомогательных механизмов шифрования в процессе обмена информацией между клиентом и сервером.
Инъекции	Инъекции возникают, если входные данные поступают в приложение в виде фрагмента команды или запроса. С помощью внедрения подзапроса, можно реализовать процедуру получения доступа к данным без какого-либо подтверждения.
Небезопасная архитектура	Разработка безопасных приложений требует особого подхода к анализу ошибок, связанных с проектированием. Необходимо производить моделирование угроз, использовать принципы безопасного проектирования и эталонных архитектур.
Нарушение конфигурации системы	Сохранение устойчивости и надежности функционирования приложений требует качественной настройки и конфигурации информационной системы. Основополагающие правила к проектированию и построению системы должны быть явно определены. Требования к настройке должны быть четко обозначены, а компоненты системы должны быть постоянно в актуальном состоянии и своевременно обновлены.

Использование компонентов с ошибками	Наборы программных компонентов, библиотек не всегда имеют те же права, что и основное приложение. Если данные компоненты содержат ошибки, они могут быть использованы для получения доступа к данным и реализации полного контроля над системой.
Нарушение механизмов аутентификации и управления сессиями	Функции приложения, отвечающие за механизмы аутентификации и управления процессами идентификации пользователей, зачастую плохо продуманы, что позволяет злоумышленники выполнить атаку и получить доступ к идентификационным данным на временной или постоянной основе.
Нарушение целостности программных компонентов и данных	Процедура обновления программных компонентов, критичных данных и заданий на сборку приложений CI/CD выполняется без соответствующей проверки целостности. В результате чего сторонние пользователи могут встроить отдельные фрагменты кода в целевое приложение и скомпрометировать его.
Отсутствие журнала безопасности и мониторинга действий	Ведение журнала безопасности и протоколирование действий позволяет своевременно реагировать на возникающие угрозы и своевременно обнаружить потенциальные входные точки, используемые для получения несанкционированного доступа к целевой системе.
Межсайтовая подделка запросов на стороне сервера (SSRF)	Реализация SSRF основана на выполнении запросов с использованием сеансовых переменных, а также другой информацию, используемой в процессе аутентификации пользователей. SSRF позволяет стороннему пользователю имитировать выполнение запросов от имени легитимного пользователя.

Агрегируя классификацию программных ошибок, приведенную в таблице 1.2, можно прийти к выводу, что она является достаточной полной, охватывает все возможные ошибки и вектора их применения. Дает детальную характеристику ошибок всех компонентов информационной системы, включая веб-приложения, средства хранения, обработки и отображения данных. Для проектирования инструментов тестирования, предназначенных для обнаружения данных ошибок, необходимо более детально понимать причинно-следственные связи возникновения ошибок, требования к заданию пороговых значений, определяющих факт присутствия программной ошибки. Также целесообразно выработать требования применительно к информационной системе с целью предотвращения появления ошибок на основе ранее полученных аналитических знаний. Для этого приведем расширенную характеристику ошибок в соответствии с классификацией *OWASP Top 10:*

1. Нарушение контроля доступа. Неправильное проектирование архитектуры для системы разграничения привилегий внутри приложений приводит к появлению ошибок нарушения контроля доступа. Сторонний пользователь может

изменить параметры запроса и, если нет соответствующей проверки, получить неавторизованный доступ к ресурсам системы. Зачастую такие параметры присутствуют в URL-строке и html-формах веб-страницы, что дает возможность целенаправленно изменять параметры для реализации процедуры повышения привилегий. В качестве объектов раскрытия данной ошибки могут выступать различные ресурсы системных настроек, файлы и другая конфиденциальная информация, хранящаяся внутри баз данных. Для получения непривилегированного доступа к ресурсам системы используется методика относительного и глобального обхода директорий. Сущность данного подхода состоит в последовательном повышении уровня директории, извлечении файлов, хранящихся внутри данной директории, а также определении структуру корневой файловой системы. В данном случае можно получить доступ не только к ресурсам пользователей, но и к глобальным настройкам операционной системы с целью установления полного контроля над ней. Другой подход подразумевает анализ динамики трансформации различных параметров для определенных ресурсов и пользователей, а также выявление закономерностей в именах параметров и их значениях. Реализуя попытки предугадать значения параметров, можно получить доступ к ресурсам, как обычных пользователей, так и администраторов системы без выполнения авторизации.

2. **Криптографические ошибки.** Большинство веб-приложений связано с обработкой и хранением конфиденциальных данных, начиная от интернет-магазинов, заканчивая крупными корпоративными банковскими информационными системами. Большинство таких систем не используют механизмы на основе шифрования для закрытия важной информации или используют их на завершающих стадиях обработки данных, давая возможность их получения в открытом виде. Такая ситуация достаточно часто встречается в приложениях, использующих базу данных как универсальное хранилище персональных данных. В них используются встроенные функции шифрования и хеширования для шифрования данных, однако это не дает никакой гарантии, что информация не будет извлечена в открытом виде в процессе обработки параметров

веб-сервером. Процесс передачи данных требует использования защищённых протоколом на основе SSL/TLS – протокол https. Это дает возможность заблокировать мониторинг траффика и предотвратить его использование для извлечения параметров запроса.

3. Инъекции. Появление данной группы связано с функционированием информационных систем, использующих различные системы хранения данных на основе реляционных, NoSQL баз данных и протокола LDAP, позволяющих динамически выполнять набор скриптовых и системных команд. Использование самой стратегии инъекции достаточно обширно, а возможности по извлечению данных и выполнению команд являются самыми функциональными. Исследователи относят инъекции к ошибкам с динамической масштабируемостью, так как вектора эксплуатации и воздействия обладают максимальным охватом для информационной системы [128]. Использование данного типа ошибок ставит под сомнение использование простейших методов фильтрации и валидации. В рамках типологизации инъекций принято выделять следующие разновидности ошибок.

SQL-инъекции. Являются самыми распространенными, так как классический подход к построению веб-приложений предусматривает систему хранения данных. В рамках проектирования и разработки веб-приложений в качестве такого хранилища выступают реляционные базы данных (СУБД). Формат SQL инъекции базируется на использование синтаксиса языка SQL, а также применении дополнительного набор параметров, позволяющего разделять фрагменты запроса. Тем самым достигается обход многих ограничений, в частности, механизмов аутентификации. Современные реляционные СУБД обладают обширными возможностями: операции с файловой системой, выполнение системных команд, сетевое взаимодействие, что позволяет выйти за рамки общепринятого подхода, направленного на получение данных, хранящихся и извлекаемых из баз данных. Классическая стратегия использования SQL инъекции реализуется на основе прямого внедрения подзапроса внутрь входного параметра и его последующая отправка в приложение. Более сложные используют двоичную логику, а также различные временные задержки для манипулирования процессом извлечения

данных. Исходя из способов формирования запроса, а также процедур отправки и получения данных посредством инъекции выделяют следующие типы ошибок:

Union (Объединенная). Ошибка данного типа допускает использование единого канала взаимодействия для выполнения инъекции и получения результирующих данных. Основным протоколом взаимодействия является http. Использование типовой union инъекции базируется на специфике построения запросов объединения, допускающих одинаковое число запрашиваемых полей в исходном запросе и в запросе с инъекцией. Для определения точного количества полей в оригинальном запросе используется предикат order by, который может принимать в качестве параметра общее число полей, к которым он может быть применен. Используя union запрос, можно одновременно выполнить, и оригинальный запрос к данным, и запрос с нагрузкой, использующей инъекцию. При этом формирование запросов не ограничивается количеством максимально получаемых строк, хранящихся внутри таблицы базы данных, что позволяет получить полное содержимое полей и записей, используя всего лишь один запрос типа union.

Boolean Blind (Логическая). Основанная идея, лежащая в основе данной ошибки, базируется на использования операций булевой алгебры, в частности, операций конъюнкции и дизъюнкции. Запрос производится путем вставки и выполнении предикатов, содержащих логические выражения, возвращающие истинное (ложное) событие в зависимости от контекста запроса. Для динамического анализа факта присутствия ошибки используются многочисленные синтаксические анализаторы различных форматов представления данных: json, xml, html. Результативность выполнения инъекции подтверждается фактом изменения структуры и содержимого выходных данных. Извлечение данных таблиц и системной информации происходит посимвольно путем пошагового сдвига позиции символа внутри запрашиваемого поля. Определение типа символа происходит путем использования побитовых операций «и» и «или», а также алгоритма бинарного поиска на основе таблицы ASCII-символов.

Time Blind (Временная). Большинство СУБД имеют функционал,

позволяющий формировать временные задержки в процессе выполнения запроса. Сочетая временной подход с булевыми условиями, можно извлекать данные по аналогии с логической инъекций. В данном случае критерием присутствия ошибки является наличие установленной временной задержки, свидетельствующей об успешном выполнении запроса. При этом вычисляется девиация времени выполнения нескольких запросов без инъекции для того, чтобы более четко определить временную задержку между стандартным запросом и запросом с SQL инъекцией.

Error Blind (На основе ошибок). Использование данной ошибки допустимо в тех случаях, когда в веб-приложении в полной мере не реализован перехват исключений, возникающих в результате взаимодействия с СУБД. Это позволяет получать ошибки как часть ответа на запрос доступа к данным. Используя специальным образом структурированный запрос, можно получить значения запрашиваемых полей как часть ошибки. В результате извлекается набор данных по определенным полям в теле ответа с ошибкой, что значительно снижает время на получение требуемой информации из базы данных.

Stack Queries (Разделенная). Возникает в тех случаях, когда драйвер подключения к СУБД позволяет выполнить несколько запросов в рамках одного пакета. Это позволяет объединять оригинальный запрос и запрос с инъекцией. Синтаксическим разделителем запросов языка SQL является символ «;». Данная методика иногда используется в комбинации с инъекций на основе временной задержки, так как не все базы данных поддерживают выполнение таких команд в рамках подзапроса. Механизм эксплуатации данной ошибки обладает расширенными возможностями, позволяющими создавать хранимые процедуры, изменять значения записей таблиц баз данных, а также использовать функционал для доступа к интерфейсам сетевого взаимодействия и файловой системе.

Out of Band (Внешняя). Основой для использования данной ошибки является присутствие нескольких каналов взаимодействия, первый предоставляет веб-приложение, а второй организуется средствами СУБД. Как правило, в качестве вторичного канала используются протоколы smtp, pop3, dns и http. Инъекция

выполняется путем отправки запроса в веб-приложение, после чего запрос интерпретируется СУБД. Далее извлеченная выборка отправляется на другой ресурс, к примеру, в формате сообщения электронной почты. Важно отметить, что dns запрос имеет ограничение длины доменного имени в 64 байта, в связи с этим требуется реализовать блочную обработку данных.

Анализируя разновидности инъекций, видно, что наибольшей скоростью чтения данных обладает объединенная и инъекция на основе ошибок, за счет того, что за один запрос можно извлекать значительный объем данных. Временная и логическая обладают наихудшей производительностью, однако имеют максимальный показатель обхода ограничений безопасности и фильтрации. В целом, использование ошибок должно быть качественно сбалансировано, а именно в тех ситуациях, когда существует возможность использования инъекций, обладающих высокой производительности, целесообразность их использования не вызывает сомнений. В тех случая, когда эти методы не реализуемы, необходимо использовать инъекции на основе булевой алгебры.

Межсайтовый скриптинг. Использование веб-приложений с расширенными пользовательскими возможностями связано с созданием и размещением программного кода в теле веб-страницы, предназначенного для выполнения динамической трансформации элементов. Это реализуется за счет динамического обращения и вставки элементов в DOM-модель веб-страницы. В основе межсайтового скриптинга лежит использование формализованных конструкций встроенного языка JavaScript и их внедрение во входные параметры веб-приложения. Целью данного вектора атаки является поиск фрагментов кода, которые позволят получить доступ к информации, хранящейся внутри веб-браузера, в том числе данные сеансов, размещаемые в cookie. Использование механизмов фрагментирования и трансформации вставляемого кода позволяет использовать механизмы, направленные на обход фильтров безопасности. Современная интерпретация межсайтового скриптинга (МСС) подразумевает использование всех доступных методов встраивания нагрузки в страницу пользователя: таблицы стилей CSS, встраиваемые приложения ActiveX (OLE),

Flash, Silverlight, Java Applet. В соответствии с механизмами внедрения МСС разделяются на следующие категории:

Хранимая. Фрагменты кода МСС используют различные хранилища данных, в частности, файлы и базы данных. Это дает возможность реализовать возможность использования ошибки на постоянной основе. При каждом запросе страницы фрагменты кода извлекаются из хранилища и встраиваются в страницу.

Отраженная. Реализует альтернативные механизмы внедрения кода с использованием ссылок, электронных сообщений и форумов. Для этого достаточно выполнить требуемое действия заложенное внутрь страницы, что позволит автоматически подгрузить произвольный код МСС.

На основе модели документа DOM. Функционал dom-модели, позволяет осуществлять взаимодействие с элементами перенаправления браузера, изменение параметров строки запроса. Это дает возможность использования данных компонентов для вставки сторонних элементов МСС-кода без непосредственного внедрения на веб-страницу.

LDAP-инъекции. Используют синтаксис протокола LDAP, позволяющего осуществлять управление запросами и манипуляцию каталогами X.500. Наиболее распространенные реализации данного протокола Microsoft Active Directory, OpenLDAP и Red Hat Directory Service. Использование LDAP позволяет пользователям осуществлять совместный доступ к ресурсам сети в рамках определенного домена: принтеры, директории, пользователи. Веб-приложения, обеспечивающие интерфейс взаимодействия по протоколу LDAP, позволяют динамически выполнять доступ к ресурсам сети. Стандартная реализация LDAP инъекции подразумевает непосредственную вставку запроса во входной параметр приложения через разделитель. Это дает возможность осуществить доступ к данным и каталогам без соответствующих полномочий. По способу выполнения LDAP инъекция может быть разделена на следующие виды:

Query (На основе подзапроса). Механизм использования ошибки реализуется путем подстановки LDAP запроса в тело одного из входных параметров за счет специального разделителя. Благодаря этому корректируется результирующий

запрос на выборку данных, позволяющий одновременно выполнить стандартный LDAP запрос и подзапрос с инъекцией.

Boolean Blind (Логическая). Аналогично SQL инъекции, принцип данного подхода базируется на операциях булевой алгебры. Использование логических утверждений в рамках подзапроса позволяет проверить факт присутствия искомого утверждения или исключить часть логики запроса. Для обхода элементов аутентификации и авторизации может быть реализовано блочное чтение данных.

Инъекции команд и кода. Представляют собой разновидность системной инъекции, направленной на выполнение динамически сгенерированных конструкций программного кода, а также системных терминальных команд. Такая ошибка имеет место в приложениях, реализующих интерфейсы для запуска процесса командной строки с последующим выполнением команд на основе входных параметров приложений. Между тем многие интерпретируемые языки предоставляют функционал динамического выполнения команд (передача команды как часть строки или запроса), что позволяет получить доступ к функциональным возможностям языка программирования и использовать права доступа веб-сервера. Данная ошибка обладает наибольшим уровнем критичности, а использование механизмов воздействия на ее основе, позволит скомпрометировать всю информационную систему.

NoSQL-инъекции. Существует множество приложений, использующих технологию больших данных. Сама концепция больших данных подразумевает использование распределенной (кластерной) системы хранения данных. В качестве такой системы выступают NoSQL базы данных. Многие NoSQL системы используют псевдоструктурированную модель хранения данных, а механизмы доступа реализованы на основе скриптовых языков программирования для расширения функциональных возможностей базы данных. Существуют СУБД, применяющие SQL подобный интерфейс доступа к данным, в таком случае, используется комбинация NoSQL и SQL инъекции для достижения целей воздействия. Классическая интерпретация NoSQL инъекции основана на методике выполнения команд в основе которой используется фрагментация запросов и

разделение логики обработки параметров приложения. Это дает возможность выполнения стандартных конструкций языка запросов для СУБД, а также расширенных скриптовых команд в рамках одного запроса. В связи с тем, что NoSQL зачастую представляют собой кластерную систему, то инъекция позволяет получить доступ ко всем данным, хранящимся на распределенной файловой системе в пределах этого кластера.

Небезопасная архитектура. Ошибки, возникающие в связи с небезопасной архитектурой веб-приложений, представляют собой обширную группу, связанную с недостаточной проработкой архитектуры приложений с учетом заданных требований безопасности. Отметим, что существует разница между небезопасной архитектурой и реализацией. Мы различаем ошибки проектирования, реализации и разработки на основе существующей архитектуры, так как они имеют разные особенности и способы устранения. Безопасная архитектура может содержать определенные ошибки, связанные с реализацией программных компонентов, которые могут привести к возникновению ошибок и использованы злоумышленниками. В свою очередь небезопасная архитектура не может быть полностью обеспечена защищенной реализацией компонентов системы, так как средства аудита не могут учесть всех особенностей построения веб-приложений. Одним из факторов небезопасного проектирования является отсутствие должного профилирования рисков, присущих разрабатываемому программному обеспечению или системе. Данные риски связаны с потерей конфиденциальных данных и полной компрометацией целевой системы. Как следствие, в процессе проектирования возникает общая неоднозначность в определении того, какой уровень безопасности требуется для данной системы.

4. Нарушение конфигурации системы. Качественная настройка информационной системы требует определения ее компонентов, механизмов взаимодействия и их комплексной конфигурации. Большинство веб-фреймворков используют настройки, которые могут быть доступны только в режиме отладки, а также имена пользователей и паролей по умолчанию, значения которых не удовлетворяют требуемому уровню сложности. Поэтому стандартные настройки

должны быть детально проанализированы и максимально изменены для адаптации работы приложения в режиме базового функционирования. Разработчики и системные администраторы должны использовать комплексный подход к настройке системы и ее отдельных элементов, лежащих в основе конфигурации веб-приложений. Использование ошибок данного типа дает широкий спектр для реализации различных векторов воздействия и может послужить отправной точкой для разрушения принципов надежности информационной системы.

5. **Использование компонентов с ошибками.** В основе построения веб-приложений часто применяются различные фреймворки, системы управления сайтами, а также наборы программных библиотек, предназначенных для расширения функциональности приложения и упрощения процесса проектирования и разработки. Между тем сами компоненты могут содержать ошибки и послужить входной точкой для получения доступа к системе и нарушить существующие принципы функционирования приложения. При выборе какого-либо компонента необходимо использовать самые последние версии и обновления, а также грамотно оценивать степень сложности процесса обновления от одной версии к другой. Все это требует от разработчика четкого понимания функциональных характеристик данных библиотек, а также возможности фрагментировать данные компоненты и исключать второстепенный функционал, который не используется в процессе работы приложений. Важно понимать, что различные ошибки обладают достаточно неоднозначным функционалом. Одни позволяют лишь изменять пользовательские данные, другие способствует повышению системных привилегий и получению доступа к компонентам информационной системы.

6. **Нарушение механизмов аутентификации и управления сессиями.** Использование механизмов аутентификации является основой для построения веб-приложений с логикой разделенного функционала. Основным способом реализация данной концепции в рамках веб-приложений являются механизмы создания сеансов (сессий). Сущность сеанса заключается в создания постоянного

подключения пользователя к приложению. На этапе инициализации соединения для каждого пользователя формируется идентификатор сессии, который сохраняется в веб-браузере. Стандартный сеанс существует до закрытия браузера, а сам процесс аутентификации можно представить в виде двухэтапного алгоритма. В начале пользователь вводить свои аутентификационные данные, затем отправляет их в приложение, после чего инициализируется сессия со своим уникальным идентификатором, который отправляется в виде cookie-заголовка. При последующих обращениях происходить проверка соответствия идентификатора сессии, хранящегося в cookie и на стороне сервера. Если они эквивалентны, то аутентификация прошла успешно, в противном случае нет. Существуют также механизмы аутентификации, предоставляемые средствами веб-сервера (http аутентификация). В данном случае сеанс связи между приложением и пользователем существует только до закрытия веб-браузера. Механизмы эксплуатации, использующие ошибки данного рода, позволяют обходить средства аутентификации и выполнять определенные операции внутри веб-приложений, не имея на это соответствующих полномочий. Исходя из особенностей использования ошибок аутентификации и управления сессиями, можно представить следующие методики выявления данной группы ошибок веб-приложений:

Прямой перебор значений. Используется в тех ситуациях, когда идентификаторы пользователей являются предсказуемыми и обладают слабой грамматической стойкостью. В основе прямого перебора используется подход на основе порождения всех возможных значений из определенного алфавита или словаря. Каждый словарь содержит наиболее распространенные слова и словосочетания, которые используются в качестве паролей, а также идентификационных параметров и являются статистически обоснованными.

Перехват сессий. Идентификаторы сессий хранятся внутри cookie веб-браузера. Выполнения JavaScript сценариев на странице позволяет осуществить извлечения данных идентификаторов с последующим использованием и выполнением произвольных действий от имени пользователя.

Фиксация сессий. Данной ошибке подвержены приложения с нарушением

механизма удаления сессий. В таком случае можно получить идентификатор сессии без аутентификации. Пользователь перенаправляется на ресурс и выполняет аутентификацию, сторонний пользователь получает тот же идентификатор и те же полномочия. Данная ошибка возникает в тех случаях, когда приложение использует тот же идентификатор пользователя при повторной генерации сессии, а также в том случае, когда одному пользователю разрешено инициализировать и поддерживать несколько сеансов соединения одновременно.

Передача сессии. По функционалу имеет некоторое сходство с фиксацией сессии, однако ключевой особенностью является перенаправление пользователя на ресурс с заранее установленным идентификатором сессии. Ошибка имеет место в том случае, если механизмы сессий передают параметр и его значение как часть URL-строки.

7. Нарушение целостности программных компонентов и данных. Нарушения, связанные с целостности данных и программного обеспечения, относятся к программному коду и инфраструктуре, которые не позволяют исключить возможность нарушения целостности. Приложения, использующие подключаемые библиотеки или модули из ненадежных источников, репозиториев и сетей доставки содержимого (CDN) могут быть подвержены возникновению данных ошибок. Небезопасная организация конвейера сборки приложений CI/CD может привести к несанкционированному доступу, выполнению вредоносного кода или компрометации целевой системы. В свою очередь, многие приложения включают функции автоматического обновления, а патчи загружаются без должной проверки целостности и применяются к ранее установленному доверенному приложению. Злоумышленники потенциально могут загружать свои собственные обновления для распространения и запуска потенциально опасных модулей для веб-приложений. Другой особенностью является использование веб-приложением механизмов сериализации и десериализации данных, что дает возможность изучения и дальнейшего использования данных механизмов злоумышленником с целью получения доступа к приложению, содержащему такого рода ошибки.

8. Отсутствие журнала безопасности и мониторинга действий. Протоколирование всех действий и ошибок, возникающих в веб-приложении, играет особую роль для решения задач по предотвращению, локализации воздействиях, снижению уровня эскалации, своевременному реагированию и противодействию активным нарушениям работы приложения и связанных с ним программных и инфраструктурных элементов. Недостаточное логирование, ведение журнала безопасности, мониторинг активности данных, циркулирующих внутри приложений, может привести к компрометации системы и утере критически важных данных информационных систем, построенных на основе веб-технологий.

9. Межсайтовая подделка запросов на стороне сервера (SSRF). Программная архитектура, используемая для построения веб-приложений, допускает выполнение запросов на сторонние сервисы. Ошибки SSRF возникают всякий раз, когда веб-приложение обращается к удаленному ресурсу без проверки URL-адреса. Такая ошибка позволяет злоумышленнику заставить приложение отправить сформированный запрос в произвольное место назначения по URL, которое находится под его контролем. Поскольку современные веб-приложения предоставляют конечным пользователям удобные механизмы для загрузки сторонних данных и подключения через интерфейсы взаимодействия (API) других приложений, обращение к ресурсам URL внутри веб-приложений становится обычным сценарием. Особую сложность в процессе обнаружения SSRF представляют облачные сервисы, имеющие сложную архитектуру и расширенный API, через который они могут взаимодействовать с другими приложениями.

1.2. Применение методов математического моделирования и интеллектуального анализа данных для поиска и локализация программных ошибок

Разработка эффективных инструментов анализа программных ошибок выявления нестабильных модулей, выступающих в качестве составных

компонентов архитектуры приложений, является важным направлением современных теоретических и практических исследований. В качестве хорошо апробированных инструментов проведения подобных исследований выступают методы математического моделирования на основе нейронных и байесовских сетей, сетей Петри, конечных автоматов, нечетких множеств, элементов стохастического моделирования. Применение данных инструментов позволяет реализовать методологию временного тестирования. Это дает возможность отслеживать трансформацию ошибок при переходе от одного момента времени к другому для заданного интервала. В рамках проведения тестирования, формализованный процесс анализа ошибок представляет набор взаимосвязанных компонентов модульного тестирования. Общая спецификация модульного тестирования имеет следующий вид:

$$\bigwedge_{M_i \in TestCase} M_i(x_1, x_2, \dots, x_n) = Ret_{expect}(M_i), \tag{1.1}$$

где $M_i(x_1, x_2, \dots, x_n)$ – результат выполнения некоторого тестового метода $TestCase$ с n набором входных параметров, $Ret_{expect}(M_i)$ – возвращаемое значение тестового метода. Применение операции конъюнкции дает возможность утверждать, что правильность всего тестового набора будет тогда и только тогда, когда все ожидаемые и возвращаемые значения будут соответствовать друг другу.

Процесс тестирования программ можно представить в виде некоторого конечного автомата. Данный автомат представляет собой некоторую абстрактную модель, описывающую процесс изменения состояния объекта в зависимости от его текущего состояния и входных параметров. Для описания конечного автомата применительно к процессу тестирования, его можно представить в виде следующей совокупности объектов $A = \left(S, s_0, \delta, S_{fin}, D\right)$, где S – множество конечных состояний рассматриваемой тестируемой системы, s_0 – начальное состояние, δ – функция переходов, аргументами которой являются: текущее состояние автомата, входной символ, новое состояние, S_{fin} – множество заключительных (допускающих) состояний (является подмножеством множества S), D – алфавит автомата. Если формально представить автомат в виде графа, δ может быть

представлена в виде дуг, соединяющих состояния автомата. Если q – состояние автомата, a – входной символ, то $\delta(q, a)$ – состояние p для которого существует дуга a, ведущая из a в p. Под состоянием тестируемой системы понимается фрагмент или модуль программного кода, характеризующий логически завершенный результат. В классической интерпретации программа представляет собой множество программных модулей, входящих в ее состав. С целью повышения степени абстракции в модели тестирования на основе конечного автомата, целесообразно использовать метод, основанный на группировке состояний. Применение данного подхода дает возможность укрупнения состояний за счет их группирования. Группой состояний является расширенный автомат A', включающий в себя состояния $S' \subseteq S$ из базового автомата A. Идея подхода, основанного на группировке, заключается в том, что все переходы в состояния S' будут выполняться только из состояния $s' \in S'$. Множество заключительных состояний S'_{fin}, соответствующих группе S', будет использоваться для формирования связей между несколькими группами состояний $S'_1, S'_2, \dots, S'_n$.

Моделирование процессов тестирования недетерминированных программ на основе конечного автомата в классической интерпретации довольно сложным в реализации, так как не учитывает случайное поведение компонентов программ. В связи с этим для анализа функциональных процессов, протекающих внутри приложений целесообразно расширить представление модели тестирования на основе автомата за счет добавления случайности в функцию перехода δ за счет определения вероятности перехода автомата из одного заданного состояния в другое. Далее в диссертационном исследовании будем рассматривать понятие вероятностного конечного автомата. В простейшем случае для определения факта присутствия программной ошибки можно определить следующую функцию перехода между состояниями:

$$\delta: S \times D \times P \rightarrow 2^S, \tag{1.2}$$

где 2^S – множество всех подмножеств множества состояний S, показывающее недетерминированность перехода.

Выражение (1.2) характеризует факт присутствия недетерминированных переходов из некоторого состояния $s_1 \in S$ с вероятностью $p \in P$ и применением элемента алфавита $d \in D$ в целевое состояние $s_2 \in S$. Транзитивная вероятность определяется для всех кортежей состояний, включающих текущее и целевое состояние. В данном случае переходную вероятность можно представить в матричной форме $P_{ij} = \| p_{ij} \|$, где $1 \leq i, j \leq |S|$, а каждый элемент матрицы p_{ij}, описывает вероятность перехода из состояния i в j. Для данной стохастической матрицы, описывающей конечный автомат, справедливы классические аксиомы теории вероятности

$$0 \leq p_{ij} \leq 1,$$
$$\forall i, 1 \leq i \leq |S|: \sum_{j=1}^{|S|} p_{ij} = 1 \tag{1.3}$$

Элементы вероятностной матрицы могут характеризовать априорные и апостериорные значения распределения вероятностей. Формирование априорных вероятностей производится на основе предварительных предположений о работе тестовой системы и описывает вероятностные связи между состояниями конечного автомата. Данные вероятности задаются на основе статистических знаний о результатах тестирования однородных приложений и показывают вероятности обнаружения ошибок, характерные определенным группам программных приложений. Апостериорные вероятности описывают транзитивные вероятности состояний автомата, определяемые после выполнения фиксированного цикла тестирования. Они описывают фактические вероятности переходов для конечного автомата. В тоже время, рассматривая процесс тестирования в виде временного интервала, апостериорные вероятности для времени t могут быть интерпретированы как априорные вероятности для $t + 1$. В процессе тестирования мы предполагаем, что переход из текущего состояния в следующее происходит без учета посещенных состояний, и рассматриваем только переходы вида (1.2) для двух смежных состояний S_i и S_j. Это достигается за счет предположения о марковском свойстве. Сущность данного свойства заключается в том, что для

каждого рассматриваемого состояния $S_i \in S$ в заданное временя t все смежные состояния S_j и S_k будут являться условно независимыми при наличии S_i

$$P\left(S_j \Rightarrow S_k | \xi(t) = S_i\right) = P\left(S_j \Rightarrow S_i \middle| \xi(t) = S_i\right) P(S_i \Rightarrow S_k | \xi(t) = S_i), \quad (1.4)$$

где $\xi(t)$ функция состояния автомата для времени t, $S_j \Rightarrow S_k$, $S_j \Rightarrow S_i$, $S_i \Rightarrow S_k$ – переходы из состояний j в k, j в i, i в k соответственно, $P\left(S_j \Rightarrow S_k | \xi(t) = S_m\right)$ – вероятность перехода из состояния j в k, учитывая что автомат находился в состоянии S_m.

Применение теории вероятностных конечных автоматов в процессе тестирования на практике доказывает свою высокую эффективность, однако не дает возможности прогнозировать возникновение программных ошибок, а лишь позволяет осуществлять вычисление апостериорных оценок и корректировать состояния автомата. Это ограничивает возможности применения вероятностного конечного автомата в процессе обнаружения аномалий и не позволяет определить причинно-следственные связи в процессе трансформации ошибок с течением времени. Автомат может лишь рассматриваться как составной элемент, используемый в процесс построения тестовых цепочек для методов черного и серого ящика.

Для обхода ограничений конечных автоматов на практике используется моделирование на основе нейронных сетей. Стратегия вычислений нейронных сетей базируется на обучении. Сущность процедуры обучения заключается в выстраивании синаптических весов искусственной нейронной сети с целью определения требуемой структуры, описывающей процедуру взаимодействия нейронов. Трансформация синаптических весов представляет собой классический подход, применяемый для настройки нейронной сети. Основными свойствами, присущими нейронным сетям являются: нелинейность, адаптивность, отображение входной информации в выходную. Свойство *нелинейности* подразумевает, что по своей природе искусственные нейроны могут быт, как линейными, так и нелинейными. Нейронные сети, построенные в виде цепочки нелинейных нейронов, являются нелинейными. Свойство нелинейности становится особенно

важным, если сам входной сигнал изменяется по нелинейному закону, в частности, речеподобный сигнал. *Адаптивность* заключается в гибкой адаптации синаптических весов к исполняемой среде. Данная особенность наиболее проявляется при работе с нестационарными процессами, статистические данные в которых трансформируются с течением времени, что дает возможность изменения синаптических весов нейронной сети в реальном времени. *Отображение входной информации во выходную* дает возможность использования стратегии обучения, основанной на обучении с учителем. Именно это позволяет производить изменение синаптических весов на основе маркированных тестовых примеров, что в свою очередь способствует снижение расхождения выходного сигнала.

Применительно к процессу тестирования, нейронная сеть выступает в виде набора взаимосвязанных тестовых механизмов, представленных в виде искусственного нейрона и образующих единую логическую структуру. Сигналы, поступающие на входы каждого нейрона, образуют некоторую каскадную модель, описывающую процесс передачи информативных сигналов от одного нейрона к другому. Для построения математической модели классической нейронной сети тестирования необходимо определение двух составляющих: линейную комбинацию входных воздействий и функцию активации. В данном случае подразумевается, что выходные сигналы нейронов имеют непосредственную зависимость от функции активации

$$u_k = \sum_{i=1}^{n} w_{ki} x_i \,, \tag{1.5}$$

$$y_k = \varphi(u_k + b_k), \tag{1.6}$$

где $x_i = \{x_1, x_2, \dots, x_n\}$ – входные сигналы, $w_{ki} = \{w_{k1}, w_{k2}, \dots, w_{kn}\}$ – синаптические веса для k нейрона, u_k – линейная комбинация входных воздействий, $\varphi(u_k + b_k)$ – функция активации, y_k выходной сигнал искусственного нейрона.

Постсинаптический потенциал (потенциал активации) определяется как $v_k = u_k + b_k$. В интересах тестирования в качестве функции активации целесообразно применение сигмоидальной функции. Данная функция является

быстровозрастающей и устанавливает баланс между линейным и нелинейным поведением. Одной из наиболее применимых сигмоидальных функций является логистическая функция, специфика которой задается с помощью следующего соотношения:

$$\varphi(v_k) = \frac{1}{1 + e^{-av_k}}, \tag{1.7}$$

где a – коэффициент наклона сигмоидальной функции.

Классическая область значения данной функции определяется отрезком $[-1; 1]$, а сама функция является нечеткой функцией потенциала активации. Пороговую функцию применительно к процессам тестирования можно представить в виде следующего выражения:

$$\varphi(v_k) = \begin{cases} 1, \text{если } v_k > 0 \\ 0, \text{если } v_k = 0 \\ 1, \text{если } v_k < 0 \end{cases} \tag{1.8}$$

Для моделирования процессов тестирования используются многослойные нейронные сети прямого распространения. Данная сеть характеризуется наличием некоторого множества скрытых слоев, целью которых является выделение статистических зависимостей между входным и выходным слоем. Принцип работы процесса тестирования на основе нейронной сети можно представить в следующем виде. Источники входного слоя генерируют соответствующие тестовые шаблоны (элементы активации), которые формируют входной сигнал, передаваемый на первый скрытый слой. В свое время выходные данные скрытого слоя являются входными для последнего слоя. По аналогии с конечными автоматами предполагается, что нейроны текущего слоя имеют связь только с предыдущим слоем. Множество выходных сигналов описывают реакцию целевой программы на определенный набор тестовых цепочек, сформированных на первом слое нейронной сети. Для построения эффективных алгоритмов обнаружения аномальных ошибок приложений в процессе тестирования используется обучение нейронной сети. В общем случае обучение направлено на корректировку синаптических весов и пороговых значений для нейронной сети, это происходит за счет получения знаний в процессе тестирования различных приложений. На

практике для обучения многослойных нейронных сетей используется алгоритм обратного распространения ошибки. Рассмотрим представление процесса тестирования в виде рекуррентной нейронной сети (РНН).

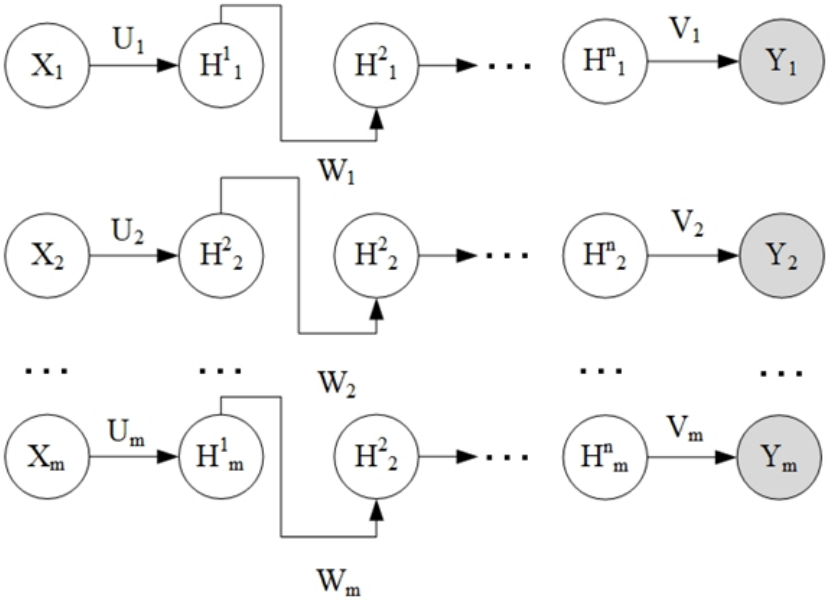

Рисунок 1.3. Развернутая во времени рекуррентная нейронная сеть

Для обучения рекуррентной сети фаззинга необходимо определить обучающую выборку (X, Y), где X – множество тестовых генераций, полученных по результатам фаззинга, Y – исходы событий тестирования, характеризующие результаты применения конкретной выборки для анализа ошибок типа «инъекции». Для реализации обучения РНН фаззинга необходимо произвести расчет ее параметров для каждого следующего состояния $t+1$. Следовательно, для соответствующего момента времени $t+1$ развернутой нейронной сети можно определить уравнения обновления параметров сети x_t, h_t, y_t в прямом направлении с учетом соответствующих им матриц весов U, W и V

$$\begin{gathered} h_t = tahn(UX_t + WH_{t-1}), \\ y_t = Vh_t \\ \widehat{y_t} = softmax(Vh_t), \end{gathered} \tag{1.9}$$

Выражение (1.9) соответствует алгоритму прямого распространению РНН. Для того, чтобы сформулировать алгоритм обратного распространения определим функцию потерь. Пусть $x: \{x_1, x_2, \ldots, x_n\}$ представляет собой множество тестовых генераций, поступающих на вход РНН. Тогда полную потерю тестовой генерации x в совокупности с у можно определить в виде логарифмической функции потерь

$$L = L(x_t, y_t) = -\sum_t ln\, P(y_t | x_1, x_2, \dots, x_t) \tag{1.10}$$

В таком случае градиент по выходам нейронной сети Y_t можно определить в следующем виде [162]:

$$\nabla_{\widehat{y_t}} L = \frac{dL}{d\widehat{y_t}} \tag{1.11}$$

Двигаемся от выхода Y_τ к скрытому слою H_τ. Так как скрытый слой будет иметь лишь одного предка, соответствующего выходной переменной Y_τ, градиент функции потерь для скрытого слоя H_{t+k} получим в виде следующего произведения:

$$\nabla_{h_\tau} L = V^T \nabla_{\widehat{y_\tau}} L \tag{1.12}$$

Тогда рекурсивная процедуру обратного распространения применительно к скрытым слоям $H_{t:\tau}$ может быть выполнена за счет рассчета градиента $\nabla_{\mathrm{H}_t}\mathrm{L}$:

$$\nabla_{H_t} L = \frac{dL}{dW} = \frac{dL}{d\widehat{y_t}} \sum_{k=1}^{\tau} \frac{d\widehat{y_t}}{dh_t} \frac{dh_\tau}{dh_k} \frac{h_k}{dW} \tag{1.13}$$

Якобиан для функции h_τ, характеризующей переходы между скрытыми слоями, можно записать в виде следующего произведения:

$$\frac{dh_\tau}{dh_k} = \prod_{m=k+1}^{\tau} \frac{dh_m}{dh_{m-1}} \tag{1.14}$$

Градиент, стоящий под знаком суммы можно выразить через соответствующую матрицу весов W и производную активационной функции $\mathrm{h_t}$ в соответствии с выражением (1.9)

$$\frac{dh_m}{dh_{m-1}} = W^T diag((\mathrm{h_t})') = W^T diag(1 - (\mathrm{h_t})^2) \tag{1.15}$$

Тогда рекурсивное выражение (1.13), соответствующее алгоритму обратного распространения ошибки $L(\mathrm{x}_t, \mathrm{y}_t)$ приведем к следующему виду:

$$\nabla_{H_t} L = \frac{dL}{dW} = \frac{dL}{d\widehat{y_t}} \sum_{k=1}^{\tau} \frac{d\widehat{y_t}}{dh_t} \left(\prod_{m=k+1}^{\tau} \frac{dh_m}{dh_{m-1}} \right) \frac{h_k}{dW}, \tag{1.16}$$

$$\frac{dL}{dW} = \frac{dL}{d\widehat{y_t}} \sum_{k=1}^{\tau} \frac{d\widehat{y_t}}{dh_t} \left(\prod_{m=k+1}^{\tau} W^T diag(1 - (h_m)^2) \right) \frac{h_k}{dW}.$$

Из выражения (1.16) следует, что алгоритм обратного распространения будет иметь сложность $O(\tau)$ и напрямую зависеть от числа срезов τ на которые развертывается рекуррентная нейронная есть. Преимуществом нейронных сетей в сравнении с другими средствами интеллектуального анализа данных является возможность нейронной сети выступать в виде детерминированного автомата и нечеткой системы одновременно. Данная концепция дает возможность формирования новых статистических оценок в результате работы сети, как следствие, применения процедур обучения сети. Нейронная сеть обладает высокой степенью распараллеливания вычислительных процессов, так как структурно данная сеть состоит из совокупности нейронов. В таком случае значения градиентов могут быть рассчитаны для каждого нейрона независимо.

Рассмотренные математические модели находят широкое применение в работах отечественных и зарубежных ученых, специализирующихся на моделировании процесса тестирования, поиска программных ошибок и выявлении аномального поведения внутри приложений.

В исследованиях А.А. Браницкого [110] и Т.И. Булдаковой [111] рассматриваются вопросы адаптации нейронечетких систем в процессе построения систем обнаружения вторжений (СОВ). Основная стратегия работы СОВ направлена на реализацию математических методов предотвращения недокументированных действий (атак) в рамках информационной системы. Модель СОВ представляет собой множество ситуаций $X = \{X_1, X_2, \dots, X_n\}$, описывающих воздействие на компоненты информационных систем, $X_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$ – вектор признаков, который описывает i событие, n – общее число возможных угроз, $Y = \{y_1, y_2, \dots, y_m\}$ множество защитных действий, направленных на предотвращение недокументированных действий, m – общее число предлагаемых защитных мер. В основе моделирования СОВ лежит концепция интеллектуального анализа данных, позволяющая решать слабоформализованные задачи по выявлению вторжения в компьютерные

системы. В рамках решения задач построения СОВ используется комбинирование математического аппарата нечетких множеств и искусственных нейронных сетей.

Использование нечетких нейронных сетей позволяет решить ряд задач, связанных с повышением эффективности обнаружения вторжений и реализовать возможность представления атак и применяемых средств защиты в виде набора нечетких правил. Для решения данных задач рассматривается целевая зависимость $y^*: X \to Y$, значения которой определены в обучающей выборке $X^r = [(x_1, y_1), \dots, (x_r, y_r)]$. Установление данной целевой зависимости позволяет определить событие атаки по его признакам и выработать ряд защитных мер. Это может быть достигнуто путем применения нечетких правил $R = \{R_1, R_2, \dots, R_2\}$ следующего вида [107]:

$$\begin{gathered} R_1\text{: если } x_1 \in A_1^1,\dots,x_n \in A_1^n\text{, то } Y \text{ есть } y_1, \\ R_2\text{: если } x_1 \in A_1^1,\dots,x_n \in A_2^n\text{, то } Y \text{ есть } y_2, \\ \dots \\ R_k\text{: если } x_1 \in A_{k_1}^1,\dots,x_n \in A_{k_n}^n\text{, то } Y \text{ есть } y_m, \end{gathered} \tag{1.17}$$

где $A_{k_1}^1$ – нечеткие множества, элементы которого характеризуют проведение атаки, $k = k_1, k_2, \dots, k_n$.

Для описания принадлежности выборки $x \in X$ к нечеткому множеству A будем использовать функцию принадлежности $\mu_A(x): X \to [0,1]$. В этом случае получим при $\mu_A(x) = 1$ полную принадлежность, $0 < \mu_A(x) < 1$ частичную принадлежность и при $0 = \mu_A(x)$ отсутствие принадлежности x нечеткому множеству A. Для определения утверждений вида «x почти равно y» или «x значительно больше y» целесообразно ввести понятие нечеткого правила. Правило R представляет собой нечеткое подмножество, формализованное в виде классического декартового произведения $X \times Y$ и может быть описано в виде следующего выражения:

$$R \subseteq X \times Y = \sum_{X \times Y} \frac{\mu_R(x, y)}{(x, y)} \tag{1.18}$$

Результирующая модель нечеткой нейронной сети СОВ с сигмоидальной функцией активации, предложенная Т.И. Булдаковой, иметь следующий вид

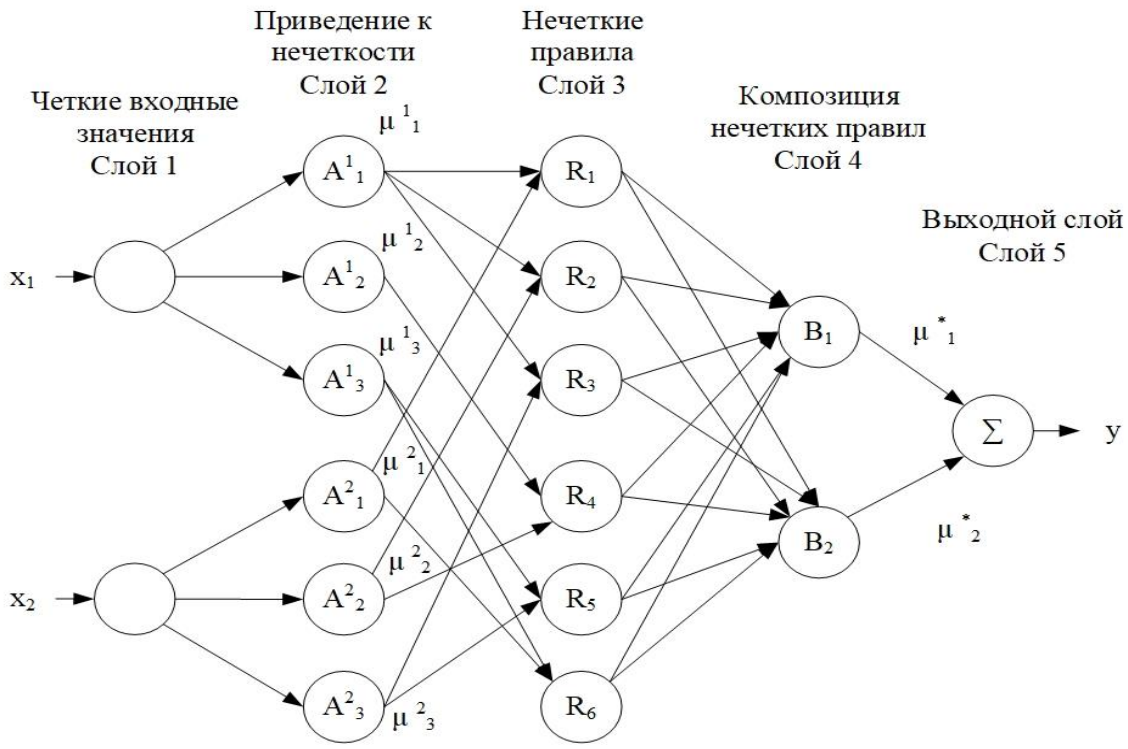

Рисунок 1.4. Нечеткая нейронная сеть СОВ

Рассматривая нечеткую нейронную систему с позиции эквивалентности, она представляет собой классическую многослойную искусственную нейронную сеть, адаптированную к применению нечеткой модели вывода. Рассматривая процедуру нечеткого вывода с точки зрения системного подхода, можно выделить следующие составные этапы. Первый этап характеризуется введением нечеткости на основе функций принадлежности $\mu_{A_1^1} - \mu_{A_3^1}, \mu_{A_1^2} - \mu_{A_3^2}$, описывающей процесс трансформации четких входных параметров x в показатель истинности для множества нечетких правил R. Далее на основе критерия истинности функций $\mu_{A_i^2}$ и $\mu_{A_i^2}, i = 1,2 \dots n$ для каждого из представленных нечетких правил $R_i \in R$ определяется заключение. Третий этап определяется композицией заключений для каждого из правил $R_i \in R$ на основе вычисления функций принадлежности μ_i^* для порождения нечеткого множества $B = \{B_1, B_2, \dots, B_n\}$. На завершающем этапе определяется композиция подмножеств $B_i \in B$, далее осуществляется приведение к четкости в выходном слое, что в свою очередь обеспечивает формирование четкого выходного значения y.

В основу формирования нечетких правил положены экспертные знания в области анализа вторжений, что в свою очередь обеспечивает целесообразность использования алгоритма обучения с учителем, в частности, метода минимизации среднеквадратической ошибки. Целью обучения СОВ, представленной в виде нейронечеткой системы, является настройка весов связей (коррекция правдоподобия нечетких правил) и устранение конфликтности внутри самой системы нечетких правил. Данная система работает в случае наличия апостериорных данных, так и без таковых. В последнем варианте подразумевается, что при достаточной полноте обучающей выборке, сеть самостоятельно трансформирует закономерности, скрытые в обучающих выборках, в множество нечетких правил, используемых в процедуре логического вывода.

В работах ученого-исследователя И.В. Котенко [110,150,151] рассматриваются вопросы обнаружения сетевых воздействий на основе нейронных и нейронечетких классификаторов. В основе исследования положена идея детектирования сетевых атак на основе анализа сетевого соединения нескольких удаленных узлов компьютерной сети. Структура процесса обнаружения формально имеет несколько основных шагов. На первом шаге происходит фильтрация сетевых пакетов, поступающих с агентов-сниферов, установленных на узлах сети. Далее пакеты агрегируются для последующего хранения с целью их семантического анализа и обработки. На завершающем этапе происходит формирование результирующего уведомления, характеризующего типы атак, применяемых в процессе проведения вторжения. За основу построения математической модели обнаружения воздействий взята стратегия композиции моделей нейронных и иммунных сетей, а также нейронечетких классификаторов. В рамках стандартного подхода к разработке механизмов обнаружения атак предложена система, состоящая из многослойной нейронной сети с единственным скрытым слоем. Любая сеть состоит как минимум из трех слоев, соединенных по классической схеме, а именно выходы нейроном соединяются с входами нейронов другого слоя. Первый слой (внешний) характеризуются вычисленными значениями компонентов сетевого соединения и отвечает за распределение входных сигналов $X^1 =$

$(x_1^1, \dots, x_n^1)$, где $n = 29$ – представляет собой фиксированное число атрибутов сетевого трафика. Каждый элемент $x_i^1 \in X^1$ является вещественным числом, определенным в рамках отрезка $[0,1]$. Число скрытых слоев определяется на основе эвристического анализа в процессе построения модели нейронной сети. Внешний слой нейросети создает на входе элемента активации для каждого нейрона последующего слоя сигнал, имеющий вид

$$x_i^2 = \omega_{i1}^2 \cdot x_1^1 + \dots + \omega_{i29}^2 \cdot x_{29}^1 + \theta_i, i = 1, \dots 20 \quad (1.19)$$

где ω_{ij}^2 – синаптические веса для X^1, θ_i – параметр смещения скрытого нейрона.

Выходной слой представляет собой один нейрон, трансформирующий входные сигналы в две категории: первая определят факт присутствия атаки, вторая нормальное сетевое соединение. При этом формирование входного сигнала X^3, поступающего на вход нейрона выходного слоя, имеет следующее представление:

$$X^3 = \omega_1^2 \cdot \varphi(x_1^2) + .. + \omega_{20}^2 \cdot \varphi(x_{20}^2) + \theta, \quad (1.20)$$

где $\varphi(x) = th(x)$ – сигмоидальная функция активации, ω_j^2 – синаптические веса нейронов второго слоя, θ – параметр, характеризующий смещение выходного нейрона.

При этом, если значение выходного нейрона $(\varphi(X^3) > 0)$ находится в положительной области значений, то в канале соединения присутствует атака, в противном случае имеем нормальное сетевое соединение. Для оптимизации процедур распознавания определенного типа атак формирование нейросетевых детекторов происходит независимо, а процесс классификации выполняется параллельно.

В рамках моделирования иммунных детекторов за основу взята модель Хофмаера-Фореста, расширенная двухэтапной фазой обучения и способностью общего использования накопленной статистической информации. В общей концепции, каждый детектор представляется в виде «карты Кохонена», представляющей собой двухслойную сеть. Входной слой отвечает за распределение сигнала, представляющего собой вектор характеристик $X = (x_1, \dots, x_{29})^T$. Выходной слой можно представить в виде квадратной решетки

размером 15x15. Любая составляющая входного сигнала $x_i, i = 1, \dots 29$ имеет непосредственную связь с соответствующим нейроном выходного слоя и синаптическим весом $\omega_{i,j,k}$. Для обучения сети Кохонена используется стратегия конкурентного обучения. В процессе поступления сигнала на вход сети Кохонена наибольшим приоритетом обладает нейрон, множество весов которого минимально отличаются от множества входных сигналов. Для нейрона с наивысшим приоритетом расстояние между вектором входных сигналов и вектором весов данного нейрона $W_{p,q} = \left(w_{1pq}, \dots, w_{29pq}\right)^T$ определяется как минимальное из всех имеющихся расстояний

$$d\left(X, W_{p,q}\right) = min\, d\left(X, W_{i,j}\right), i = 1, \dots, 15, j = 1, \dots 15 \qquad (1.21)$$

В рамках выполнения процесса обучения в окрестностях нейрона с наибольшим приоритетом формируется множество нейронов, веса которых обладают наименьшим отклонение от веса нейрона с наивысшим приоритетом. Корректировка весов векторов, расположенных в данной окрестности, производится на основе правила Кохонена

$$W_{kl}(t+1) = W_{kl}(t) + \gamma\left(X - W_{kl}(t)\right), \qquad (1.22)$$

где γ – коэффициент, определяющий скорость обучения, t – номер текущего шага итерации. В процессе обнаружения и распознавания атак используется несколько иммунных детекторов, способных независимо обучаться на основе различных выборок, формируемых из обучающих данных. Использование данного подхода позволяет оптимизировать процесс создания уникальных классификаторов, способных подстраиваться на обнаружение определенного типа воздействий. В качестве верификации классификаторов используется критерий отрицательного отбора, сущность которого основана на подаче на вход детекторов заведомо корректных пакетов сетевого трафика. Если детекторы определили, что данные пакеты обладают признаками нормально трафика, то они будут использованы в процессе анализа, в противном случае детекторы проходят повторное обучение.

В рамках применения нейронечетких классификаторов к анализу воздействий используется пятислойная нейронная сеть прямого распространения

на основе нечеткого вывода с применением алгоритма Такаги-Сугено. В качестве входных сигналов для данной сети выступают числовые значения параметров сетевого траффика, а выход сети – результирующие параметры, характеризующие процесс идентификации сетевого соединения. Каждое значение, поступающее на вход сети, представляется в виде пяти термов: малое, небольшое, среднее, достаточно большое, большое. Входной слой используется для введения фаззификации входных параметров сети, а также выставляет значения нечетких термов для них. В качестве функции принадлежности термов, используется колоколообразная функция $\left(1+\left|\frac{x-c}{a}\right|^{2b}\right)^{-1}$. Выходными параметрами для данного слоя выступают значения функции принадлежности для определенных входных значений. На следующем слое происходит определение нечетких правил, после чего вычисляется произведение входных сигналов или определение минимума. В рамках третьего слоя происходит нормализация выходов со второго слоя. На следующем слое оценивается вклад нечетких правил в формирование выходного сигнала. Финальный слой состоит из одного элемента и объединяет результаты по каждому из правил.

Научные исследования П.Д. Зегжды и С.С. Корта [94] посвящены рассмотрению процессов детектирования аномальных ситуаций, возникающих в процесс функционирования приложений на основе математического аппарата цепей Маркова. Основой предложенного подхода является системная модель S, характеризующая нормальное поведение приложения. В качестве анализируемых событий, возникающих в процессе выполнения программы, используется понятие системных вызовов, образующих множество возможных состояний $\{S_k\}_{k=1}^{n}$. В большинстве ситуаций состояния рассматриваются в рамках выполнения некоторой программной операции: чтение файлов, работа с системными ресурсами. Процедура обучения аномального поведения приложений основывается на математическом аппарате цепей Маркова. В процессе построения марковской цепи при переходе между двумя состояниями S_k и S_{k+1} будем полагать, что каждое из состояний характеризуется параметром «окна» w, задающим число

переменных для каждого из состояний. Используя предположение о марковости, установим, что каждое новое состояния S_{k+1} , будет иметь непосредственную зависимость от предыдущего состояния S_k. Для определения максимально допустимого числа состояний модели S зададим следующие параметры: Pr – множество допустимых операций анализируемого приложения, O – множество операций, характеризующих начальное состояние. Тогда размерность модели S можно определить в виде следующего выражения:

$$|S| = N^w, \tag{1.23}$$

где $N = |Pr \cup \{O\}|$ мощность множества.

Для оценки случайности «окон» каждого из состояний модели S_k используется регулярное распределение. Следовательно, для выбора оптимального значения «окна» требуется найти модель S_k с наибольшим показателем регулярности распределения. Для оценки регулярности рассмотрим применение условной энтропии. При этом регулярность распределения и условная энтропия имеют обратную зависимость – чем больше показатель регулярности, тем меньше значение условной энтропии. Характеристика регулярности распределения может быть выражена через соответствующую условную энтропию $H_w(x, y)$:

$$\begin{aligned} H_w(x,y) &= -\sum_{x \in Pr, y \in Pr^{w-1}} P(x,y) \log_2 P(x|y) \\ &= -\sum_{(x,y) \in D} \frac{1}{|D|} \log_2 P(x|y), \end{aligned} \tag{1.24}$$

где $P(x, y)$ – вероятность наступления обоих событий x и y, $P(x|y)$ условная вероятность появления события x при условии наступления события y, D – множество всех возможных подмножеств рассматриваемой последовательности аномальных операций приложений, $|D|$ – мощность множества D.

Для получения оптимального значения w целесообразно вычислить энтропию последовательности из обучающей выборки наибольшей размерности. Последовательно вычисляя энтропию для некоторого числа «окон», выбрать минимальное из полученных значений, которое и будет соответствовать оптимальному значению «окна». Отметим, что процедура поиска оптимального

значения для w требует циклического перестроения марковской цепи, что накладывает ряд ограничений, связанных с временной сложностью алгоритма. На практике использование больших значений $w > 10$ значительно усложняет процедуру поиска w, так как требует перестроения цепи Маркова. Для этого становится целесообразно использовать параллельную процедуру поиска оптимального значения w, в таком случае построение цепи Макова для каждого из «окон» будет выполняется независимо.

Для выделения аномального поведения при функционировании приложений используется марковская цепь. Алгоритм классификация аномалий происходит за счет вычисления вероятностей P параметров исследуемой последовательности операторов программы. Сущность данного алгоритма основывается на определении формулы для вычисления условной вероятности, описывающей вероятность перехода из состояния s_1 в s_3 через некоторое промежуточное состояние s_3

$$P\big((s_3|s_2)\big|s_1\big) = P(s_2|s_1) \times P(s_3|s_2) \quad (1.25)$$

Распределение вероятностей $P(s_1)$, соответствующее начальному состояния s_1 вычисляется по результатам реализации процедуры обучения. Условие получения аномального поведения связано с превышением значения вероятности некоторого порогового значения r

$$P < r \quad (1.26)$$

Из формулы (1.26) можно получить характерное следствие – марковская цепь описывает поведение программы, которое имеет признаки аномального функционирования, только в том случае, если вероятность переходов между соседними срезами марковской цепи превышает заданное ранее значение параметра r.

Рассмотренные математические методы детектирования программных ошибок и аномального поведения доказывает обоснованность и целесообразность применения данных подходов. Анализируя стратегию тестирования в виде непрерывного процесса, можно установить, что эффективность выявления нестабильных компонентов напрямую связана с разбиением процесса

тестирования на временные интервалы (срезы). Данная особенность характерна для приложений с непрерывной поддержкой и расширением функционала. В данных условиях обоснованность применения математических моделей, обладающих памятью, не вызывает сомнения. Алгоритмические и инструментальные средства, основанные на применении рассмотренных математических моделей, обладают возможностью прогнозирования факта присутствия программных ошибок в определенных группах приложений, а сам процесс тестирования становится более гибким и структурированным. Использование эффективных численных методов в рамках применения алгоритмов обучения способствует формированию критериев надежности программ, более детальной проработке механизмов порождения тестовых активов и шаблонов тестирования программ на основе эмпирических знаний. С точки зрения математического моделирования процессы детектирования ошибок могут рассматриваться, как в виде дискретных процессов с фиксированным числом допустимых состояний, так и в виде непрерывного (гауссовского) процесса. Такие процессы устанавливают стохастические связи между элементами тестирования и направлены на формирование причинно-следственных связей между событиями сканирования и эксплуатации. Использование данного подхода доказывает целесообразность применения алгоритмов искусственного интеллекта, позволяет в полной мере адаптировать алгоритмическую базу для тестирования широкого спектра программных ошибок, а также расширять классические подходы, применяемые в качестве составных компонентов тестовых систем.

1.3. Инструменты статических и динамических байесовских сетей для анализа стохастических процессов и их адаптация к процессам фаззинг-тестирования веб-приложений

Тестирование представляют собой сложный вероятностный процесс, определяющий критерии и механизмы обнаружения программных ошибок. Несмотря на научно-практическую значимость с точки зрения математического моделирования, данные процессы изучены не в полной мере. В основном это

связано с тем, что внутренняя структура такого процесса и причинно-следственные связи, возникающие в процессе проведения тестирования неоднозначны. В связи с этим возникает необходимость структуризации и определения стратегии применения средств интеллектуального анализа с точки зрения разработчика и конечного пользователя. Первое определяет критерий надежности программных конструкций к внешним воздействиям, второе устанавливает, какие именно входные параметры могут послужить предметом возникновения ошибок. В рамках моделирования стохастических процессов тестирования используются различные логико-вероятностные модели. Основой построения таких моделей является определение структуры зависимостей переменных в виде некоторого графа, имеющего древовидную структуру и определяющего связи и направленность связей между узлами данных моделей. Каждая переменная такой модели имеет распределение вероятностей, описывающее условие появления определенных информативных событий. Использование такого представления позволяет моделировать состояния процессов тестирования в зависимости от наличия тех или иных внешних факторов, а также сформировать наиболее вероятные наборы тестовых данных. В рамках работы используется одна из таких вероятностных моделей – байесовская сеть (БС). Модели на основе БС реализуют процесс самообучения, используя наборы статистических данных. Это позволяет данным моделям сохранять устойчивость даже в условиях неполноты данных и наличии ошибок. Моделирование процессов тестирования на основе байесовских сетей обусловлено возможностью представления достаточно разнородных элементов тестирования в виде совокупности причинно-следственных связей.

Байесовская сеть является направленным графом, где каждая из вершин задается соответствующей таблицей условной вероятностей, а дуги определяют связи между данными вершинами. В классической интерпретации, если вершина x будет родительской по отношению к y, то существует направленная связь от x к y и $x \in Parents(y)$. Если переменная y не имеет непосредственной связи с x, то существует предположение об условной независимости, а вероятности являются безусловными (маргинальными). В качестве простейших байесовских сетей могут

выступать марковские цепи. Марковская цепь представление собой граф без циклов, который может иметь как направленные, так и ненаправленные дуги, узлы которого описывают модель допустимых состояний переходов, заданную на определенном временном интервале.

Гипотеза условной независимости утверждает, что узел X_i не будет зависеть от $Y \subseteq X, Y \cap Children(X_i) = \emptyset,$ при заданных значениях $Parent(X_i)$. Сформулирует критерий условной независимости для вершин X_i и Y_i для байесовской сети [61]

$$P(X_i, Y|Pr\ (X_i) = P\big(X_i\big|Parent(X_i)\big)P(X_i|\ Parent(Y)) \\ \forall Y \subseteq \frac{\{X_1, X_2, \ldots, X_n\}}{\{X_i\}}, Y \cap Children(X_i) = \emptyset. \tag{1.27}$$

Условная независимость для каждой вершины БС от других вершин тесно связана с понятием марковского покрытия. Марковское покрытие представляет собой некоторое множество, включающее в себя рассматриваемую вершину, множество родительских, дочерних вершин и всех родителей дочерних вершин. Обозначим марковское покрытие для у $\in X$ как $Y = X^y$. Для получения распределения марковского покрытия непосредственно из $P(X)$ необходимо просуммировать значения переменных $x \in X \backslash Y$:

$$P(Y) = \sum_{x_{r_1}} \sum_{x_{r_2}} \ldots \sum_{x_{r_k}} P(X), x_{r_k} \in X \backslash Y \tag{1.28}$$

Делая преобразование выражения (1.28) за счет разложения на множители и вынесения параметров, не зависящих от x_{r_k}, за знак суммы, можно привести данное выражение к следующему виду:

$$P(Y) = \sum_{x_{r_1}} \sum_{x_{r_2}} \ldots \sum_{x_{r_{k-1}}} \prod_{x_i \notin child^*\left(x_{r_k}\right)} P\big(x_i\big|Parent(x_i)\big) \\ \times \sum_{x_{r_k}} \prod_{x_i \in child^*\left(x_{r_k}\right)} P\big(x_i\big|Parent(x_i)\big), x_{r_k} \in X \backslash Y, \tag{1.29}$$

где $child^*\left(x_{r_k}\right) = child\left(x_{r_k}\right) \cup \left\{x_{r_k}\right\}$, $child\left(x_{r_k}\right)$ – множество дочерних вершин для узла x_{r_k}.

Вводим обозначение $T(X^{r_k}) = \sum_{x_{r_k}} \prod_{x_i \in child^*\left(x_{r_k}\right)} P\left(x_i \middle| Parent(x_i)\right)$ и выражение (1.29) перепишем в следующем виде [9]:

$$P(Y) = \sum_{x_{r_1}} \sum_{x_{r_2}} \dots \sum_{x_{r_{k-1}}} \prod_{x_i \notin child^*\left(x_{r_k}\right)} P\left(x_i \middle| Parent(x_i)\right) T(X^{r_k}), x_{r_k} \in X \backslash Y \qquad (1.30)$$

Из выражения (1.30) видно, что данную процедуру можно повторить для любого разложения $P(X)$ даже в том случае, если значения условных вероятностей не имеют смысла, что позволяет последовательно получить распределение для марковского покрытия.

Каждая переменная x БС определяется распределением условных вероятностей, устанавливающих стохастические связи между текущей вершиной и ее родительскими вершинами $Parents(x)$. Для вершин с дискретным распределением определяются таблицы условных вероятностей, формируемые с учетом домена значений рассматриваемой вершины и всех родительских вершин, связанных с ней. Определим полное совместное распределение вероятностей для всех переменных, входящих в состав байесовской сети в следующем виде [60]:

$$\begin{aligned} P(x_1, x_2, \dots, x_n) &= P(X_1 = x_1, X_2 = x_2, X_n = x_n) \\ &= \prod_{i=1}^{n} P\left(x_i \middle| Parents(X_i)\right) \end{aligned} \qquad (1.31)$$

Одним из характерных свойств, которыми обладает БС, является свойство d-разделенности. Логически данной свойство заключается в том, что при наличии вероятностной связи между двумя узлами байесовской сети справедливо одно из следующих утверждения: одна из переменных зависит от другой, обе переменные имеют общего родителя. Обобщенное определение d-разделенности можно сформулировать в следующем виде. Два узла байесовской сети a и b являются d-разделенными, если существует некоторый промежуточный узел с, такой что связь между данными узлами является: последовательной $a \to b \to c$, расходящейся $a \leftarrow \mathrm{c} \to b$, сходящейся $a \to \mathrm{c} \leftarrow b$. Свойство d-разделенности, применительно к некоторым множествам вершин X, Y относительно множества Z, сформулируем следующим образом: множества X, Y будут d-разделенными, в том случае если

дуга, соединяющая любой элемент из множества X и Y, разделена элементом из множества Z. Данное свойство позволяет определить узлы байесовской сети, не имеющие условной зависимости от родительских вершин, и характеризуются только безусловной вероятностью.

Для байесовских сетей при выполнении специального правила упорядочивания имеет место цепное правило [14,98]

$$P(x_1, \dots, x_n) = P(x_n|x_{n-1}, \dots, x_1)P(x_{n-1}|x_{n-2}, \dots, x_1) \dots P(x_2|x_1)P(x_1) = \prod_{i=1}^{n} P(x_i|x_{i-1}, \dots, x_1) \quad (1.32)$$

Для построения байесовской сети из n узлов используется алгоритм Д. Перла. В основе алгоритма применяется рекурсивный обход сети и добавление новых вершин из минимального множество родителей. Для каждой вершины выполняется гипотеза об условной независимости по отношению к вершинам, не вошедшим в состав родительских узлов ранее обработанных узлов.

Данный алгоритм реализует обход сети и добавление дуг из минимального множества родительских вершин, однако для текущего узла соблюдается правило условной независимости по отношении к узлам, не входящих во множество родительских вершин предшествующих узлов. Данный алгоритм имеет следующий вид [59]

Шаг 1. Выполняется формирование множество переменных $X = \{X_i\}_{i=1}^{n}$, характерных для данной предметно области.

Шаг 2. Задается порядок следования для $X_1, \dots, X_n$.

Шаг 3. Повторяем цикл до тех пор, пока $X \neq \emptyset$.

- Производим добавление X_i в БС.
- Добавляем ребра для узлов, уже содержащихся в БС с учетом выполнения свойства условной независимости

$P\left(X_i\middle|Parents(X_i), X_1^{'}, \dots, X_m^{'}\right) = P\left(X_i\middle|Parents(X_i)\right)$,

где $X_1^{'}, \dots, X_m^{'}$ – предшествующие для X_i переменные, $(X_1^{'}, \dots, X_m^{'} \notin Parents(X_i))$.

- Формируем таблицу условных вероятностей для текущей переменной X_i с учетом всех значений ее родительских вершин $Parents(X_i)$.
- Удаляем переменную из X_iиз множества X.

Алгоритм Перла является унифицированным механизмом построения байесовской сети, однако требует заранее определенной топологии сети. В зависимости от типов переменных, байесовские сети можно разделить на следующие основные категории – дискретные, непрерывные и гибридные. Последняя сеть сочетают характеристики дискретных и непрерывных сетей.

Алгебраические байесовские сети являются основой для построения интеллектуальных систем с использованием интервальных оценок. В общем случае алгебраическая байесовская сеть представляет собой граф, описываемый в виде набора фрагментов знаний. Под фрагментом знания понимается структура (C, p). Обозначение C соответствует идеалу цепочек конъюнктов над набором атомарных пропозиций (за исключением пустого конъюнкта). Интервальнозначная функция $p = \left[p(X); \mathrm{p}(X) \in [0,1]\right] \forall X \in C$ является вероятностным распределением истинности для указанного идеала цепочек конъюнктов.

Создавая для каждого идеала C_i структуру фрагмента знаний (C_i, P_i) на множестве атомарных пропозиций, можно представить алгебраическую байесовскую сеть в виде семейства фрагментов знаний $\{(\mathrm{C}_i, P_i)\}_{i=1}^{n}$, $\mathrm{C}^{\Delta} = \bigcup \mathrm{C}_i$ – носитель алгебраической БС. Оценки для распределения P_i могут быть как точечными, так и интервальными. Построение алгебраических байесовских сетей производится путем определения соответствующего графа G_N. В качестве узлов данного графа выступают фрагменты знаний, входящие в данную байесовскую сеть, и которым назначаются веса. Ребра соединяют узлы, соответствующие пересекающимся фрагментам знаний и для них также определяются веса. Весом для ребра является максимальный идеал, характеризующий пересечение некоторых фрагментов знаний.

Дискретные байесовские сети являются разновидностью БС, где все узлы представляют собой дискретные величины. В данном случае вершина представляет собой переменную, которая может принимать некоторое возможное множество значений D. Каждому значению $d \in D$ соответствует значение вероятности (условной или безусловной). При этом, если вершина имеет родителей, то ее значения описываются таблицей условных вероятностей, в противном случае

вершина имеет безусловную вероятность. Для вершин, имеющих k родителей, необходимо будет заполнить $O(2^k)$ значений в таблице условных вероятностей для оценки всех возможных вероятных исходов. Любая дискретная байесовскую сеть представляет собой совокупность следующих параметров: $K = \{X, G, P\}$, где $G = (T, H)$ – граф без циклов, содержащий $T = \{t_1, t_2, \dots, t_n\}$ вершин, связанных ребрами $H = \{h, h_2, \dots, h_n\}$, X – множество подмножеств $X = \{X_1, X_2, \dots, X_n\}$ дискретных значений X_i, принимаемых вершинами графа G, P – совокупность вероятностных распределений (условных, безусловных) для каждой из вершин графа.

Непрерывные байесовские сети характеризуются вершинами с непрерывным вероятностным законом распределения. Основная область применения непрерывных байесовских сетей связана с моделированием стохастических процессов. Пространство значений для переменных X может быть представлено в виде некоторого допустимого интервала значений $(x|a \le x \le b)$. Для переменной X с множеством родительских вершин $Pa = (Pa_1, Pa_2, \dots, Pa_n)$, достаточно часто используется нормальное распределение $f(X|Pa_i) = N(x; \mu_x + b_i\mu_i, {\sigma_x}^2)$, которое определяется следующим выражением:

$$N(x; \mu_x + b_i\mu_i, {\sigma_x}^2) = \frac{1}{\sigma_x\sqrt{2\pi}} exp\left(\frac{\left(x - (\mu_i + b_i * \mu_i)\right)^2}{{\sigma_x}^2}\right), \quad (1.33)$$

где μ – математическое ожидание, σ^2 – дисперсия, b_i – коэффициент, устанавливающий связь между переменной X и ее родителем Pa_i.

Зависимость между переменной X и ее родителями можно представить в виде регрессионной модели:

$$X = b_1 * U_1 + b_2 * U_2 + \cdots + b_n * U_n + Q_x, \quad (1.34)$$

где Q_x – представляет собой шумовой коэффициент, который может быть записан в виде распределения Гаусса с математическим ожиданием равным 0.

Гибридные байесовские сети представляют собой суперпозицию дискретных и непрерывных байесовских сетей и могут содержать как дискретные, так и непрерывные случайные величины. Однако при этом существует ряд ограничений:

непрерывные переменные не могут содержать дискретных потомков и должны иметь нормальный закон распределения случайных величин. Определим закон распределения для переменной X с дискретными и непрерывными родительскими вершинами X и Y соответственно

$$P(X|Y=y,Z=z)=N\left(\mu_x(\mu_y,\mu_z),\sqrt{\sigma_x\left(\sqrt{\sigma_y}\right)}\right), \tag{1.35}$$

где $\mu_x,\mu_y\mu_z$ – математические ожидания, σ_x,σ_y – дисперсии, $\sqrt{\sigma_y},\sqrt{\sigma_x}$ – среднеквадратические отклонения.

Пусть имеем непрерывную переменную Y с дискретными $X_1\left((x_{11},p_{11}),\dots,\left(x_{1n_1},p_{1n_1}\right)\right),\dots,X_s\left((x_{s1},p_{s1}),\dots,\left(x_{sn_s},p_{sn_s}\right)\right)$ и непрерывными $Z_1\left(N(\mu_1,\sigma_1)\right),\dots,Z_m\left(N(\mu_m,\sigma_m)\right)$ родителями. Тогда вероятностные характеристики величины Y можно представить в следующем виде:

$$\mu=\sum_{i_1=1}^{n_1}\dots\sum_{i_1=1}^{n_1}p_{1i_1}\dots p_{si_{1s}}\left(\mu_{i_1\dots.i_s}+\sum_{l=1}^{r}K_{l,i_1\dots.i_s}\mu_l\right) \tag{1.36}$$

$$\sigma=\sum_{i_1=1}^{n_1}\dots\sum_{i_1=1}^{n_1}p_{1i_1}\dots p_{si_{1s}}\left(\left(\mu_{i_1\dots.i_s}+\sum_{l=1}^{r}K_{l,i_1\dots.i_s}\mu_l\right)^2+\sigma_{i_1\dots i_s,}+\sum_{l=1}^{r}K^2{}_{l,i_1\dots.i_s}\sigma_l\right)-\mu^2 \tag{1.37}$$

$\mu_{i_1\dots.i_s}$, $\sigma_{i_1\dots i_s,}$ – математическое ожидание и дисперсия совместного влияния дискретных случайных величин, $K_{l,i_1\dots.i_s}\mu_l$ – весовой коэффициент совместного влияния непрерывных случайных величин.

Нечеткие байесовские сети являются следствием объединения математического аппарата дискретных (непрерывных) байесовских сетей и теории нечетких множеств. Для определения данной байесовской сети используется процесс введения нечеткости. Функция принадлежности $\mu_{X=x_i}(Y=y_j)$ для родительской переменной X на множестве значений дочерней переменной Y применяется к байесовскому правилу, которое приобретает следующий вид:

$$P\left(X = x_i \middle| Y = y_j\right) = \frac{\mu_{X=x_i}(Y = y_j)P(X = x_i)P\left(Y = y_j \middle| X = x_i\right)}{\sum_{i=1}^{n} P(X = x_i)P\left(Y = y_j \middle| X = x_i\right)},$$
$$\sum_{i=1}^{n} \mu_{X=x_i}(Y = y_j) = 1 \quad (1.38)$$

Для введения нечеткости, вероятности состояний вершин байесовской сети заменяются нечеткими числами, а арифметические операция заменяются расширенными операциями над нечеткими множествами. Задается функция принадлежности $\mu_{\tilde{X}}(x) \in [0,1], x \in L$ и нечеткое число $\tilde{X}$ представляется в виде нечеткого множества. Должно соблюдаться следующее условие: $\exists x \in L$, такое, что $\mu_{\tilde{X}}(x) = 1$ или $\exists N \in L$, такое, что $\forall x \in L$, если $|x| \leq N$, то $\mu_{\tilde{X}}(x) = 1$. Основой построения нечеткой байесовской сети является применение цепного правила для нечеткого совместного распределения условных вероятностей, имеющее следующее представление:

$$\tilde{P}_f(X_1, X_2, \dots, X_n) \cong_{i=2}^{n} \widetilde{\otimes}\ \tilde{P}_f\left(X_i \middle| Parents(X_i)\right) \quad (1.39)$$

Определим формула Байеса для нечеткой байесовской сети

$$\tilde{P}_f\left(X = x_i \middle| Y = y_j\right) \cong \frac{\tilde{P}_f(X = x_i)\ \widetilde{\otimes}\ \tilde{P}_f\left(Y = y_j \middle| X = x_i\right)}{\tilde{P}_f\left(Y = y_j\right)} \quad (1.40)$$

Динамические байесовские сети (ДБС) могут рассматриваться как набор байесовских сетей, взятых для каждого временного среза $t = t_0, t_1, \dots, t_n$. Временной срез представляет собой статическую байесовскую сеть, взятую в конкретный момент времени t_i с соответствующим распределением вероятностей. Структурно все переменные динамической байесовской сети можно разделить на два множества: наблюдаемых переменных X_t и переменных свидетельств E_t. В процессе создания ДБС, необходимо определить следующие вероятностные параметры. *Начальное распределение* $P(X_0)$, описывающее распределение условных вероятностей для начального момента времени t_0, вычисляемое на стадии обучения ДБС. *Модель перехода* $P(X_t|X_{t-1})$ *у*станавливает транзитивные условные вероятности между двумя временными срезами t и $t + 1$. *Модель восприятия* $P(E_t|X_t)$ – определяет распределение вероятностей для всех

переменных свидетельств для момента времени $t+1$. Для задания модели перехода и восприятия требуется определить связи между временными срезами, а также установить переменные состояния и свидетельства X_t и E_t. Переменные состояния X_t начинают поступать с нулевого момента времени $t=0$, в то время как свидетельства E_t с момента времени $t=1$. Из семантики ДБС вытекает следующее следствие – любые переменные запроса и свидетельства могут иметь неограниченное число связей с переменными из смежных временных срезов. В процессе построения транзитивных связей предполагается, что переменные зависят лишь от конечного числа предшествующих состояний. Эти требование достигается за счет использования предположения о марковости применительно к динамической байесовской сети. Формально модель перехода и восприятия может быть представлена в виде Марковской цепи.

Марковская цепь представляет собой направленный (ненаправленный) граф без ориентированных циклов. Модель возможных состояний Марковской цепи, описывающая процесс перехода из одного состояния в другое, представлена в виде совокупности узлов графа. В рамках динамической байесовской сети, марковская цепь описывает некоторую последовательность случайных дискретных величин, взятых на заданном временном интервале. Понятие времени в данном случае также носит дискретный характер. Модели перехода и восприятия могут быть представлены в виде марковского процесса первого рода

$$P(X_t|X_{0:t-1}) = P(X_t|X_{t-1}) \tag{1.41}$$

$$P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t) \tag{1.42}$$

Для того, чтобы сформировать модели (1.41) и (1.42) необходимо определить топологию связей между рассматриваемыми временными срезами ДБС, а также для всех существующих переменных запроса и свидетельств. В силу стационарности рассматриваемых моделей (являются одинаковыми для всех временных срезов t), данную топологию можно определить только для первого среза модели ДБС. На основе введенных определений и распределений вероятностей для моделей (1.41) и (1.42), можно представить полное совместное

распределение вероятностей для всех переменных ДБС [43] с учетом всех определенных связей между слоями

$$P(X_0, X_1, \dots, X_t, E_1, \dots, E_t) = P(X_0) \prod_{i=1}^{t} P(X_i|X_{i-1}) P(E_i|X_i) \quad (1.43)$$

Для определения механизмов тестирования стохастических процессов, представленных в виде ДБС, установления интерфейсов взаимодействия между подсистемами тестирования и реализации байесовского вывода необходимо сформировать механизмы управления. Механизмы управления предназначены для координации запуска, анализа и корректирования тестов, осуществления взаимодействия с модулем вероятностного вывода. Структурируем процесс управления в виде следующих этапов: генерация, выполнение, анализ, коррекция, прогнозирование. На этапе генерации происходит определения входных точек приложения, для которых необходимо выполнить тестирования, после чего происходит порождение наборов тестовых данных. На этапе выполнения происходит пошаговый запуск тестовых сценариев (цепочек сценариев) для обнаружения ошибок веб-приложения. На этапе анализа происходит выявление наиболее эффективных наборов тестовых данных, вычисление начального распределения $P(X_0)$, моделей перехода $P(X_t|X_{t-1})$ и восприятия $P(E_t|X_t)$. На завершающей фазе происходит построение прогнозов вероятности возникновения ненаблюдаемых событий для момента времени $t+1$ на основе реализации процедуры вероятностного вывода.

Концептуальная модель процесса тестирования представляет собой обобщенное логически завершенное представление процесса тестирования ошибок устойчивости с возможностью введения временных стохастических связей в рамках применения математических моделей, построенных на основе динамических байесовских сетей. Для формального описания процесса тестирования используется концептуальная модель тестирования [125], представленная на рисунке 1.5

Рисунок 1.5. Концептуальная модель тестирования веб-приложений на основе ДБС

В силу того, что процесс тестирования имеет вероятностную природу, для моделирования данного процесса целесообразно использовать модели, способные объединять отдельные функциональные компоненты в виде комплексного случайного процесса. Применение концептуальной модели, представленной на рисунке 1.5 дает возможность сформировать абстрактное представление рассматриваемой предметной области, использующее набор взаимосвязанных понятий, применяемых для описания области тестирования, наборы подходов, характеристик, классификаций по ситуативным признакам и законам протекающих процессов. На основе понятийного аппарата концептуальных моделей формируются связи между функциональными элементами тестирования, математическими алгоритмами и компонентами имитационного моделирования, применяемыми в процессе анализа ошибок веб-приложений. Сущность данной модели заключается в наглядном представлении и декомпозиции процесса тестирования на ряд элементарных действий. Одни из которых реализуют стратегию функционального тестирования, другие использует математический аппарат, построенный на основе применения ДБС. Использование имитационных моделей позволяет выявить возможные поведенческие сценарии работы приложений в случае присутствия одной из OWASP-групп ошибок, а также способствует определению форматов и механизмов взаимодействия между элементами, предназначенными для фаззинг-тестирования.

1.4. Постановка задач исследования

Анализ современных исследований в области тестирования веб-приложений, проведенный в главе 1, позволяет сделать вывод, что фаззинг-тестирование является апробированным, широко используемым и перспективным направлением в сфере развития методов тестирования, предоставляющим возможности комплексного подхода к анализу и выявлению программных ошибок. Для повышения эффективности фаззинг-тестирования необходимы специальные формализованные инструменты, которые позволили бы для стохастического процесса тестирования

моделировать вероятностные связи между фрагментами тестирования и осуществлять предсказание результатов тестирования. ДБС являются графическими вероятностными моделями, они хорошо отражают сущность стохастического процесса тестирования и его логические связи. Теория байесовских сетей интенсивно развивается, создаются новые более мощные алгоритмы обучения и вероятностного вывода, которые решают основные задачи моделирования для сложных ДБС с большим количеством вершин и вероятностных связей. Несмотря на большое количество исследований в данной сфере, актуальными являются разработки, связанные с развитием формализованных инструментов построения структуры сетей, обучения их параметров и реализации прогнозирования, фильтрации и сглаживания.

Целью исследования является разработка новых теоретических и практических подходов, направленных на повышение эффективности организации информационных процессов фаззинг-тестирования и анализа веб-приложений на основе применения моделей ДБС, способных накапливать информацию о результатах тестирования и формировать прогнозы на основе применения методов вероятностного вывода, реализовать развитие математического аппарата и инструментария исследования, разработать основные направления для временной и ресурсной оптимизации алгоритмов обучения и вероятностного вывода ДБС тестирования веб-приложений.

В рамках реализации цели исследования поставлены следующие теоретические и практические задачи диссертационного исследования:

1. с позиции системной методологии провести анализ существующих подходов к моделированию процессов тестирования ошибок функционирования веб-приложений с целью выявления направлений совершенствования методов обработки информации, генерируемой в процессе тестирования, и прогнозирования результатов тестирования, предложить и обосновать подход моделирования процессов тестирования с помощью динамических байесовских сетей;

2. построить модели динамических байесовских сетей для процессов тестирования основных классов ошибок функционирования веб-приложений методом фаззинга, отражающие динамику изменения состояния надежности различных аспектов функционирования приложений во времени;

3. разработать комплекс методов, реализующих гибридные технологии обучения структуры и параметров динамических байесовских сетей, включающий обучение топологии, направленности связей между отдельными узлами сети и вероятностных параметров вершин сети, базирующихся на методах статистического оценивания и структурной оптимизации, а также адаптированных для решения задач обучения в условиях неполных данных;

4. разработать метод повышения эффективности процедуры вероятностного вывода на основе многочастичного фильтра для динамических байесовских сетей, базирующийся на применении метода Метрополиса-Гастингса на этапе повторной генерации выборок, деревьев сочленения и теории достаточных статистик;

5. разработать метод синхронизации большого объема данных, передаваемых в распределенной вычислительной системе в процессе решения задач обучения и вероятностного вывода для динамических байесовских сетей тестирования веб-приложений методом фаззинга;

6. разработать программное обеспечение, предназначенное для реализации предложенных в исследовании методов и алгоритмов организации процессов фаззинг-тестирования веб-приложений на основе моделей динамических байесовских сетей и провести на базе данного программного обеспечения комплексный вычислительный эксперимент по оценке эффективности применения предложенных в исследовании инструментов для организации процесса тестирования основных групп ошибок функционирования веб-приложений по классификации OWASP и MITRE.

Во второй и третьей главах диссертации описаны все теоретические аспекты полученных в работе моделей, методов и алгоритмов решения задач обучения и вероятностного вывода для ДБС организации процесса фаззинг-тестирования веб-приложений. Совокупность предлагаемых методов и алгоритмов позволяет реализовать современную стратегию интеллектуального тестирования, способную адаптироваться к решению сложных задач тестирования различных классов ошибок веб-приложений, имеющих разную архитектуру построения. Предложенный инструментарий позволяет реализовать основополагающие принципы обнаружения аномального поведения приложений и создать условия для

своевременного выявления нетипичных ошибок за счет осуществления процессов прогнозирования. В четвертой главе рассмотрены основные результаты исследования, связанные с оптимизацией процессов синхронизации в распределенных системах, осуществляющих решение задач обучения структуры, параметров и вероятностного вывода ДБС для организации процесса фаззинг-тестирования веб-приложений. Основные результаты, описанные в данной главе, связаны с подбором параметров моделей массового обслуживания за счет решения оптимизационных задач, базирующихся на принципе максимального правдоподобия. В пятой главе исследования описан реализованный в рамках исследования вычислительный эксперимент по оценке эффективности применения предложенных в диссертации инструментов для тестирования основных групп ошибок функционирования веб-приложений по классификации OWASP и MITRE и сформированы основные результаты вычислительного эксперимента. В рамках данной главы: приведены параллельные алгоритмы, реализующие методы решения задач обучения и вероятностного вывода, проведена оценка достоверности построенных моделей на основе метрики Хэмминга и структурной энтропии, рассмотрена концепция применения облачных и распределенных вычислений для реализации основных алгоритмов, проведено построение вычислительной среды тестирования, представляющей совокупность тестируемых веб-приложений и взаимодействующих компонентов модулей (веб-сервер, система управления базами данных, сервер приложений и т.д.).

Глава 2. Разработка методов обучения структуры и параметров динамических байесовских сетей

2.1. Разработка методов обучения структуры динамических байесовских сетей

Обучение структуры связей является важным компонентом моделирования процессов тестирования в виде динамических байесовских сетей. В процессе обучения устанавливаются топологические связи между компонентами тестирования, формируются уровни критичности для каждой из ошибок. В связи с тем, что динамическая байесовская сеть представляет собой совокупность байесовских сетей, рассматриваемых в рамках заданного временного интервала, следует, что внутренняя структура сетей не меняется с течением времени. В процессе обучения структуры динамических байесовских сетей применяются тоже алгоритмы, что и для статических байесовских сетей. Между тем данные алгоритмы не учитывают факт наличия транзитивных связей между различными временными состояниями. В рамках данной диссертации рассматривается проблема обобщения алгоритмов обучения структуры динамической байесовской сети с учетом модели перехода.

Обучение байесовской сети производится на основе структурированного набора обучающих данных, характеризующих состояние узлов байесовской сети. Оценивая общее число возможных нециклических структур-кандидатов допустимых при использовании n переменных, воспользуемся формулой Робинсона [73]:

$$f(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i). \qquad (2.1)$$

Из формулы (2.1) следует, что для $n = 2$ общее число возможных структур БС будет равно 3, для $n = 5$ и $n = 10$ будет 29000 и 4.2×10^{18}. Таким образом применение методов прямого перебора структуры является ограниченным, доступно лишь обучение БС с минимальным набором параметров. Для решения

данной проблемы наиболее часто используются методы на основе статистического анализа обучающей выборки. В качестве основы для обучения структуры байесовских сетей выделяются методы ограничения условной независимости (сеть может быть частично обученной) и оценки качества сети (повышение вычислительных затрат с увеличением числа узлов байесовской сети). В основе алгоритмической базы метода на основе ограничения условной независимости выбора оптимальной структуры байесовской сети лежат механизмы проведения статистических тестов, определяющих условную независимость между анализируемыми элементами сети. Применение данных методов не позволяет в полной мере определить направленность и топологию связей между узлами БС. Данные методы применяются для того, чтобы сузить диапазон поиска и получить базовую структуру ориентированного графа байесовской сети. В основе данных методов используется определение вероятностных взаимосвязей между узлами байесовской сети на основе вычисления Марковского покрытия сети и применения понятия d-разделенности, заключающегося в оценке условной независимости вершин X и Y при заданных значениях узла Z. Наиболее распространенными алгоритмами на основе метода поиска с ограничениями являются [81].

1. *Растяжения-сжатия (grow-shrink).* Данный алгоритм основывается на вычислении марковской границы, представляющей собой минимально допустимое значение марковского покрытия. Марковское покрытие для узла X достаточно просто может быть выделено из графа байесовской сети и включает в себя множество родительских, дочерних вершин и родителей для каждой из дочерних вершин, ассоциируемых с переменной X. Исходя из своего названия, алгоритм структурно разделен на два этапа: этап растяжения и этап сжатия. На первом этапе формирования марковского покрытия для переменной X происходит заполнение специального множества S, состоящего из переменных, имеющих связь с переменной X. В результате выполнения первого этапа, в множество S могут попасть переменные, не входящие в марковское покрытие X. Для решения данной проблемы на втором этапе происходит удаление данных переменных.

2. *Быстрых инкрементных ассоциаций (fast incremental association).*

Отличительной особенностью алгоритма является возможность переупорядочить множество переменных при добавлении каждой новой переменной в состав марковского покрытия для переменной X. Процесс переупорядочивания переменных происходит на основе эвристической функции h, характеризующей взаимную условную информацию. На каждом шаге происходит сортировка переменных, являющихся кандидатами на добавление в состав марковского покрытия $MB(X)$. Порядок отражает направление снижения условной зависимости. В тоже время алгоритм использует статистический критерий G^2 для определения связей между переменной X и элементами из анализируемого множества переменных S. Для оптимизации алгоритма выбираются значения G^2 с наибольшими значениями. Повторная сортировка на каждой итерации не производится.

3. *Инкрементных ассоциаций с чередованием (interleaved incremental associations)*. Данный алгоритм является разновидностью алгоритма инкрементных ассоциаций, использует операцию чередования фаз «вперед и назад» и позволяет получать наименьшие области Марковского покрытия $MB(X)$. Его использование направлено на снижение общего числа ложных срабатываний алгоритма в процессе вычисления $MB(X)$.

4. *Минимаксный родитель-потомок (max-min parent and children)*. Алгоритм растяжения-сжатия использует в качестве критерия обучения марковское покрытие. В основе алгоритма применяется классическая структура, состоящая из двух фаз. На начальной фазе происходит определение переменных-кандидатов, которые заносятся в множество кандидатов (CPC) на основе вычисления минимальной ассоциации для текущей переменной. На завершающей фазе происходит проверка выполнения критерия d-разделенности для каждого подмножества множества CPC и переменной X.

В качестве критериев условной независимости для алгоритмов поиска с ограничениями в дискретных байесовских сетях используется ряд статистических оценок: взаимная информация, критерий Пирсона χ^2, информационные критерии Шварца и Акаике.

Критерий Пирсона является универсальной статистикой, характеризующей степень принадлежности некоторой выборки значений переменной заданной (ожидаемой) функции распределения. При этом предполагается, что гипотеза о принадлежности принимается только в том случае, если значение статистики критерия Пирсона попадает в определенную область. Критерий Пирсона используется для проверки следующей гипотезы:

$$P(X,Y|Z) = P(X|Z)P(Y|Z) \tag{2.2}$$

Статистика критерия Пирсона имеет вид [38]

$$\chi^2(X,Y|Z) = \sum_{a,b,c} \frac{\left(N_{a,b,c} - E_{a,b,c}\right)^2}{E_{a,b,c}} \tag{2.3}$$

где $E_{a,b,c}$ – ожидаемая частота события $x = a, y = b, z = c$

$$E_{a,b,c} = \frac{N_{ac}N_{bc}}{{N_c}^2}, \tag{2.4}$$

где $N_{a,b,c}$ – наблюдаемая частота появления данных, где $x = a, y = b, z = c$.

Критерий Акаике (АИК) применяется в процедуре оценивания моделей с переменным числом параметров. Понятие критерия АИК тесно связано с расстоянием Кульбака-Лейблера $I(f,g) = \int f(x)\ln\frac{f(x)}{g(x|\theta)}dx$, позволяющего оценить расстояние между рассматриваемыми моделями. Отличительной особенностью данного критерия является наличие функции штрафов, имеющей линейную зависимость от числа параметров.

Алгоритмы на основе метода оценки степени качества представляют собой целое семейство алгоритмов в первую очередь предназначенных для оптимизации процедур поиска. Основная сущность данных алгоритмов заключается в вычислении оценки для переменных-кандидатов байесовской сети, а также максимизация данной оценки. Наибольшее распространение получили алгоритмы поиска с восхождением (hill climbing), имитации отжига (simulated annealing), а также поиск с запретами (tabu search).

1. *Поиск с восхождением* представляет собой разновидность алгоритма локального поиска. В классической интерпретации представляет собой цикл, на

каждой итерации которого происходит смещение в сторону максимума. При достижении максимального значения, если отсутствуют соседние состояния с большими значениями, алгоритм останавливается. В процессе выполнения алгоритма нет необходимости накапливать данные в процессе обхода дерева поиска, достаточно хранить только текущее состояние и соответствующее ему значение функции. Данный алгоритм обладает достаточно высокой производительностью, так как процесс сдвига в область максимума, как правило, не занимает много времени. Между тем основным недостатком данного алгоритма является появление локальных оптимумов, что особенно заметно, если состояние имеет резко выраженные по уровню значения.

2. *Имитация отжига* является рандомизированным алгоритмом локального поиска, направленным на исключение локальных оптимумов в процессе поиска. Природа алгоритма имеет сходство с физическим процессом отжига твердых веществ, направленным на получение физическими телами, обладающих низкой энергией, состояния за счет максимального нагрева и последующего постепенного снижением температуры. Алгоритм имитации отжига имеет сходство с методом градиентного спуска, однако каждое решение формируется случайным образом из некоторой окрестности в процессе достижения требуемого значения, взамен поиска позиции, обладающей минимаксным значением. Если решение действительно позволяет улучшить значение целевой функции, то такое состояние принимается, в противном случае оно будет иметь вероятностное значение меньшее единицы. В тоже время, уменьшение значения вероятности имеет экспоненциальную зависимость от изменения состоянии на величину оценки ΔE. В общем случае использование вероятностной процедуры выбора решения позволяет сократить число возможных локальных оптимумов и тем самым обеспечить эффективность поиска оптимального решения.

3. *Поиск с запретами.* Основная стратегия данного алгоритма базируется на использовании так называемого списка запретов, позволяющего исключить возможность повторного посещения решений R. Список запретов представляет собой множество всех просмотренных ранее решений T, на которые

накладываются ограничения (запреты), и они не могут быть выбраны в процессе выбора окрестности значения. Процедура принятия решения на каждой следующей итерации алгоритма зависит не только от качества и полноты, но также от ретроспективны данных. Лучшее решение R_{opt}, не входящее во множество запретов $R_{opt} \notin T$ из некоторой окрестности, выбирается для следующей итерации.

Использование алгоритмов на основе методов оценки степени качества поиска с ограничениями по отдельности не могут быть адаптированы в качестве базовых алгоритмов обучения байесовских сетей, так как первые позволяют лишь определить направленность связей между отдельными узлами сети, но не могут построить оптимальную структуру сети, вторые же наоборот, не могут определить направленность связей внутри сети. Для решения данных проблем в научном сообществе была выделена группа алгоритмов, позволяющая сочетать преимущество обоих групп рассмотренных алгоритмов и обеспечить гибкую адаптацию к решению задачи обучения структуры байесовской сети. Такие алгоритмы получили названия гибридные. Основное преимущество данных алгоритмов – это декомпозиция процесса обучения на два основных блока, реализованных на основе определения марковского покрытия и условных связей между вершинами байесовской сети.

Наибольшее распространение получили алгоритмы минимаксного восхождения и ограниченная двухфазная максимизация. Рассмотрим их содержание.

1. *Минимаксное восхождение*. В классической интерпретации алгоритм может быть разделен на два основных этапа, характеризующих применение минимаксного алгоритма «родителя-потомка» и поиска с восхождением. Рассмотрим более детально каждый из этапов в отдельности. Первый этап алгоритма характеризуется формированием топологии байесовской сети на основе применения алгоритма минимаксный родитель-потомок (ММРП). Для обозначения байесовской сети используется следующее выражение $B_n = \langle G, P \rangle$, где G – граф, определяющий структуру байесовской сети, P – распределения вероятностей для случайных переменных X. В качестве входных данных для

алгоритма ММРП используется множество обучающих выборок $D = \{D_1, D_2, \dots, D_n\}$. Для построения структуры сети используется критерий условной независимости $I(X;Y|Z)$, характеризующий условие независимости двух переменных X и Y при наличии Z. Для вычисления условной независимости $I(X;Y|Z)$ используется оценка статистического критерия на множестве данных D. Для определения степени ассоциации между переменной X и Y при наличии переменной Z используется функция $A(X;Y|Z)$, при этом для данной функции выполняется следующее выражение:

$$I(X;Y|Z) \Leftrightarrow (A(X;Y|Z) = 0) \quad (2.5)$$

Структурно ММРП [85] может быть разделен на два основных шага. Первый шаг характеризуется формированием множества кандидатов родителей и потомков CPC_X для каждой переменной X. Заполнение множества CPC_X происходит циклически за счет вычисления эвристической функции, позволяющей выбирать переменные, максимизирующие минимальную связь с переменной X при условии наличия переменных-кандидатов, входящих в состав множества CPC_X. На основе выполнения эвристической функции можно сделать заключение, что условием включения переменной X_i в состав множества CPC_X, является наличие ребер, входящих или исходящих из X. Переменная X в одном случае является родительской вершиной для X_i, в другом дочерней. Эвристика алгоритма заключается в выборе переменной X_i, имеющей устойчивую связь с переменной X, при условии наличия различных элементов множества CPC_X. Переменные, значения связей которых по отношению к переменной X равны нулю, не включаются во множество CPC_X и исключаются из рассмотрения. Критерием завершения первого этапа выступает отсутствие связи всех необработанных переменных с X, значение минимаксной связи становится равным нулю. Для установления факта условной независимости переменной X_i от X при наличии всех переменных X_k, входящих в состав CPC_X, используются G^2 критерий

$$G^2 = 2\sum_{a,b,c} N_{X,i,k}^{a,b,c}\, ln\frac{N_{X,i,k}^{a,b,c}}{E_{X,i,k}^{a,b,c}} = 2\sum_{a,b,c} N_{X,i,k}^{a,b,c}\, ln\frac{N_{X,i,k}^{a,b,c} N_k^c}{N_{X,k}^{a,c} N_{i,k}^{b,c}}, \quad (2.6)$$

$$E_{X,i,k}^{a,b,c} = \frac{N_{X,k}^{a,c} N_{i,k}^{b,c}}{N_k^c},$$

где $\mathrm{E}_{X,i,k}^{a,b,c}$ и $N_{X,i,k}^{a,b,c}$ – количество всех ожидаемых и наблюдаемых в обучающих данных D комбинаций, где $X = a, X_i = b, X_k = c$, а $N_{X,k}^{a,c}$ – аналогичный показатель для двух переменных.

G^2 имеет общее число степеней свободы

$$df = (|D_m(X)| - 1)\,(|D_m(X_i)| - 1) \prod_{X_k} |D_m(X_k)|, \tag{2.7}$$

где $D_m(X), D_m(X_i), D_m(X_k)$ – мощности доменов значений для рассматриваемых переменных X, X_i, X_k (домен – множество неповторяющихся значений, которое может принимать переменная).

В качестве порогового уровня значимости для критерия G^2 используется значение 0.05. В процессе вычисления тестов на условную независимость на основе G^2, определяется число вхождений всех возможных комбинаций $N_{X,i,k}^{a,b,c}, \forall a, b, c$ для переменных X, X_i, X_k, при этом общее число обучающих данных, необходимых для вычисления оценки таких комбинаций имеет экспоненциальную зависимость от числа условных связей. Общее число условных связей не ограничивается количеством родительских и дочерних вершин, а зависит от числа элементов, входящих в состав множества узлов-кандидатов CPC_X. На втором этапе происходит корректировка множества CPC_X и удаление ошибочно добавленных переменных-кандидатов. В качестве критерия удаления повторно происходит выполнение тестов на условную независимость $I(X; Y|Z)$. Если данное условие выполняется, то производится удаление переменной из множества CPC_X. После завершения работы алгоритма ММРП получаем сформированную ненаправленную байесовскую сеть.

На следующем этапе происходит установление направленности ребер байесовской сети за счет применения алгоритма поиска с восхождением. Процедура локального поиска на основе алгоритма восхождения начинается с пустого графа. Поиск является рекурсивной операцией, позволяющей оценить

изменение результирующей оценки графа в зависимости от добавления, удаления ребер, а также изменения их направленности.

Рассмотрим эквивалентную метрику Байеса-Дирихле (БДЭ). Данная метрика позволяет определить степень эквивалентности графов на основе совпадения утверждений условной независимости для каждого из узлов графа.

Для каждой переменной X_i рассмотрим $\{1,2,\dots,q_i, q_i = \prod_{S\in \mathrm{Pr}(X_i)} r_s\ , r_s = D_m(S)\}$ множество комбинаций значений родителей. Для всего графа G введем множество параметров $\{\Theta^G_{i,j,k}, i \in \{1,\dots,n\}, j \in \{1,\dots,q_i\}, k \in \{1,\dots,r_i\}, \Theta^G_{i,j,k} > 0; \sum_{k=1}^{r_i} Q^G_{i,j,k} = 1\}$.

Предполагается, что каждая переменная с заведомо известным фиксированным значением родительских вершин имеет распределение Дирихле

$$p\left(\Theta^G_{i,j}|G\right) = \\ = \frac{\Gamma\left(\sum_{s=1}^{r_i} \alpha_{i,j,s}\right)}{\prod_{s=1}^{r_i} \Gamma\left(\alpha_{i,j,s}\right)} 1_{\left\{\Theta^G_{i,j,1},\dots,\Theta^G_{i,j,r_i-1}>0; \sum_{k=1}^{r_i-1} Q^G_{i,j,k}<1\right\}} \prod_{k=1}^{r_i} \left(\Theta^G_{i,j,k}\right)^{\alpha_{i,j,k}-1}, \quad (2.8)$$

где $\alpha_{i,j,k} > 0$ – параметры распределения Дирихле, $\Theta^G_{i,j} = \left(\Theta^G_{i,j,1},\dots,\Theta^G_{i,j,r_i-1}\right)$ – вектор параметров, соответствующий одной переменной и единственному значению ее родителя, $\Theta^G_{i,j,r_i} = 1 - \sum_{k=1}^{r_i-1} \Theta^G_{i,j,k} > 0$.

Для формирования метрики используются два понятия независимости параметров сети: локальная и глобальная независимость. Локальная независимость характерна для параметров в том случае, если родительские вершины для каждой переменной X_i независимы друг от друга

$$p\left(\Theta^G_i|G\right) = \prod_{j=1}^{q_i} p\left(\Theta^G_{i,j}|G\right), \quad (2.9)$$

где $\Theta^G_i = \bigcup_{j=1}^{q_i} \Theta^G_{i,j}$ – вектор параметров для переменной X_i байесовской сети при всех возможных значениях родителей.

Глобальная независимость характеризует отсутствие связей между векторами параметров переменных байесовской сети

$$p(\Theta^G|G) = \prod_{i=1}^{n} p\left(\Theta_i^G|G\right), \tag{2.10}$$

где $\Theta^G = \bigcup_{i=1}^{n} \Theta_i^G$.

Основываясь на свойствах локальной и глобальной независимостей байесовской сети, распределение Дирихле принимает следующий вид [56]:

$$p(\Theta^G|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{i,j,k}\right)}{\prod_{k=1}^{r_i} \Gamma\left(\alpha_{i,j,k}\right)} \prod_{k=1}^{r_i} \left(\Theta_{i,j,k}^G\right)^{\alpha_{i,j,k}-1} \tag{2.11}$$

Используя определение плотности распределения вероятности на основе распределения Дирихле (2.8), апостериорное распределение для байесовской сети принимает следующий вид:

$$P(D|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{i,j,s}\right)}{\Gamma\left(\sum_{s=1}^{r_i} N_{i,j,s} + \alpha_{i,j,s}\right)} \prod_{s=1}^{r_i} \frac{\Gamma\left(N_{i,j,s} + \alpha_{i,j,s}\right)}{\Gamma\left(\alpha_{i,j,s}\right)}, \tag{2.12}$$

где $D = \left\{D_1 = \{X_1^1, X_2^1, \dots, X_n^1\}, D_2 = \{X_1^2, X_2^2, \dots, X_n^2\}, \dots, D_m = \{X_1^m, X_2^m, \dots, X_n^m\}\right\}$ – выборка, состоящая из m независимых одинаково распределенных случайных величин для параметра Θ^G.

Из формулы Байеса $P(G|D) = \frac{P(D|G)P(G)}{P(D)}$ видно, что только числитель имеет непосредственную зависимость от структуры сети, тогда $P(G|D) \propto P(D|G)P(G)$, а априорная вероятность $P(G)$ будет идентична для любой байесовской сети

$$P(G|D) \propto P(D|G)P(G) = P(G) \int P(D|G, \Theta^G)\, P(\Theta^G|G) d\Theta \tag{2.13}$$

Метрика Байеса-Дирихле (БД) представляет собой выражение

$$ln\, P(D|G) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} ln \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{i,j,s}\right)}{\Gamma\left(\sum_{s=1}^{r_i} N_{i,j,s} + \alpha_{i,j,s}\right)} + \sum_{s=1}^{r_i} \frac{\Gamma\left(N_{i,j,s} + \alpha_{i,j,s}\right)}{\Gamma\left(\alpha_{i,j,s}\right)} \tag{2.14}$$

Для задания эквивалентной метрика Байеса-Дирихле необходимо определить положительную меру, заданную на $\{1, \dots, r_1\} \times \dots \times \{1, \dots, r_n\}$ и формируемую на основе следующего выражения:

$$\alpha(x_1, \dots, x_n) = \frac{1}{\prod_{i=1}^{n} r_i}, \tag{2.15}$$

где α – равновероятностная мера, позволяющая присвоить каждому вектору параметров одинаковое значение

$$\alpha_{i,j,k} = \alpha(x_i = k, P(x_i) = j) = \frac{1}{r_i q_i}. \quad (2.16)$$

Из выражения, описывающего логарифмическую форму метрики Байеса-Дирихле (2.12), и выражения для равновероятностной меры (2.14) эквивалентная метрика Байеса-Дирихле имеет вид [65]

$$ln\, P(D|G) = \sum_{i=1}^{n}\sum_{j=1}^{q_i} ln \frac{\Gamma\left(\frac{1}{q_i}\right)}{\Gamma\left(\sum_{s=1}^{r_i} N_{i,j,s} + \frac{1}{q_i}\right)} + \sum_{s=1}^{r_i} \frac{\Gamma\left(N_{i,j,s} + \frac{1}{q_i r_i}\right)}{\Gamma\left(\frac{1}{q_i r_i}\right)}. \quad (2.17)$$

Выражения (2.17), (2.16) используется в качества критерия максимизации процедуры локального поиска на основе алгоритма восхождения и является эквивалентной метрикой Байеса-Дирихле. При этом процедура сравнения двух метрик определяется на основе вычисления и последующего сравнения априорных параметров распределения байесовской сети.

В сочетании с алгоритмом ММРП, применение поиска с восхождением позволяет получить оптимальную структуру байесовской сети путем задания ограничения возможности добавления ребер только в том случае, если они были определены на этапе выполнения алгоритма минимаксный «родитель-потомок». Это дает возможность уменьшить число вычисленных эквивалентных метрик и оптимизировать процедуру поиска наиболее оптимального значения, за счет отбрасывания переменных, не входящих в результирующее множество CPC_X, задающего базовую структуру байесовской сети. В процессе локального поиска, если после 15 итераций значение оценки не увеличивается, алгоритм останавливается и возвращает наилучшее (максимальное) значение оценки для данного узла. Число итераций является эмпирически вычисленным и может принимать произвольное значение. Исходя из описанных особенностей алгоритма поиска с восхождением, увеличение числа итераций не всегда ведет к улучшению оптимума, так как при попадании в локальный максимум, глобальный оптимум так и не будет достигнут.

Этапы алгоритма минимаксного восхождения для модели байесовской сети, состоящей из 6 узлов, представлены на рисунке 2.1

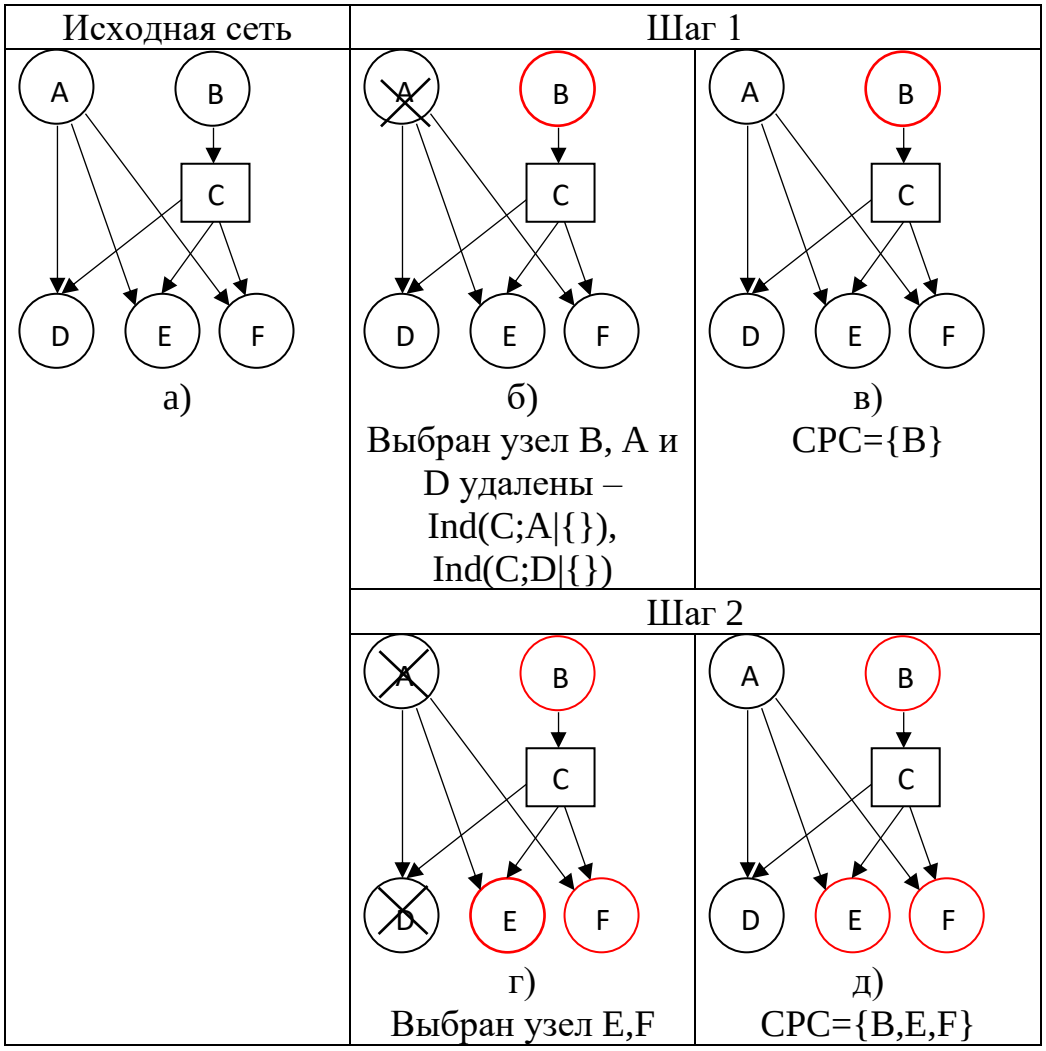

Рисунок 2.1. Этапы алгоритма минимаксного восхождения

Алгоритм минимаксного восхождения реализует логику построения базовой структуры БС и дальнейшее определение направленности ее ребер. Использование эвристической функции на основе статистических тестов позволяет установить связи между отдельными узлами сети с учетом сформированной обучающей выборки и тем самым повысить правдоподобие целевой структуры сети. Данный алгоритм обладает рядом недостатков, в частности, использование алгоритма поиска с восхождением иногда возвращает локальный оптимум, что ведет к некорректному формированию направленности связей и тем самым создает условия для появления некорректных значений в таблицах условных вероятностей байесовской сети.

Рассматривая возможность адаптации алгоритма минимаксного восхождения к семантике ДБС, необходимо определить структуру сети для каждого из временных срезов, включая формирование моделей перехода и восприятия. Данный алгоритм не позволяет это сделать, так как не учитывает временные стохастические связи между узлами сети. В рамках исследования, рассматривается возможность применение моделей ДБС для представления процессов тестирования. В связи с этим возникает необходимость разработки математических методов и алгоритмов, предназначенных для обучения структуры темпоральных БС, в частности ДБС с фиксированным числом временных состояний. В рамках диссертационного исследования будут разработаны методы, объединяющие как статистический подход к формированию топологии БС, так и механизмы определения направленностей дуг между узлами сети. В рамках построения топологии сети будет разработан метод и соответствующий алгоритм, реализующий вычисление марковского покрытия с учетом модели перехода между срезами ДБС. В состав марковского покрытия для соответствующих транзитивных переменных из среза $t+1$ будут попадать лишь узлы из соседнего среза t.

В основе метода построения структуры ДБС используется эвристический анализ всех допустимых связей для каждого узла Z с учетом определения его марковского покрытия M. Для вычисления марковского покрытия будем использовать последовательное проведение тестов на условную независимость для каждой вершины ДБС [67]. Применение данных тестов обусловлено необходимостью оценки устойчивости связи между дочерними и родительскими узлами. В основе проведения тестов на условную независимость в разработанном алгоритме используется G^2 критерий, математическая формулировка которого представлена в выражении (2.6). Между тем для определения факта наличия временных связей между срезами, будем использовать наборы обучающих выборок, соответствующие каждому из состояний временной сети t и $t+1$. Однако при этом предполагается, что выборки для момента времени $t+1$ имеют лишь набор значений из соответствующей модели перехода, остальные

переменные сети не будут включены в данную выборку. Выделим два основных шага данного метода обучения структуры ДБС.

На первом шаге производится вычисление узлов-кандидатов, которые могут входить в состав марковского покрытия M' для рассматриваемой переменной Z. Далее происходит удаление узлов, ошибочно добавленных в состав M'. Удаление производится за счет повторного выполнения статистических текстов для каждой вершины Z при наличии допустимого подмножества $M \subset \mathrm{M}'$. Для описания алгоритма введем следующие обозначения $Z(t) = Z_t$ – множество переменных для момента времени t, аналогично $Z(t+1) = Z_{t+1}$ – для момента времени $t+1$. Выражение марковского покрытия, соответствующего вершине Z ДБС, будет иметь следующий вид:

$$M_{t:t+1} = M_t \cup M_{t+1}. \tag{2.18}$$

При формировании марковского покрытия между переменной Z слоя Z_t и переменными Z_{t+1}, необходимо учитывать только дочерние узлы C_{t+1}, поэтому выражение (2.18) можно переписать в следующем виде:

$$M_{t:t+1} = M_t \cup \mathrm{C}_{t+1}, \tag{2.19}$$

где C_{t+1} – дочерние вершины для узла Z рассматриваемого среза $t+1$.

Приведем основные шаги построения структуры ненаправленной ДБС [184].

1. Задаются начальные значения для выполнения алгоритма: текущая переменная Z, множество обучающих данных D, множество кандидатов в состав марковского покрытия ${M_{t:t+1}}' = \emptyset$.

2. Выполняется итерация среди всех узлов сети и находится переменная F, для которой достигается максимальное значение среди минимальных показателей устойчивости связи между целевой переменной Z и текущей переменной X_i, при условии наличия всех возможных подмножеств $M^* \subset {M_{t:t+1}}'$. Переменная F добавляется в множество ${M_{t:t+1}}'$.

3. На следующем шаге происходит формирование результирующего марковского покрытия для каждой из вершин за счет исключения узлов, ошибочно добавленных в ${M_{t:t+1}}'$. Удаление узлов происходит путем проведения G^2 тестов на условную независимость вершины Z при наличии подмножества $M^* \subset {M_{t:t+1}}'$.

Если соблюдается правило d – разделенности, то текущая вершина удаляется из ${M_{t:t+1}}'$.

По результатам выполнения алгоритма для каждой из вершин ДБС определим состав марковского покрытия и окончательную математическую модель сети с учетом рассматриваемой области исследования. Расчет статистических параметров для каждой вышины ДБС позволяет определить те узлы, которые имеют непосредственные связи в смежных временных срезах. В тоже время, определение базовой структуры ДБС не дает возможность оценить направленность ребер сети [182].

Задача установления направленности связей между вершинами ДБС и представление сети в виде направленного ациклического графа решается в работе в виде оптимизации функции $L(G, D)$, связанной с операциями добавления, удаления и изменения направленности связей между узлами ДБС. В качестве оценочной функции могут быть использованы следующие метрики: логарифм правдоподобия, критерии Шварца, Акаике и Байеса-Дирихле.

Структура разработанного метода марковского покрытия с применением статистических оценок (МПСО) для поиска направленной структуры графа ДБС состоит из следующих основных шагов.

Шаг 1. Для выбранной вершины графа X_i с помощью алгоритма имитации отжига на основе вычисления оценочных критериев производится добавление, удаление и изменение направления ребер между всеми возможными вершинами. При этом операция добавления дуги из $X_{i+1} \rightarrow X_i$ возможна только в том случае, если X_{i+1} входит в состав марковского покрытия для X_i. В результате получаем множество возможных родительских $Parents(X_i)$ и дочерних вершин $Children(X_i)$. Дугам приписываются веса, вычисляемые через заданные оценочные критерии.

Шаг 2. Дуга с максимальным весом добавляется в результирующий направленный ациклический граф ДБС.

Шаг 3. Повторяются шаги 2 и 3 для всех узлов ДБС.

Шаг 4. Получается направленный ациклический граф ДБС с максимальными значениями весов для каждого ребра.

Для решения задачи поиска максимального значения оценки для каждой из дуг $X_{i+1} \to X_i$ ДБС может применятся ряд методов: поиск с восхождением, имитация отжига, поиск с запретами, K2-алгоритм. В рамках предложенного в работе метода предлагается использовать стохастический метод имитации отжига.

Исходными данными для метода имитации отжига [62] служит набор обучающих выборок D. Данный набор D содержит значения вершин ДБС для нескольких смежных временных срезов $t = t_1, t_2, \dots, t_n$. В процессе выполнения имитации отжига рассматривается максимизация целевой оценочной функции на множестве дуг марковского покрытия

$$\max_{x \in M} f(x) \tag{2.20}$$

В основе метода имитации отжига используется распределение Больцмана, характеризующее вероятностью P_j нахождения термодинамической системы в состоянии j для значения функции $f(x_j)$ при температуре T

$$P_j = \frac{exp\left(-\frac{f(x_j)}{kT}\right)}{Q} = \frac{exp\left(-\frac{f(x_j)}{kT}\right)}{\sum_{j=1}^{N} exp\left(-\frac{f(x_j)}{kT}\right)}, \tag{2.21}$$

где N – число состояний, которое может принимать рассматриваемая термодинамическая система.

На каждом этапе метода имитации отжига происходит постепенное повышение энергии $\Delta f(x) = f(x_{k+1}) - f(x_k)$ и переход системы в новое состояние x_{k+1}. Каждый последующий переход $x_k \to x_{k+1}$ в новое состояние определяется в соответствии с распределением Больцмана

$$P(x_{k+1}|x_k) = exp\left(-\frac{\Delta f(x)}{kT}\right), \tag{2.22}$$

где k – постоянная Больцмана.

В рамках метода имитации отжига будем использовать аппроксимацию постоянной Больцмана $k = 1$. Тогда вероятность того, что система перейдет в

новое состояние с учетом приращения $\Delta f(x)$ будет определяться следующим выражением:

$$P(x_k^* \to x_{k+1}|x_k) = \begin{cases} P(x_{k+1}|x_k), \text{если } \Delta f(x) > 0 \\ 1, \text{если } \Delta f(x) \leq 0 \end{cases} \quad (2.23)$$

Из выражений (2.22) и (2.23) следует, что если переход из $x_k \to x_{k+1}$ достижим, то система переходит в новое состояние, в противном случае, остается в текущем состоянии x_k. Вычисления выполняются циклически до тех пор, пока не будет достигнуто значения максимума температуры T_{max}. В результате получаем максимум оценочной функции $f(x)$.

В результате применения описанных выше методов обучения структуры мы получаем ДБС, представляющую собой направленный ациклический граф с оптимальными оценочными значениями параметров сети и корректно определенной направленностью связей между узлами данной сети, где каждая из направленных дуг будет иметь максимум значения оценочной функции. Использование стохастического метода имитации отжига позволяет снизить риск попадания оценочной функции в локальные экстремумы и повысить точность определения вероятностных связей между узлами динамической байесовской сети как в рамках одного среза, так и нескольких временных срезов, взятых в хронологическом порядке.

2.2. Разработка методов обучения параметров динамических байесовских сетей

Обучение параметров сети предназначено для определения вероятностных взаимосвязей между фрагментами данных обучающей выборки D и вычислением распределения для всех вершин, входящих в состав байесовской сети. В процессе обучения байесовской сети возникает две возможные ситуации, определяющие специфику методов обучения параметров данных сетей. Каждая из ситуаций определяются в зависимости от присутствия в сети скрытых (ненаблюдаемых) переменных. Обучение параметров байесовской сети может осуществляться с

помощью полных данных (полная наблюдаемость) и в условиях присутствия скрытых переменных (частичная наблюдаемость). Рассмотрим более детально каждый из подходов к обучению параметров байесовской сети.

В основе метода байесовского обучения с помощью полных данных предполагается вычисление вероятностей для всех переменных байесовской сети на основе множества наблюдений D (обучающая выборка), а процесс обучения сводится к решению задачи вероятностного вывода. Вероятность определённой гипотезы (соответствия определенному значению) для каждой из переменных, характеризуемых множеством наблюдаемых значений [124] в текущей выборке $d \subset D$, имеет следующий вид:

$$P(H_i|d) = \alpha P(d|H_i)P(H_i) \tag{2.24}$$

В терминах байесовской сети, характеризующейся множеством переменных X, вероятность гипотезы появления события из $d \subset D$ для каждой переменной из множества X имеет следующий вид:

$$P(X|d) = \sum_{i=1}^{n} P(X|d, H_i)P(H_i|d) = \sum_{i=1}^{n} P(X|H_i)P(H_i|d) \tag{2.25}$$

Выражение (2.25) показывает, что предсказание представляет собой среднее значение каждой из гипотез по отдельности $H = \{H_1, H_2, \dots, H_1\}$. В данном случае, количественными показателями для байесовской сети будут распределение вероятности гипотез $P(H_i)$ и $P(d|H_i)$ – правдоподобие данных для каждой гипотезы $H_i \in H$. В качестве одного из основных методов обучения байесовских сетей является наивный байесовский классификатор. Сущность данного классификатора базируется на вычислении логарифма правдоподобия в сочетании с апостериорным правилом принятия решения (MAP), определяемым на основе следующего выражения [22,40]:

$$\theta_{MAP}^{G} = arg\,\max_{\theta}(\,lnP(X|\theta) + ln\,P(\theta)), \tag{2.26}$$

где G – граф байесовской сети, θ^G – множество параметров сети, $\ln P(X|\theta) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{i,j,k}\ln\theta_{i,j,k}^{G}$ – логарифм правдоподобия.

Использование MAP-правила позволяет производить предсказание с вероятностью близкой по значению к байесовскому распределению $\mathrm{P}(\mathrm{X}|d) \approx \mathrm{P}(\mathrm{X}|H)$, что дает возможность оптимизировать процедуру обучение сети. Важно отметить, что использование MAP-правила наиболее применимо в тех случаях, когда число параметров θ, входящих в состав байесовской сети, превышает размер обучающей выборки D. Такой подход способствуют снижению вероятности переобучения сети.

Для определения логарифма правдоподобия необходимо обозначить и ввести следующие ограничения. Рассмотрим выражение

$$P_\theta\big(X_i, Y, Pa(X_i)\big)P_\theta\big(Pa(X_i)\big) = P_\theta\big(X_i, Pa(X_i)\big)P_\theta\big(Y, Pa(X_i)\big)$$
$$= \sum_{\{X_{1,\dots,}X_n\}\setminus(\{X_i\}\cup Pa(X_i))} P_\theta(X_1, \dots, X_n) \sum_{\{X_{1,\dots,}X_n\}\setminus(Y\cup Pa(X_i))} P_\theta(X_1, \dots, X_n), \quad (2.27)$$

где $Pa(X_i)$ – множество родителей для переменной X_i, $Y \bigcap \mathrm{C}(X_i) = \oslash$, $\forall X_i \in \{X_{1,\dots,}X_n\}, \forall Y \in \{X_{1,\dots,}X_n\}/X_i$, $\mathrm{C}(X_i)$ – множество дочерних вершин для переменной X_i .

Факторизация вероятностного распределения переменных, принадлежащих множеству $X = \{X_{1,\dots,}X_n\}$, при выполнении некоторой гипотезы H может быть представлена в виде следующего выражения [31]:

$$P_\theta\big(X_{1,\dots,}X_n\big) = \prod_{i=1}^{n} P_\theta\left(X_i\middle|Pa(X_i)\right) = \prod_{i=1}^{n} \theta_{i,j,k}^{G}, \forall\theta \in \Theta_G \quad (2.28)$$

$$\theta_{i,j,k}^{G} = P_\theta(X_i = k|Pa(X_i) = j). \quad (2.29)$$

Далее предполагаем, что данные, используемые для обучения сети $D = \left\{X^l = \left(X_1^l, \dots, X_n^l\right)\middle|l = 1, \dots, N\right\}$, можно представить в виде набора N независимых случайных величин, имеющих одинаковое распределение вероятностей для всех значений параметров из множества θ^G. Проведем оценку параметров θ^G на основе метода максимального правдоподобия

$$P_{\theta^G}\big(X_1^l, X_2^l, \dots, X_n^l\big) = \sum_{i=1}^{n} \theta_{i,j,k}^{G} = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\big(\theta_{i,j,k}^{G}\big)^{1_{l_{i,j,k}}}, \quad (2.30)$$

$$1_{l_{i,j,k}} = \begin{cases} 1, X_i = k, Pa(X_i) = j \\ 0, \text{в противном случае} . \end{cases} \quad (2.31)$$

Учитывая тот факт, что все строки из множества $D = \{X^l = (X_1^l, \dots, X_n^l) | l = 1, \dots, N\}$ являются условно независимыми, вероятностное распределение (2.30) принимает следующий вид:

$$P_{\theta^G}(D) = \prod_{l=1}^{N} P_{\theta^G}(X_1^l, \dots, X_n^l) = \prod_{l=1}^{N} \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k}^G)^{1_{l_{i,j,k}}} = \\ = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k}^G)^{\sum_{i=1}^{N} 1_{l_{i,j,k}}}, \tag{2.32}$$

где $\sum_{i=1}^{N} 1_{l_i,j,k}$ – общее число строк в множестве D со значением i переменной равным k значению родителей, равным j.

Функция правдоподобия принимает следующий вид:

$$L(G, \theta^G, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{i,j,k} ln\, \theta_{i,j,k}^G \tag{2.33}$$

Для решения задачи условной оптимизации с учетом ограничения $\theta_{i,j,k}^G > 0, \sum_{k=1}^{r_i} \theta_{i,j,k}^G = 1$ используется функция Лагранжа

$$Lagr(G, \theta^G, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{i,j,k} ln\, \theta_{i,j,k}^G + \sum_{i=1}^{n} \sum_{j=1}^{q_i} u_{i,j} \sum_{k=1}^{r_i} \theta_{i,j,k}^G, \tag{2.34}$$

где $u_{i,j}$ – коэффициенты Лагранжа для ограничений, накладываемых на θ^G.

Решение (2.34) сформулируем в виде классической задача выпуклого программирования. Исходя из этого, сформулируем достаточные условия в следующем виде:

$$\frac{\partial}{\partial\, \theta_{i,j,k}^G} Lagr(G, \theta^G, D) = \frac{N_{i,j,k}}{\theta_{i,j,k}^G} + u_{i,j} = 0 \Rightarrow \frac{N_{i,j,k}}{\theta_{i,j,k}^G} = -u_{i,j}, \forall k \\ \Rightarrow \sum_{k=1}^{r_i} N_{i,j,k} = -u_{i,j} \sum_{k=1}^{r_i} \theta_{i,j,k}^G \Rightarrow N_{i,j} := \sum_{k=1}^{r_i} N_{i,j,k} = -u_{i,j}, \tag{2.35}$$

где $N_{i,j} = \sum_{k=1}^{r_i} N_{i,j,k}, \{l \in \{1, \dots, N\} | Pa(X_i) = j\}$.

Оценка максимального правдоподобия для байесовской сети будет определяться на основе следующего выражения [52]:

$$\frac{N_{i,j,k}}{\theta^{G}_{i,j,k}} = N_{i,j}, \forall k \Rightarrow \theta^{G}_{i,j,k} = \frac{N_{i,j,k}}{N_{i,j}}. \quad (2.36)$$

Выражение максимального правдоподобия для байесовской сети с дискретным законом распределения параметров представляет собой частотную оценку, характеризующую интенсивность поступления определенных значений в обучающей выборке.

Использование представленного подхода на основе наивного байесовского классификатора для обучения параметров становится трудоемким с ростом числа узлов сети, в связи с чем возникает необходимость предварительного упрощения структуры сети. В качестве одного из таких подходов рассмотрим метод упрощение БС на основе построения дерева сочленения (ДС). Использование ДС позволяет исключить стадию опроса первичной сети и реализовать переход к древовидным графам. Это позволяет снизить общее число этапов, необходимых для вычисления вероятностных исходов каждого из параметров сети. Применение ДС [209] по сравнению с классическими методами, позволяет сократить сложность вычислений и добиться определенной точности оценок параметров при меньшем объеме обучающей выборки.

Структура алгоритма построения дерева сочленений иметь следующий вид.

1. Морализация доменного графа G_m. Доменный граф является базовой формой представления байесовской сети, в котором отсутствуют направления ребер и какие-либо вероятностные характеристики, в частности, таблицы условных вероятностей. В процессе морализации происходит трансформация доменного графа. В него добавляются ребра по следующему принципу: для каждой вершины X_i соединяются ребрами узлы, попавшие в множество $C_{X_i} = \{Parents(X_i)\}$ (множество родительских узлов, имеющих непосредственную связь с переменной X_i). Применение морализации имеет место для любого числа узлов, имеющих общие дочерние вершины.

2. Триангуляция. Граф G_t является триангулярным, только в том случае, если в нем присутствуют циклы, число вершин в которых не более четырех. В основе процедуры построения триангулярного графа используется механизм

последовательного исключения узлов графа в установленном порядке. На начальном этапе создается полная копия G_1 графа G_m. Повторяем цикл с выбором узла V с минимальным числом дочерних вершин пока G_1 не пуст. В том случае, если существует несколько вершин, удовлетворяющих условию минимальности, то выбираем вершину, имеющую наименьший вес. Весом в данном случае является произведение значений переменной, ассоциируемой с узлом БС, и всех ее соседних вершин. Если одновременно имеется несколько вершин с наименьшим весом, то выбираем любую из них. Соединяем все узлы смежных вершин V. Для графа G процедура добавления дуг является аналогичной [213]. Производится удаление узла V из графа G_1. В результате выполнения всех операций граф G_m преобразуется в триангулярный граф G_t.

3. Построение дерева объединений. Под деревом объединений будем понимать некоторый древовидный граф, где любой его узел является подмножеством вершин триангулярного графа и две вершины этого подмножества соединены между собой [214]. Данное множество вершин графа называется его кликой.

Построение дерева объединений.

– Пусть задан триангулярный граф G_t. Инициализируем переменную-счетчик $i = 0$.

– Повторяем выбор вершины V с минимальным числом смежных узлов из графа $\mathrm{G_t}$ пока он не пуст. Если узлов, удовлетворяющих данному условию несколько, то выбираем вершину с минимальным весом. Множество, в которое входит сама вершина вершины V и все ее соседние узлы, помечаем как C_i. Обозначим S_i как множество-сепаратор, содержащее все вершины V из C_i, имеющих смежные узлы, не входящие в состав C_i. Удаляем из графа G_t все соседние с V вершины, не вошедшие в S_i. Присваиваем счетчику значение $i + 1$. Повторяем данный шаг.

– Поочередно соединяем все узлы клики C_i с сепаратором S_i. Кроме того, последний созданный узел остается несвязанным с другими узлами в виду того, что число сепараторов на единице меньше общего числа вершин дерева.

– Соединяем каждый сепаратор S_i с узлом C_j с учетом условий: $j > i$, все элементы сепаратора S_i будут являться подмножеством клики C_j.

– Полученная структура, состоящая из клик триангулярного графа, узлы которой связаны через сепараторы, будет являться деревом объединений.

4. Построение дерева сочленений. Деревом сочленений для БС является дерево объединений, для которого любой вершине соответствует множество потенциалов условных вероятностей и каждому потенциалу соответствует вершина, которая полностью содержит его область определения.

Потенциалы, входящие в состав дерева сочленения, представляют собой дискретные функции нескольких переменных следующего вида:

$$P(x|y_1, y_2 \dots) = \phi(x, y_1, y_2, \dots).$$

Для сокращения области определения потенциала используют маргинализацию, которая производится следующим образом:

$$\varphi(x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_N) = \sum_{x_k} \phi(x_1, x_2, \dots, x_{k-1}, x_k, x_{k+1}, \dots, x_N).$$

Аналогичным образом работает проецирование – маргинализация всех переменных, не принадлежащих множеству относительно которого выполняется проецирование

$$\psi(x_1, \dots, x_k) = \phi(x_1, x_2, \dots, x_N)^{\downarrow(x_1,\dots,x_k)} = \sum_{x_{k+1},\dots,x_N} \phi(x_1, x_2, \dots, x_N).$$

При перемножении потенциалов используем правило [212]

$$\psi(x_1, \dots, x_n, y_1, \dots, y_m, z_1, \dots, z_k) =$$
$$\sigma(x_1, \dots, x_n, y_1, \dots, y_m)\phi(y_1, \dots, y_m, z_1, \dots z_k).$$

В результате будем иметь структуру, состоящую из узлов, связанных через соответствующие сепараторы. Каждая вершина, входящая в состав данной структуры, является кликой триангулярного графа, а область определения потенциалов полностью содержится во множестве переменных, принадлежащих данной клике. В процессе построения дерева сочленений возможна ситуация, когда

сепараторы будут включать в себя потенциалы, областью определения которых будет являться множество переменных сепаратора. Для рассматриваемого случая потенциалы содержатся лишь в узлах дерева. Реализовав процедуру построения дерева сочленений БС, опишем основные этапы проведения опроса сети.

5. Опрос сети. Опрос сети осуществляется для вычисления распределения вероятностей вершин байесовской сети. Для опроса дерева сочленений используется алгоритм распространения доверия (алгоритм Шефера-Шеноя). Данная процедура структурно делится на два этапа: сбор и распространение сведений. Предварительно необходимо выбрать произвольную вершину V, принадлежащую одной из вершин дерева сочленения C_i в качестве его корня [202].

Сбор сведений. Выберем узел ДС $N \notin \{V\}$, включающий в себя только один сепаратор S_i. Выполним перемножение всех потенциалов, связанных с вершиной N и спроецируем полученный результат на сепаратор S_i. Выполним добавление потенциала в сепаратор S_i. Осуществим поиск вершины, содержащей один смежный сепаратор, который не включает потенциала. Перемножим потенциалы данной вершины на потенциалы всех смежных с нею вершин. Проецируем полученное произведение на множество смежных сепараторов. Все смежные с вершиной N узлы будет иметь потенциал. Выполняем перемножение всех потенциалов для N на потенциалы всех смежных вершин и проецируем результат на $\{V\}$. В завершении получаем, что потенциал ψ_i имеет непосредственную зависимость лишь от V и после нормировки будет содержать искомые вероятности.

Распространение сведений. Выбираем такую вершину V, для которой выполнен сбор сведений. Пусть $\{S_i\}$ и $\{\psi_i\}$ – множества сепараторов и потенциалов, ассоциируемых с данной вершиной. Множество потенциалов, полученных в результате сбора сведений, обозначим как $\{\psi_i'\}$. S' множество сепараторов всех вершин, смежных с V. Поместим в каждый сепаратор S_j потенциал из S'

$$\psi_j' = \left(\phi_V \prod_{S_i \in S'} \psi_i \right)^{\downarrow(S_j)}, i \neq j, \tag{2.37}$$

где ϕ_V – потенциал, соответствующий вершине V.

Тогда значение вероятности переменной $V \in T$ можно вычислить на основе следующего выражения:

$$P(V) = \left(\phi_V \prod_{S_i \in S'} \psi_i \right)^{\downarrow(V)} . \quad (2.38)$$

Найдем вершину $V' \in V$, содержащую два и более сепаратора, один из которых имеет два потенциала. Пусть S'' – множество сепараторов, связанных с вершиной V', $\phi_{V'}$ – потенциал вершины V'. Тогда можно поместить потенциал ψ'_j в сепаратор S_j

$$\psi'_j = \left(\phi_{V'} \psi'_k \prod_{S_i \in S''} \psi_i \right)^{\downarrow(S_j)} , i \neq j \neq k. \quad (2.39)$$

Вероятность для $V \in V'$ найдем исходя из следующего выражения:

$$P(V) = \left(\phi_{V'} \psi'_k \prod_{S_i \in S''} \psi_i \right)^{\downarrow(V)} , i \neq k. \quad (2.40)$$

Повторяем процедуру заполнения потенциалов сепараторов для каждой вершины V' из ДС пока каждый из сепараторов S_j не будет включать в себя только два потенциала. В результате выполнения процедуры распространения сведений получим искомое распределение вероятностей $P(V)$ по всем вершинам ДС.

Классический подход обучения параметров БС является достаточно трудоемким, если заданная сеть имеет большое число узлов. Это обусловлено тем, что необходимо выполнить перебор всего домена возможных значений, который может принимать конкретный узел байесовской сети, а если узел имеет условную зависимость от родительских вершин, то требуется проанализировать все возможные сочетания для значений родительских вершин по отношению к текущей вершине. В свою очередь это ведет к возможности получения ошибок обучения и неточному определению таблиц условных вероятностей для байесовской сети. В связи с этим возникает необходимость в проведении оптимизации процедур обучения параметров.

Одним из методов оптимизации обучения является бустинг (boost). Данный

метод является представителем семейства методов оптимизации на основе ансамблей. Основная идея данного метода заключается в оценке не единичной модели БС методом максимального правдоподобия, а ансамблей классификаторов байесовской сети. В качестве классификатора может выступать наивный байесовский классификатор на основе алгоритма максимального правдоподобия. В процессе выполнения алгоритма бустинга происходит циклический вызов классификатора. На каждой итерации алгоритма веса каждого неправильно классифицированного элемента возрастают, что способствует акцентированию внимания классификатора на следующем проходе. По завершении каждого шага происходит обновление распределения весов w, в соответствии со степенью важности каждого образца данных обучающего множества D. Одним из наиболее распространенных алгоритмов бустинга является алгоритм адаптивного усиления (AdaBoost). Исходными данными для данного алгоритма являются следующие параметры: набор данных $x = \{x_1, x_2, \dots, x_n\}$ и множество ответов $r = \{r_1, r_2, \dots, r_n\}, r_i \in \{-1,1\}$. Веса, связанные с данными параметрами, инициализируются одинаковым значением $w_n = 1/n$. В связи с этим сумма всех весов обучающих выборок равна $\sum_{i=1}^{n} w_i = 1$. Структурно алгоритм состоит из следующих основных шагов.

Шаг 1. Построение модели ансамблей F_m за счет обучения $y_m(x)$

$$F_m = \sum_{i=1}^{n} w_i^m [y_m(x_i) \neq r_i], \tag{2.41}$$

где $y_m(x_i)$ – классификатор, w_i^m – веса, соответствующие каждому набору обучающих данных из множества D.

Шаг 2. Рассчитываем функцию ошибок

$$\varepsilon_m = \frac{\sum_{i=1}^{n} w_i^m [y_m(x_i) \neq r_i]}{\sum_{i=1}^{n} w_i^m}. \tag{2.42}$$

Шаг 3. Обновить взвешенное нормированное распределение образцов для каждой из обучающих выборок

$$w_i^{m+1} = \frac{w_i^m e^{-\alpha_m [y_m(x_i) \neq r_i]}}{\Theta_i^m}, \tag{2.43}$$

$$\Theta_i^m = \sum_m w_i^m e^{-\alpha_m [y_m(x_i) \neq r_i]}, \tag{2.44}$$

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_m}{\varepsilon_m} \right). \tag{2.45}$$

Шаг 4. Произвести предсказание

$$Y_m(x) = sign \left(\sum_{m=1}^{M} \alpha_m y_m(x) \right). \tag{2.46}$$

Анализируя алгоритм адаптивного усиления, можно прийти к выводу, что основная особенность алгоритма заключается в возможности реализации процесса обучения слабых классификаторов с использованием одного и того же набора обучающих данных, которым приписываются соответствующие весовые коэффициенты на различных этапах алгоритма. Повышение значений весов происходит для тех данных, на которых классификатор допустил ошибки: наибольшие веса w_i^m присваиваются образцам данных $S_i \subset D$ с наименьшим значением величины отступа G

$$G = y \sum_{m^*} \alpha_{m^*} [y_{m^*}(x_i) \neq r_i], m^* = 1, \dots, m-1. \tag{2.47}$$

Величина отступа G определяет разницу между двумя конкретными образцами и имеет положительное значение, если классификация образца произведена правильно, в противном случае значение отступа принимает отрицательное значение. Использование отступа позволяет оптимизировать общую процедуру переприсваивания весов для определенного образца данных S_i в процессе классификации.

Использования бустинга позволяет повысить качество обучения БС на основе оптимизации за счет использования приемлемых механизмов обучения параметров сети. Данная оптимизация заключается в эмпирическом подборе классификатора, позволяющего получить минимальную ошибку в процессе

обработки образцов данных $S_i \subset D$. В свою очередь, использование бустинга приводит к повышению временной сложности в процессе выполнения алгоритма, так как необходимо производить повторную итерацию по всем данным в процессе выбора оптимального классификатора.

Рассмотрим обучение параметров байесовской сети со скрытыми переменными и неполными данными. В основе данного типа обучения лежит упрощенное предположение, что скрытые переменные независимы от переменных наблюдений. Переменные наблюдения представляют собой узлы байесовской сети, для которых происходит определение значений таблиц условных вероятности в процессе поступления свидетельства e. В процессе обучения параметров байесовской сети со скрытыми переменными невозможно использовать оценку методом максимального правдоподобия в явном виде, как было рассмотрено ранее. Для решения задач обучения в случае наличия в выборке скрытых данных, приведем первоначально модель, формируемую на основе полных данных Z. Данная модель имеет плотность распределения $f(Z|\theta)$, которая зависит от неопределенного параметра θ. Множество всех параметров $Z = (X, H)$, описывающее математическую модель байесовской сети, представляет собой объединение скрытых H и наблюдаемых X параметров. В таком случае сформулируем выражение для логарифма правдоподобия параметра θ для БС, включающее в себя параметры Z, состоящие из множества наблюдаемых и ненаблюдаемых параметров

$$L(\theta|X) = \int f(X, H|\theta)\, dH. \tag{2.48}$$

С учетом того, что выражение правдоподобия (2.48) дифференцируемо и одномодально, оценку максимального правдоподобия параметра θ можно вывести путем дифференцирования (2.48)

$$S(\theta|X) = \frac{d \ln L(\theta|X)}{d\theta} = 0 \tag{2.49}$$

В том случае, если явно нельзя найти решение для уравнения (2.49), оценку параметра θ можно найти с использование алгоритма Ньютона-Рафсона

$$\theta^{t+1} = \theta^t + I^{-1}(\theta^t|X)S(\theta^t|X), \quad (2.50)$$

где $I(\theta^t|X)$ – наблюдаемая информация, полученная путем вычисления второй производной логарифма правдоподобия $L(\theta|X)$

$$I(\theta^t|X) = -\frac{d^2L(\theta|X)}{d\theta d\theta}. \quad (2.51)$$

Если логарифм правдоподобия $L(\theta|X)$ представляет собой выпуклую функцию, то значения параметров θ^t может быть определено на основе оценки максимального правдоподобия θ^*, соответствующей параметру θ. Данное условие выполняется, если $L(\theta|X)$ представляет собой квадратичную функцию от θ.

Для определения сходимости $L(\theta|X)$ воспользуемся методом функции вкладов. Для этого преобразуем выражение (2.50) путем замены наблюдаемой информации $I(\theta^t|X)$ на ожидаемую $\Lambda(\theta)$

$$\theta^{t+1} = \theta^t + \Lambda^{-1}(\theta^t)S(\theta^t|X), \quad (2.52)$$

$$\Lambda(\theta) = E(I(\theta|X)|\theta) = -\int \frac{d^2L(\theta|X)}{d\theta d\theta} f(X|\theta)dX. \quad (2.53)$$

Анализируя описанные алгоритмы решения уравнения (2.49) для оценки параметра θ необходимо вычислить матрицу вторых производных логарифма правдоподобия. Размер такой матрицы напрямую зависит от числа параметров θ. Как следствие, при достаточно большом количестве параметров θ алгоритм ведет к повышению временной сложности и требует рассмотрения дополнительных мер оптимизации.

Для обхода ограничений, представленных выше алгоритмов, рассмотрим алгоритм ожидания-максимизации (ОМ). Алгоритм ОМ использует для нахождения оценок параметров байесовской сети со скрытыми переменными метод максимального правдоподобия [11]. Данный алгоритм обладает достаточно хорошей сходимостью. Она достигается за счет пошагового увеличения значения логарифма $L(\theta|X)$ при каждом проходе алгоритма. При этом, если $L(\theta|X)$ ограничен, то совокупность логарифмов $L(\theta^t|X)$ для каждого шага цикла будет сходиться к стационарному значению $L(\theta|X)$. Если последовательность параметров обучающей выборки $\theta^t \subset D$ сходится, то она должна сходиться к

локальному максимуму или точке перегиба. Скорость сходимости ОМ обратно пропорциональна общему числу ненаблюдаемых переменных БС. В общем случае для алгоритма ОМ выполняется замена ненаблюдаемых параметров на оценки. После чего производится оценивание параметров вплоть до сходимости процесса.

Логически структура алгоритма разделена на два основных этапа: ожидание и максимизация. В процессе ожидание происходит вычисление функции правдоподобия, при этом полагается, что в байесовской сети отсутствуют скрытые переменные. Логарифм правдоподобия для вычисления оценок параметров байесовской сети со скрытыми переменными H имеет следующий вид:

$$L(\theta) = \ln\big(P(X|\theta)\big) = \ln\left(\sum_{H} P(X, H|\theta)\right), \tag{2.54}$$

где X – наблюдаемые переменные, H – скрытые переменные.

С учетом того, что функция правдоподобия является выпуклой, то для нее справедливо неравенство Йенсена [34], определяемое на основе следующего выражения:

$$f\left(\sum_{i=1}^{n} \alpha_i \xi_i\right) \geq \sum_{i=1}^{n} \alpha_i f(\xi_i), \tag{2.55}$$

где $\xi = \xi_1, \xi_2, \dots, \xi_n$ – значения, принадлежащие области определения функции f, $\alpha = \alpha_1, \alpha_2, \dots, \alpha_n$ – положительные числа, сумма которых равна единице.

Переписывая выражение логарифма правдоподобия (2.54) с учетом неравенства Йенсена (2.55), получаем

$$L(\theta) = \ln\left(\sum_{H} q(\theta; H)\frac{P(X, H|\theta)}{q(\theta; H)}\right) \geq \sum_{H} q(\theta; H)\ln\left(\frac{P(X, H|\theta)}{q(Q; H)}\right), \tag{2.56}$$

где $q(\theta; H)$ – распределение для ненаблюдаемых переменным, для которых имеет место следующее условие:

$$q(\theta; H): 0 \leq q(\theta; H) \leq 1, \sum_{H} q(\theta; H) = 1 \tag{2.57}$$

Неравенство Йенсена позволяет задать нижнюю границу логарифма правдоподобия $L(\theta)$. В свою очередь для получения оптимальной оценки

фиксированных параметров θ', необходимо рассмотреть случай, когда неравенство Йенсена обращается в равенство. Данная ситуация имеет место, когда случайная величина, для которой справедливо неравенство, совпадает со своим математическим ожиданием

$$\frac{P(X,H|\theta')}{q(\theta';H)} = \sum_H P(X,H|\theta') = P(X|\theta'), \forall H. \quad (2.58)$$

Используя утверждение, приведенное в равенстве (2.58), определим выражение для $q(\theta;H)$ в следующей форме:

$$q(\theta';H) = \frac{P(X,H|\theta')}{\sum_H P(X,H|\theta')} = \frac{P(X,H|\theta')}{P(X|\theta')} = P(H|X,\theta'). \quad (2.59)$$

Из выражения (2.59) следует, что $q(H)$ является апостериорным распределением вероятностей по всем переменным H с учетом всех значений наблюдаемых переменных X и заданным множеством параметров θ'.

Шаг максимизации характеризуется вычислением максимальной оценки [158,164] для нижней границы логарифма правдоподобия (2.54) по параметру θ'

$$\theta' = \arg\max_{\theta'} \left[\sum_H q(\theta;H) \ln\left(\frac{P(X,H|\theta)}{q(\theta;H)}\right)\right]. \quad (2.60)$$

Анализируя этапы алгоритма ОМ видно, что в процессе максимизации функции правдоподобия имеет место случай, когда в результате поиска экстремума функции можно получить большое число локальных оптимумов, что в свою очередь повышает зависимость между входными значениями алгоритма и начальным приближением. Все это приводит к общему замедлению сходимости логарифма правдоподобия.

Рассматривая алгоритм ОМ с точки зрения сходимости, необходимо определить ряд ограничений, накладываемых исходя из особенностей вычисления функции правдоподобия $L(\theta)$. Для установления сходимости $L(\theta)$ на шаге максимизации необходимо учитывать, что распределения должны удовлетворять условию регулярности, в частности, гладкости и монотонности. Для установления данного факта используется метрика Кульбака-Лейблера (КЛ). Метрика КЛ является мерой сходства между истинной функций распределения $f_\theta(X|H)$ и

приближенной моделью $f_\theta{}'(X|H)$. В классической интерпретации метрика Кульбака-Лейблера представляет собой математическое ожидание логарифма правдоподобия и имеет следующий вид:

$$D_{KL}(f, f') = \int f_\theta(H|X) \ln \frac{f_\theta(H|X)}{f_\theta{}'(H|X)} dH. \quad (2.61)$$

Рассматривая выражение (2.61) с точки зрения теории информации, можно прийти к выводу, что метрика КЛ представляет собой меру удаленности двух распределений $f_\theta(H|X)$ и $f_\theta{}'(H|X)$. В данном случае (2.61) описывает значение потерянной информации. Если значение метрики КЛ $D_{KL}(f, f') = 0$, это свидетельствует об отсутствие потерь, а значение распределения $f_\theta(H|X)$ тождественно равно $f_\theta{}'(H|X)$. Для получения наименьших потерь, необходимо выбрать распределение $f_\theta{}'(H|X)$, имеющее наименьшее значение $D_{KL}(f, f')$. Исходя из этого, можно сформулировать два основных ограничения, присущие метрике КЛ: $D_{KL}(f, f') \geq 0, \forall\theta = \theta' \Rightarrow D_{KL}(f, f') = 0$.

Выражение (2.61) может быть представлено в виде разности двух математических ожиданий (2.62) для функций f и f' соответственно

$$\begin{aligned} D_{KL}(f, f') = &\int f_\theta(H|X) \ln f_\theta(H|X)\, dH \\ &- \int f_\theta(H|X) \ln f_\theta{}'(H|X) dH = E_f[\ln f_\theta(H|X)] \\ &- E_{f'}[\ln f_\theta{}'(H|X)]. \end{aligned} \quad (2.62)$$

Для проверки факта, что функции $Q(\theta, \theta') = \ln f_\theta{}'(X) + E_{f'}[\ln f_\theta{}'(H|X)]$ достигает своего наибольшего значения при заданном θ, необходимо определить, что она является монотонной. Для этого в качестве функции потерь будем использовать метрику КЛ $D_{KL}(f, f')$. В таком случае запишем выражение для потерянной информации с использованием выражения (2.62)

$$Q'(\theta, \theta') - D_{KL}(f, f') = \ln f_\theta{}'(X) - E_f[\ln f_\theta(H|X)] + E_{f'}[\ln f_\theta{}'(H|X)]. \quad (2.63)$$

В правой части формулы (2.63) слагаемое $E_f[\log f_\theta(H|X)]$ не зависит от θ, а его значение равно условной энтропии $I(H|X)$. В связи с этим, выражение (2.63) может быть преобразовано к следующему виду:

$$\arg\max_{\theta}\left(Q'(\theta,\theta') - D_{KL}(f,f')\right) = \arg\max_{\theta}\left(\ln f_{\theta}{}'(X) + E_{f'}[\ln f_{\theta}{}'(H|X)]\right). \quad (2.64)$$

Раскрывая выражение для $Q(\theta,\theta')$, формула (2.64) может быть приведена к следующей форме:

$$\arg\max_{\theta}\left(Q'(\theta,\theta') - D_{KL}(f,f')\right) = \arg\max_{\theta}\left(Q(\theta,\theta')\right). \quad (2.65)$$

Исходя из определения метрики КЛ $D_{KL}(f,f')$ и на основе выражения (2.65), можно сформулировать следующие рекуррентные правила, характерные для любой монотонной функции [109]

$$\theta' = \arg\max_{\theta}\left(Q'(\theta,\theta') - D_{KL}(f,f')\right)$$
$$\theta' = \arg\max_{\theta}\left(Q(\theta,\theta')\right) \quad (2.66)$$

В таком случае получаем, что логарифм правдоподобия является монотонной функцией, для которой справедливо неравенство:

$$\ln f_{\theta}{}'(H|X) \geq \ln f_{\theta}(H|X). \quad (2.67)$$

Из выражения (2.67) вытекает основное условие сходимости алгоритма ожидания максимизации, а сам алгоритм является монотонным.

В рамках решения задачи оптимизации алгоритма ОМ воспользуемся его стохастической версией (СОМ). Сущность СОМ заключается в возможности «встряхивании» значений наблюдаемых переменных на каждой итерации алгоритма [97]. Предполагаем, что наблюдаемая выборка X разбивается на множество классов $K = K_1, K_2, \dots, K_n$ таких, что

$$X = K_1 \cup K_2 \cup \dots \cup K_n. \quad (2.68)$$

В процессе определения исходных данных полагается, что каждый элемент выборки $X_i \in X$ соответствует единственному классу $K_j \in K, K_i \cap K_j = \emptyset$, а алгоритм ожидания максимизации дополняется шагом семплирования, предшествующему шагу ожидания. На этапе семплирования происходит стохастическое моделирование, заключающееся в формировании вектора $\tau_i = \tau_{i1}, \tau_{i2}, \dots, \tau_{in}$ из полиномиального распределения с параметром g_{ij}. Данный параметр численно равен значению вероятности, при котором величина вектора τ_{ij}

обращается в единицу. На основе вектора τ_i происходит определение принадлежности выборки X некоторому классу K_j и общей численности каждого из таких кластеров $\vartheta_i = \vartheta_1, \vartheta_2, \dots, \vartheta_k$

$$\vartheta_i = \sum_{j=1}^{n} \tau_{ij}, \vartheta_1 + \vartheta_2 + \cdots + \vartheta_k = n. \tag{2.69}$$

На основании этого, значение логарифма функции правдоподобия можно записать в следующей форме:

$$\ln L(\theta) = \sum_{i=1}^{k} \vartheta_i \ln \theta' + \sum_{i=1}^{k} \sum_{j:X_j \in K_i} \ln \varphi_i(X_j, \gamma_i), \tag{2.70}$$

$$\sum_{j:X_j \in K_i} \ln \varphi_i(X_j, \gamma_i) = \ln \sum_{j:X_j \in K_i} \varphi_i(X_j, \gamma_i) = \ln L(\gamma_i; K_i) \tag{2.71}$$

где $\varphi_i(X_j, \gamma_i)$ – плотность распределения подвыборки параметров, полученных из кластера K, γ_i – параметр, принадлежащий данной подвыборке.

Используя метод неопределенных множителей Лагранжа, можно доказать утверждение, что первый множитель, представленный в выражении (2.70), принимает максимум по некоторому набору параметров $\theta'_1, \theta'_2, \dots, \theta'_k$ при выполнении ограничения

$$\sum_{i=1}^{n} \theta'_i = 1,$$

$$\theta'_i = \frac{\vartheta_i}{n}. \tag{2.72}$$

Исходя из данных предположений, оценка правдоподобия параметра γ_i для каждого кластера K может быть получена на основе следующего выражения:

$$\gamma_i = \arg\max_{\gamma} L_i(\gamma, K_i). \tag{2.73}$$

Выражение (2.73) характеризует этап максимизации алгоритма COM и позволяет произвести оптимальную оценку параметров γ_i методом максимального правдоподобия. Важно отметить, что шаг ожидания в процессе выполнения алгоритма COM аналогичен классическому алгоритму ожидания-максимизации и не требует дополнительных действий.

Альтернативой алгоритму СОМ является ряд алгоритмов, позволяющих улучшить оценки параметров без разбиения их на классы. Между тем, это дает возможность получить лишь приближенную оценку. Наиболее известным алгоритмов, использующим данный подход, является ОМ с регуляризацией (ОМР). В качестве регуляризатора используется классический аддитивный регуляризатор. Принципиальное отличие данного алгоритма заключается в суммировании нескольких регуляризаторов и логарифма правдоподобия с последующей максимизацией. В таком случае происходит преобразование лишь этапа максимизации, в то время как процедура ожидания остается без существенных изменений

$$L(\theta) = \ln\left(P\left(X\middle|L(\theta)\right)\right) + \beta R(\theta), \tag{2.74}$$

$$\theta' = \arg\max_{\theta'}\left[\sum_{H} q(H)\ln\left(\frac{P(X,H|\theta)}{q(H)}\right)\right] + \beta R(\theta), \tag{2.75}$$

где β – положительный коэффициент регуляризации.

При определенных условиях, в процессе вычисления распределения $q(\theta;H)$, использование алгоритма ОМ становится затруднительно в силу прямой зависимости процедуры обучения от качества начальной выборки. Для решения данной проблемы можно воспользоваться семплированием на основе метода Монте-Карло. Основная сущность данного подхода заключается в определение некоторой функции $q'(\theta;H)$, которая ставится в соответствие искомой функции $q(\theta;H)$. Вычисление $q'(\theta;H)$ может быть выполнено на основе имитационной модели и представлено в виде следующего выражения

$$q'(\theta;H) = \frac{1}{N_m}\sum_{i=1}^{N_m}\ln f_\theta(H,X_i), \tag{2.76}$$

где $X = X_1, X_2, \dots, X_{N_m}$ – независимые случайные значения, соответствующие наблюдаемой переменной X и формируемые из распределения $f_{\theta_m}(X|H)$.

Пусть N_m принимает достаточно большое значение, то в силу закона больших чисел можно сделать следующее приближение:

$$q'(\theta; H) \approx q(\theta; H) \tag{2.77}$$

Рассматривая возможность применение метода Монте-Карло к алгоритму ОМ, получим, что для каждой итерации $m+1$ осуществляется поиск значения θ для текущей итерации θ^{m+1} в соответствии со следующим неравенством:

$$q'(\theta^{m+1}; H) \geq q(\theta^{m}; H), \forall\theta \in \Theta. \tag{2.78}$$

Для этапа максимизации приближение (2.77) порождает некоторую погрешность ξ^m следующего вида:

$$\xi^m(N_m) = \int [\ln f_\theta(X, H)]\, f_\theta(H|X)\mu_H dH - \frac{1}{N_m}\sum_{i=1}^{N_m} \ln f_\theta(H, X_i), \tag{2.79}$$

где μ_H – медиана случайной величины H.

Анализируя формулу для вычисления ошибки (2.79), можно прийти к выводу, что принимаемые значения имеет обратную зависимость от размерности имитируемой выборки N_m – чем больше размерность выборки, тем меньше значение ошибки $\xi^m(N_m)$. Получаем, что точность алгоритма ОМР прямопропорциональна значению параметра N_m. В тех случаях, когда на начальных этапах алгоритма нет необходимости формировать достаточно большое число выборок, все равно требуется пошаговое увеличение выборок на каждой последующей итерации. Это обусловлено обеспечением требуемой точности алгоритма и суммарным снижением результирующей ошибки ξ^m.

Рассмотрев классический алгоритм ОМ и его возможные оптимизации, можно сделать вывод, что каждый из подходов имеет одновременно ряд преимуществ и недостатков. Одни алгоритмы не учитывают вклад начального распределения, другие не обеспечивают требуемый уровень точности из-за ограничений по числу генерируемых выборок. Одним из путей решения данных проблем является использование гибридных алгоритмов, позволяющих преодолеть вышеуказанные ошибки. Алгоритм ОМ с применением метода Монте-Карло является гибридным алгоритмом, объединяющим в себя СОМ и ОМ и направленный на снижение ошибки, приведенной в формуле (2.79). Рандомизированный метод Монте-Карло используется на шаге стохастического

семплирования алгоритма COM, и позволяет обойти ограничения по размеру вектора τ_i, описывающего условие разбиения некоторой выборки X на классы. Это достигается тем, что на каждой итерации происходит генерирование $N_{m+1} > 1$ реализаций τ_i. Для получения оценки для γ_i с помощью метода Монте-Карло, рассмотрим кластер K_i для каждой из реализаций y_r^{m+1} на $m+1$ итерации с использования текущего кластера K_{ir}^{m+1}. Общее число элементов из выборки X, входящих в кластер обозначим как ϑ_{ir}^{m+1}, где $r = 1,2, \dots, N_{m+1}$:

$$\gamma_i{}^{m+1} = \frac{1}{nN_{m+1}} \sum_{r=1}^{N_{m+1}} \vartheta_{ir}^{m+1}. \qquad (2.80)$$

Анализируя гибридный алгоритм видно, что он позволяет расширить вероятностный алгоритм ОМ за счет оптимизации процедуры генерации начального приближения параметров, принадлежащего некоторому кластеру K_i на основе метода Монте-Карло. Использования метода Монте-Карло позволяет повысить эффективность разбиения вектора τ_i на классы, что дает возможность наиболее четко сформировать кластер K_i.

2.3. Разработка технологий повышения эффективности вычислений для алгоритмов обучения структуры и параметров динамических байесовских сетей

Как показано в предыдущих параграфах, применение гибридных технологий обучения структуры и параметров сети позволяет получить достаточно качественные результаты. Адаптация данных подходов, применительно к байесовским сетям, обладающим достаточно большим числом переменных с соответствующим доменом значений, вызывает определенные сложности. Это обусловлено тем, что число условных связей между отдельными переменными возрастает и процедура построения сети требует вычисления тестов на условную независимость для каждой из переменных для оценки наличия связей между предполагаемыми дочерними и родительскими переменными. В процессе оптимизации алгоритмов обучения

структуры и параметров ДБС необходимо подобрать методы, позволяющие сократить вычисления и повысить скорость проведения расчетов.

Рассмотрим гибридный метод обучения динамических байесовских сетей, основанный на объединении вычисления статистических тестов на основе марковского покрытия и применения алгоритма имитации отжига. Математическая формулировка данного подхода была представлена выше. Раскроем процедуру оптимизации вычислений за счет распараллеливания отдельных алгоритмических блоков. Структура метода представляет собой комбинацию процедур построения исходной структуры сети и определения направленности связей между ее узлами. В основе разработки параллельного алгоритма на основе метода вычисления марковского покрытия с применением статистических оценок и алгоритма имитации отжига предполагается распараллеливание основных блоков. В процессе построения базовой структуры на основе МПСО предполагается декомпозиция единого процесса вычисления тестов на условную независимость и определения марковского покрытия для каждой из вершин на ряд подпроцессов, связанных с обработкой каждой вершины по отдельности в параллельном режиме с учетом всего объема обучающей выборки.

Для проведения сравнительного анализа выполняется оценка существующих гибридных методов обучения структуры БС с разработанным алгоритмом обучения МПСО, предполагается использовать несколько байесовских сетей с набором обучающих данных, применяемых для решения различных прикладных задач. Можно рассмотреть процедуру распараллеливания алгоритма МПСО. Для этого необходимо определить ряд критериев, позволяющих оценить эффективность определенного параллельного блока внутри алгоритма, а также задержку, связанную с синхронизацией данных, циркулирующих внутри каждого из параллельных блоков. Специфика любой параллельной системы, адаптированной для решения математических задач, определяет некоторые параметры, используемые в процедуре расчета и постановки заданий в очередь. Одним из таких параметром является общее число доступных параллельных процессоров в вычислительной системе. Оно

показывает максимальное число блоковых операций, которые может выполнять данная система единовременно.

В качестве целевых оценочных функций, применяемых в рамках алгоритма локального поиска на основе имитации отжига, используются следующие критерии оценки байесовских сетей: логарифм правдоподобия, байесовский информационный критерий (критерий Шварца, БИК), эквивалентный критерий Байеса-Дирихле (БДЭ), а также критерий Акаике (АИК). Байесовский информационный критерий и критерий Акаике формируются на основе логарифма правдоподобия, представленного ранее в формуле (2.33). Перепишем формулу логарифма правдоподобия [179] с учетом выражения (2.36)

$$L(G, \theta^G, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{i,j,k} ln \frac{N_{i,j,k}}{N_{i,j}}. \quad (2.81)$$

Обобщенную математическая формула для критериев Шварца и Акаике представим в виде следующего выражения:

$$Q(M) = L(G, \theta^G, D) - MF(N), \quad (2.82)$$

где N – размер выборки БС, M – общее число параметров БС Q^{G}.

$$M = \sum_{i=1}^{n} (r_i - 1)\, q_i. \quad (2.83)$$

С учетом выражений (2.81) и (2.83) равенство (2.82) приобретает следующий окончательный вид [41,167]:

$$Q(M) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{i,j,k} ln \frac{N_{i,j,k}}{N_{i,j}} - \sum_{i=1}^{n} (r_i - 1)\, q_i F(N). \quad (2.84)$$

Множитель $F(N)$ в выражении (2.84) является функцией штрафов, принимающей разные значения в зависимости от рассматриваемого критерия. Для критерия Акаике – $F(N) = 1$, критерия Шварца – $F(N) = \ln(N)/2$. Эквивалентный критерий Байеса-Дирихле определяется согласно выражению (2.17).

Одной из процедур оптимизации обучения структуры ДБС является использование метода Монте-Карло. Применение данной процедуры обусловлено необходимостью реализации обучения ДБС в условиях, когда имеется неполная

выборка D. Пуст X – множество переменных ДБС, x – множество значений переменной X, содержащихся в обучающей выборке $D = (X, E)$, E – множество наблюдаемых переменных. Тогда распределение вероятностей для структуры графа ДБС G, будет иметь следующий вид:

$$P(G|E) = \sum_X P(G|E,X)P(X|E), \tag{2.85}$$

где $P(H|E)$ – распределение по всем скрытым переменным.

Пусть G' – граф ДБС, скрытые переменные которого формируются с учетом распределения $P(X|E)$. Тогда выражение (2.85) приведем к следующему виду:

$$P(G|E,G') = \sum_X P(G|E,X,G')P(X|E,G'). \tag{2.86}$$

С учетом условной независимости графов G и G' при наличии переменных X и E, можно получить условие инвариантности для распределения $P(G|E,X,G')$

$$P(G|E,X,G') = P(G|E,X) \tag{2.87}$$

Сформулируем метод Монте-Карло для получения выборок на основе распределения (2.86)

$$P(G|E,G') \approx \frac{1}{N}\sum_{i=1}^{n} P(G|E,X_i) \tag{2.88}$$

В связи с тем, что расчет распределения $P(G|E,X_i)$ сложно вычислить в реальных условиях, определим соответствующее ему распределения по значимости $Q(G|E,X_i)$. Тогда выражение (2.86) перепишем с учетом введенного распределения

$$P(G|E,G') = \sum_X P(G|E,X)\frac{P(X|E,G')}{Q(G|E,X)}Q(G|E,X) \tag{2.89}$$

С учетом того, что формирование выборок производится из $X_i \sim Q(G|E,X)$

$$P(G|E,G') = \frac{1}{W}\sum_{i=1}^{N}\frac{P(X_i|E,G')}{Q(G|E,X_i)}P(G|E,X_i),$$
$$W = \sum_{i=1}^{N} W_i \tag{2.90}$$

Значение весов W_i, соответствующих распределению по значимости, можно определить в следующем виде:

$$W_i = \frac{P(X_i\,|E, G')}{Q(G|E, X_i)} \tag{2.91}$$

Для оценки правдоподобия структуры, формируемой по результатам применения обучения ДБС введем специальную метрику $S(G, D)$, зависящую от операций добавления, удаления, а также изменения направления ребер графа G

$$S(G, D) = \sum_{i=1}^{n} s(x_i|Y_i), \tag{2.92}$$

где $Y = Parents(x_i)$ – множество родительских вершин для x_i.

Тогда критерий эквивалентности двух графом G и G' будет определяться исходя из условия $S(G, D) = S(G', D)$. За основу оценки $P(G, D) = S(G, D)$ можно взять метрику БД (2.14), а в качестве оценки $\mathrm{argmax}\, P(G|D)$ методом максимума апостериорной вероятности – эквивалентную метрику БДЭ (2.17).

В процессе формирования оценки БДЭ требуется задать неравенство, позволяющее определить характер взаимной информации между переменными X_i, Y_i и Z, а также условную независимость вершин X_i, Y_i от переменной Z

$$BDe(X \cup Z)BDe(Y \cup Z) \geq BDe(X \cup Y \cup Z)BDe(Z). \tag{2.93}$$

Взаимная информация для X_i и Y_i будет иметь следующий вид:

$$I(X, Y) = \sum_{x}\sum_{y} P_{X,Y}(x, y)\, ln \frac{P_{X,Y}(x, y)}{P_X(x)P_Y(y)} \tag{2.94}$$

А условная взаимная информация

$$I(X, Y|Z) = \sum_{x}\sum_{y}\sum_{z} P_{X,Y,Z}(x, y, z)\, ln \frac{P_{X,Y,Z}(x, y, z)P_Z(z)}{P_{X,Y}(x, z)P_{Y,Z}(y, z)} \tag{2.95}$$

Из математического описания БДЭ (2.17) получим, что условие эквивалентности метрики достигается в том случае, если для обоих графов G и G' она будет иметь одинаковое значение. Данное условие выполнимо, когда оба графа содержат одинаковый набор эквивалентных классов N'. Произведем

доказательство условия эквивалентности для метрики БДЭ на основе ее правдоподобия. Определяем $L(X)$ следующего вида:

$$L(X) = \prod_k \frac{\Gamma(N'P(X = k|G) + N_k)}{\Gamma\big(N'P(X = k|G)\big)}, \tag{2.96}$$

где k – общее число допустимых состояний для X, $N_k \subset D$ число выборок, при которых $X = k$.

Расчет правдоподобия для каждой выборки D можно определить на основе следующего выражения:

$$P(D|G) = \prod_{i=1}^{n} L(X,Y) = \prod_{i=1}^{n} \frac{L(X_i \cup Y_i)}{L(Y_i)}. \tag{2.97}$$

Для обобщения оценки эквивалентности БС воспользуемся теоремой Чикерина [18]. Рассмотрим два графа G и G', а также множество ребер $A_{G,G'}$, характеризующее различия в направлении дуг для G и G'.

Оба графа G и G' эквиваленты, если имеет место быть некоторое множество $A'_{G,G'}$, состоящее из противоположно направленные дуг, после добавления которых к графу G получим G'. Из теоремы Чикерина следует, что $Y_j = X_i \cup Y_i$ – множество родительских вершин X_j для G и $Y_i = X_j \cup Y_j$ для G'. Если два графа G и G' отличаются лишь направленностью дуги $X_i \to X_j$, то можно определить правдоподобие каждой выборки для G в следующей форме:

$$L(X,Y) = \frac{L\big(X_i \cup Y_j\big)}{L\big(Y_j\big)} \frac{L\big(\{X_i, X_j\} \cup Y_j\big)}{L\big(X_i \cup Y_j\big)}. \tag{2.98}$$

Правдоподобие для графа G' [183] имеет следующий вид:

$$L'(X,Y) = \frac{L\big(X_j \cup Y_i\big)}{L(Y_i)} \frac{L\big(\{X_i, X_j\} \cup Y_i\big)}{L\big(X_j \cup Y_i\big)}. \tag{2.99}$$

Исходя из того, что родители $Y_i = Y_j$ для переменных X_i и X_j графов G и G' то $L(X,Y)$ и $L'(X,Y)$ будут эквивалентными.

Для оценки правдоподобия структуры данных сетей рассмотрим структурное расстояние Хэмминга (СРХ) и оценку метрик БДЭ на основе принципа максимальной энтропии (МПЭ). Под СРХ $H^*(G,G')$ будем понимать общее число

направленных ребер, на которые отличаются два графа G и G'. Отметим, что изменение значения CPX происходит при несоответствии направленности дуг графов и, если один из графов имеет ненаправленную дугу между узлами.

Воспользовавшись принципом МПЭ можно осуществить оценку правдоподобия структуры БС. Для этого зададим выражение для расчета энтропии Шеннона с учетом особенностей БС [148]

$$H(X|\theta) = \sum_{i=1}^{N} H\left(X_i\middle|\theta_{X_i}\right), \tag{2.100}$$

где $H\left(X_i\middle|\theta_{X_i}\right)$ – энтропия узлов X_i при заданном множестве родительских узлов $Y = Parents(X_i)$.

Тогда апостериорное значение математического ожидания для $H(X|\theta)$ при заданной выборке X имеет вид

$$\mathbb{E}(H(X_i)|D) = \int H\left(X_i\middle|\theta_{X_i}\right)P\left(\theta_{X_i}\middle|D\right)d\theta_{X_i}. \tag{2.101}$$

Распределение Дирихле $P\left(\theta_{X_i}\middle|D\right)$ запишем в следующем виде:

$$P\left(\theta_{X_i}\middle|D\right) = \int P\left(\theta_{X_i}\middle|D, \alpha_{i,j,k}\right)P\left(\alpha_{i,j,k}\middle|D\right)\, d\alpha_{i,j,k}. \tag{2.102}$$

С учетом метрики БД (2.14), запишем математическое ожидание (2.101)

$$\mathbb{E}(H(X_i)|D) = \mathbb{E}\left(H(X_i)\middle|D, \alpha_{i,j,k}\right)BD\left(X_i\middle|Y, \alpha_{i,j,k}\right). \tag{2.103}$$

Для вычисления множителя $\mathbb{E}\left(H(X_i)\middle|D, \alpha_{i,j,k}\right)$ воспользуемся свойством монотонности гамма-функции, приведенным Андерсоном [1]

$$\begin{aligned} &P\left(H(X_i)\middle|D, \alpha_{i,j,k}\right) \\ &= -\sum_{j=1}^{q}\sum_{k=1}^{r}\frac{a_{ijk}+N_{ijk}}{a_{ij}+N_{ij}}\ln\left(\frac{a_{ijk}+N_{ijk}}{a_{ij}+N_{ij}}\right) \\ &-\sum_{j=1}^{q}\frac{r-1}{2\left(a_{ij}+N_{ij}\right)} = H\left(X_i|D, \alpha_{i,j,k}\right) - \sum_{j=1}^{q}\frac{r-1}{2\left(a_{ij}+N_{ij}\right)}. \end{aligned} \tag{2.104}$$

Сформулируем МПЭ для двух графов G и G', отличающихся добавлением родительской вершины в граф G'

$$P(H(X)|D) \le P(H'(X)|D). \quad (2.105)$$

Сформулируем принцип МПЭ применительно к семантике БС на основе подхода, предложенного Сузуки [83]

$$H\left(X_i|Y, p_{i,j|k}\right) \le H\left(X_i|Y', p_{i,j|k}\right), Y' \subset Y'', \quad p_{i,j|k} = \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}, \quad (2.106)$$

$$BD(X_i|Y, \alpha_i) \ge BD(X_i|Y', \alpha_i). \quad (2.107)$$

С ростом числа размерности выборки D получаем сходимость $p_{i,j|k}$ к α_{ijk}. Следовательно, выражение (2.106) асимптотически будет эквивалентно следующему выражению:

$$H\left(X_i|Y, \alpha_{ijk}\right) \le H\left(X_i|Y', \alpha_{ijk}\right), Y \subset Y'. \quad (2.108)$$

Из неравенств (2.107) и (2.108) получим обратную зависимость между энтропией и метрикой БД.

Принцип МПЭ для БДЭ определим с учетом (2.106) и (2.108) за счет задания параметра $\alpha_{ijk} = \alpha_i/rq$

$$\mathbb{E}\left(\mathrm{H}'(\mathrm{X_i})\middle|\mathrm{D}, \alpha_{i,j,k}\right) BDe(X_i|Y', \alpha_i) \le \mathbb{E}\left(\mathrm{H}''(\mathrm{X_i})\middle|\mathrm{D}, \alpha_{i,j,k}\right) BDe(X_i|Y'', \alpha_i). \quad (2.109)$$

Для случая обучения БС в условиях неполных данных D, области выборок для каждого из графов G и G' могут отличаться. Оценка БДЭ для переменных X_i двух графов G и G' будет принимать разные значения, так как параметры родительских вершин Y' для X_i будут не определены [163]. Тогда выражение (2.109) может быть переписано за счет уточнения параметра α_i

$$\mathbb{E}\left(\mathrm{H}'(\mathrm{X_i})\middle|\mathrm{D}, \alpha_{i,j,k}\right) BDe(X_i|Y', \alpha_i) \le \mathbb{E}\left(\mathrm{H}''(\mathrm{X_i})\middle|\mathrm{D}, \alpha_{i,j,k}\right) BDe\left(X_i|Y'', \alpha_i\left(\frac{\tilde{q}}{q}\right)\right), \quad (2.110)$$

где $\tilde{q}$ – число возможных состояний, для которых $N_{i,j,k} > 0$.

Далее рассмотрим процедуру оптимизации похода к обучению параметров ДБС на основе ОМ и метода Монте-Карло с применением цепей Маркова (МКМЦ). Необходимость применения МКМЦ связана тем, что достаточно сложно

применить классический алгоритм ОМ для построения рандомизированных выборок многомерных функций распределений, так как генерация выборок происходит независимо и без учета каких-либо предшествующих состояний. Использование цепей Маркова обусловлено наличием транзитивных связей между смежными временными срезами ДБС. В основе метода Монте-Карло лежит порождение случайных стохастических переменных в соответствии с законом распределения, после чего требуется произвести их усреднение. Усреднение в методе Монте-Карло можно записать в виде следующего выражения:

$$\mathbb{E}f(x) = \int_0^1 f(x)\psi(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i), \tag{2.111}$$

где $\varphi(x)$ – плотность распределения случайной величины x.

Применительно к оценке максимального правдоподобия, используемой в алгоритме ОМ, выражение (2.111) может быть преобразовано к следующему обобщенному виду:

$$P(X, X') = \frac{1}{N}\sum_{i=1}^{N} ln\, P(X, X_i'), \tag{2.112}$$

где $P(X, X')$ – распределение по значимости, сходящееся к искомому распределению $P(X)$ в силу закона больших чисел (ЗБЧ) при достаточно больших значениях N.

Рассматривая сущность МКМЦ необходимо определить переходную вероятность $P(X_t|X_{t+1})$ из состояния X_t в X_{t+1}. Данная вероятность и будет определять вероятностную модель цепи Маркова. Процесс вычисления апостериорных вероятностей методом МКМЦ [74,115] происходит путем развертывания цепи Маркова на n шагов, описывающих состояния цепи. Вероятность нахождения системы в состояниях X_t и X_{t+1} в моменты t и $t+1$ будет соответственно $\mathrm{P}(X_t)$ и $P(X_{t+1})$. Исходя из этого, вероятность пребывания системы в состоянии X_{t+1} при заданном значении вероятности $P(X_t)$ может быть найдена путем суммирования всех возможных состояний, допустимых для временного

среза t. Вероятность $P(X_{t+1})$ может быть получена на основе следующего выражения:

$$P(X_{t+1}) = \sum_{X_t} P(X_t)P(X_{t+1}|X_t). \tag{2.113}$$

Стационарная (инвариантная) цепь, соответствующая выражению (2.113) может быть представлена следующим выражением:

$$P(X_t) = \sum_{X_{t+1}} P(X_{t+1})P(X_t|X_{t+1}), \forall X_t \in X. \tag{2.114}$$

Достаточным условием того, что сеть является стационарной, а вероятность $P(X_t)$ определяется согласно выражению (2.114) является то, что для распределений вероятностей $P(X_t|X_{t+1})$ и $P(X)$ выполнялся принцип детального равновесия

$$P(X_t)P(X_t|X_{t+1}) = P(X_{t+1})P(X_{t+1}|X_t). \tag{2.115}$$

В таком случае, используя принцип детального равновесия из выражения (2.115) достаточно доказать, что распределение $P(X)$ будет инвариантным

$$\sum_{X_{t+1}} P(X_{t+1})P(X_t|X_{t+1}) = \\ = \sum_{X_{t+1}} P(X_t)P(X_t|X_{t+1}) = P(X_t)\sum_{X_{t+1}} P(X_t|X_{t+1}) = P(X_t). \tag{2.116}$$

В качестве алгоритма рандомизации по методу Монте-Карло предполагается использовать алгоритм генерации выборок по схеме Гиббса. В общем представлении алгоритм Гиббса является частным случаем другого алгоритма – Метрополиса-Гастингса (МГ). Алгоритм Метрополиса-Гастингса используется для вычисления состояния X_{t+1}. Вначале выполняется формирование выборки X' с учетом распределения по значимости $Q(X, X')$, $\sum_{X'} Q(X, X') = 1$, сходящегося к $P(X)$. Распределение вероятностей $Q(X, X')$ в данном случае используется для генераций значений Y, когда система находится в состоянии X.

Рассматриваемое распределение по значимости $Q(X, X')$ в идеальных условиях должно быть симметричным, тогда имеем следующее неравенство:

$$P(X)Q(X, X') > P(X')Q(X', X). \tag{2.117}$$

Исходя из неравенства (2.117) следует, что процесс перехода из X в X' происходит часто, а из X' в X редко. На основании этого предположения Метрополисом было предложено увеличить число переходов из состояния из X в X' за счет определения вероятности $\varphi(X, X') < 1$ для каждого из рассматриваемых переходов.

В таком случае выражение, соответствующее вероятности перехода для алгоритма МГ имеет вид

$$P_M = Q(X, X')\varphi(X, X'). \tag{2.118}$$

Из свойства равенства вероятностей для P_M получим выражение для расчета вероятности $\varphi(X, X')$

$$\varphi(X, Y) = \frac{P(X')Q(X', X)}{P(X)Q(X, X')}. \tag{2.119}$$

Из выражения (2.119) получим, что при $\varphi(X, X') \geq 1$ переменная $X' = X_{i+1}$, с вероятностью $\varphi(X, X')$ и $X = X_{i+1}$ с вероятностью $1 - \varphi(X, Y)$.

Выражение (2.119) было обобщено Гастингсом и говорит о том, что аппроксимирующее распределение $Q(X, X')$ не всегда должно быть симметричным. В свою очередь, если правило симметричности выполняется, уравнение (2.119) может быть преобразовано к виду, предложенному Метрополисом $\varphi(X, X') = P(X')/P(X)$. Марковская цепь, соответствующая алгоритму МГ [53], может быть также получена из выражения (2.119)

$$M = P(X, X') = min\left[1, \frac{P(X')Q(X', X)}{P(X)Q(X, X')}\right], X \neq X'. \tag{2.120}$$

Метод выборки по Гиббсу предназначен для оптимизации генерации выборок для многомерных распределений. Основная идея алгоритма базируется на упрощении операции порождения выборок. Это достигается тем, что генерация выборок на каждой итерации производится только для одной рассматриваемой переменной с однородным распределением, остальные же фиксируются. Тогда процесс генерации каждой следующей выборки X^{t+1} будет определяться следующим образом [69]:

$$X_1^{t+1} \sim P\left(X_1 \middle| X_2^i, X_3^i, \dots, X_m^i\right), \tag{2.121}$$

$$X_2^{i+1} \sim P\left(X_2 \middle| X_1^i, X_3^i, \dots, X_m^i\right),$$

$$\dots$$

$$X_m^{i+1} \sim P\left(X_m \middle| X_1^i, X_2^i, \dots, X_{m-1}^i\right).$$

Порядок генерации выборок для переменных X_j может быть произвольным, однако не должен изменяться на разных итерациях. Данное правило позволяет соблюдать условие однородности марковской цепи для каждой из рассматриваемых переменных, в противном случае возникает необходимость приведения данной сети к эргодической, а именно доказать сходимость к единственному распределению. Рассматривая двумерное распределение $P(X, X')$, сущность алгоритма может быть представлена в виде совокупности следующих основных шагов [178].

На первом шаге выполняется определение начального распределения для всех переменных распределения, а именно X_0.

На следующем этапе в соответствии с механизмом генерации выборок (2.121) получаем значение X'_{i+1} из распределения $P(X'|X_i)$, а затем X_{i+1} на основе $P(X|X')$. Далее повторяем второй этап до тех пока марковская цепь не достигнет своего стационарного состояния.

Из описания алгоритма Гиббса видно, что он является частным случаем алгоритма МГ, если заменить распределение по значимости $Q(X', X)$ на $P(X'_i|X'_{i-1})$. Одним из ограничений, накладываемым алгоритмом Гиббса, является то, что для его выполнения необходимо знать распределение вероятностей $P(X_i|X_1, X_2, \dots, X_{i-1}, X_{i+1}, \dots, X_m)$. Однако в рамках обучения параметров ДБС данное распределение несложно получить, так как уже известна топология сети, а также определен набор ее параметров из обучающей выборки. В связи с этим схема Гиббса является достаточно простой, эффективной и может быть легко адаптирована к шагу ожидания алгоритма ОМ, так как не требуется дополнительных знаний относительно ДБС. В данном случае достаточно лишь иметь структуру сети для определения вероятностных связей и необходимый объем обучающей выборки для получения наиболее точных генераций в процессе выполнения алгоритма.

С учетом введенных замечаний, можно определить погрешность для обоих алгоритмов – Гиббса и МГ на этапе ожидания алгоритма ОМ. Данная погрешность может быть установлена на основе следующего выражения:

$$\varepsilon(N) = \sum_{X'} ln\, P(X, X')\, P(X'|X) - Q(X, X'). \tag{2.122}$$

Одним из критериев снижения погрешности $\varepsilon(N)$ является увеличение значения N. Между тем число N накладывает ресурсные ограничения. В связи с чем, оптимальным подходом для достижения приемлемого уровня погрешности $\varepsilon(N)$ является использование небольших выборок на начальном этапе алгоритма, в то время как на поздних стадиях алгоритма, необходимо постепенное увеличения размера выборки для достижения приемлемого значения погрешности $\varepsilon(N)$.

Далее рассмотрим сходимость алгоритма выборок Гиббса. В результате выполнения алгоритма получаем набор распределений для переменных X_k, где $k = 1,2,\dots,N$. Если общее число генераций $N \to \infty$, то данные распределения по значимости $Q(X, X')$ будут эквивалентны искомому распределению $P(X|X')$. В практической области применения алгоритма Гиббса для достижения приемлемого уровня точности предполагается, что число формируемых генераций N должно быть достаточно большим. При этом полагается, что M выборок $M < N$ должно быть использовано в качестве тестовых испытаний с целью приближения выбранных значений к целевому переходному распределению вероятностей $P(X|X')$ для всех срезов ДБС.

2.4. Выводы

Во второй главе работы:

1. Проведен анализ основных подходов к обучению динамических байесовских сетей, реализованных на основе методов статистического анализа, бустинга и семплирования по методу Монте-Карло.

2. Описан предложенный в работе метод обучения структуры динамических байесовских сетей на основе вычисления марковского покрытия, учитывающий при

обучении транзитные связи между соседними временными срезами сетей, использующий гибридных технологий обучения, базирующиеся на статистических подходах оценки топологии байесовских сетей и применении оптимизационной процедуры имитация отжига для определения направленности связей между отдельными узлами сетей.

3. Описан предложенный в работе метод обучения вероятностных параметров динамических байесовских сетей, адаптированный к условиям неполных данных, реализованный на основе применения стохастического алгоритма ожидания максимизации и семплирования по методу Монте-Карло с применением цепей Маркова.

4. Реализована методика оценки адекватности структуры обученной динамической байесовской сети на основе структурной энтропии, расстояния Хэмминга и принципа максимума перекрестной энтропии.

Глава 3. Разработка гибридных методов вероятностного вывода на основе взвешивания с учетом правдоподобия

3.1. Вероятностный вывод в статических и динамических байесовских сетях

Для проведения вероятностного вывода все переменные сети делятся на две категории: переменные запроса и свидетельства. Основная сущность вероятностного вывода заключается в формировании апостериорного распределения для всех возможных переменных запроса $X = \{X_1, X_2, \dots, X_n\}$ при наступлении события e, для которого множество свидетельств $E = \{E_1, E_2, \dots, E_n\}$ принимают значения $E_1 = e_1, E_2 = e_2, \dots, E_n = e_n$. Совокупность всех вершин принято обозначать в виде объединения $X \cup E \cup Z$, где Z – ненаблюдаемые (скрытые) переменные сети. Сформулируем основные задачи, решаемые с помощью вероятностного вывода [197].

Фильтрация. Решение данной задачи заключается в определении апостериорных вероятностей для всех переменных $P(X_{t+1}|E_{1:t+1})$ при получении свидетельств $E_{1:t}$ от момента $t = 1$ до рассматриваемого момента времени $t = k$. Такое распределение может быть получено с учетом уравнения Чепмена-Колмогорова, определяющего распределение вероятностей переменных X_{t+1} при наличии всех свидетельств $E_{1:t}$

$$P(X_{t+1}|E_{1:t}) = \sum_{X_t} P(X_{t+1}|X_t)\, P(X_t|E_{1:t}). \tag{3.1}$$

С учетом выражения (3.1) процесс фильтрации можно представить в виде распределения вероятностей

$$P(X_{t+1}|E_{1:t+1}) = P(E_{t+1}|X_{t+1})\left(\sum_{X_t} P(X_{t+1}|X_t)\, P(X_t|E_{1:t})\right) \tag{3.2}$$

где $P(X_{t+1}|X_t)$ – модель перехода, $P(E_{t+1}|X_{t+1})$ – модель восприятия.

Предсказание. Задача определения апостериорного распределения вероятностей $P(X_{t+k}|E_{1:t})$ переменных в будущем состоянии $t + k$ при получении

свидетельств к заданному моменту времени. Общее решение задачи предсказания достигается за счет пошагового вычисления распределения вероятностей $P(X_{t+k+1}|E_{1:t})$ для среза $t+k+1$, основываясь на вероятности предсказания $P(X_{t+k}|E_{1:t})$

$$P(X_{t+k+1}|E_{1:t}) = \sum_{X_{t+k}} P(X_{t+k+1}|X_{t+k})P(X_{t+k}|E_{1:t}). \tag{3.3}$$

В общем случае задача предсказания может быть представлена в виде фильтрации без модели восприятия. Данное утверждение имеет смысл, так как фильтрация уже содержит в себе единичное предсказание.

Сглаживание. Решение задачи предполагает вычисление апостериорных вероятностей $P(X_k|E_{1:t}), 1 \le k \le t$ предыдущих состояний при поступлении свидетельств до текущему момента времени

$$P(X_k|E_{1:t}) = P(X_k|X_{1:k})P(E_{k+1:t}|X_k). \tag{3.4}$$

Анализируя выражение (3.4) видно, что распределение для первого множителя $P(X_k|X_{1:k})$ может быть получено путем фильтрации (3.2) в прямом направлении на интервале $(1,k)$. Распределение для второго множителя $P(E_{k+1:t}|X_k)$ может быть получено исходя из свойства условной независимости свидетельств E_{k+1} и E_{k+2}, если задана переменная X_{k+1}

$$\begin{aligned} P(E_{k+1:t}|X_k) &= \sum_{X_{k+1}} P(E_{k+1:t}|X_k, X_{k+1})\, P(X_{k+1}|X_k) \\ &= \sum_{X_{k+1}} P(E_{k+1}|X_{k+1})P(E_{k+2:t}|X_{k+1})P(X_{k+1}|X_k) \end{aligned} \tag{3.5}$$

Задача поиска наиболее вероятной последовательности (задача декодирования Витерби). Задача декодирования Витерби сводится к определению наиболее вероятной последовательности переменных запроса. Каждая последовательности может быть представлена в виде графа, в котором значение правдоподобия для любого пути из одной вершины в другую представляет собой произведение вероятности перехода для данного пути на вероятность появления свидетельств для каждого из состояний рассматриваемого пути. Другими словами,

существует связь между наиболее вероятными путями для X_t и X_{t+1}. Данную связь можно представить в следующем виде:

$$\max_{X_1,X_2,\dots,X_t} P(X_1, X_2, \dots, X_t, X_{t+1}|E_{1:t+1}) = P(E_{t+1}|X_{t+1}) \max_{X_t} P(X_{t+1}|X_t) \times \max_{X_1,X_2,\dots,X_{t-1}} P(X_1, X_2, \dots, X_{t-1}, X_t|E_{1:t}). \quad (3.6)$$

Уравнение (3.6) по своему виду идентично выражению для фильтрации (3.2). Отличительной особенностью является замена множителя $P(X_t|E_{1:t})$ вероятностями пути, связанными с состоянием X_t, а именно $m_{1:t} = \max_{X_1,X_2,\dots,X_{t-1}} P\left(X_{1,X_2,\dots,X_{t-1}}|E_{1:t}\right)$. А знак суммы в выражении (3.2) заменяется максимизацией.

В некоторых случаях представление модели перехода и восприятия удобно записать в более сжатой матричной форме. Если переменная X принимает некоторый домен значений $D = D_1, D_2, \dots, D_n$, то модель перехода $\mathrm{P}(X_{t+1}|X_t)$ может быть представлена в виде квадратной матрицы переходов T размерностью $n \times n$, имеющей следующий вид:

$$T = P(X_{t+1}|X_t) = \begin{pmatrix} T_{11} & T_{12} & \cdots & T_{1n} \\ T_{21} & T_{22} & \cdots & T_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n1} & T_{n2} & \cdots & T_{nn} \end{pmatrix}, \quad (3.7)$$

$$T_{ij} = P(X_{t+1} = j|X_t = i)$$

По аналогии можно описать модель восприятия $P(\mathrm{E_{t+1}}|X_{t+1})$, полагая, что свидетельство E_{t+1} принимает значение e_{t+1}. В связи с этим в процессе формирования матрицы восприятия, необходимо учитывать лишь ту часть выборок, которая согласуется со свидетельством e_{t+1}, а именно $P(e_{t+1}|X_{t+1})$

$$S = P(e_{t+1}|X_{t+1}) = \begin{pmatrix} S_{11} & 0 & \cdots & 0 \\ 0 & S_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S_{mm} \end{pmatrix} = diag\{S_{ij}\}. \quad (3.8)$$

Матрицы перехода и восприятия T и S могут быть использованы в процессе решения задач вероятностного вывода ДБС, а также позволяют применять подходы, направленные на оптимизацию алгоритмов фильтрации и сглаживания.

Для решения описанных выше задач можно использовать точные алгоритмы вероятностного вывода. Применение точных алгоритмов основывается на развертывании динамической байесовской сети путем повторения временных срезов до заданного момента времени. Одним из алгоритмов, реализующих данный подход, является алгоритм прямого распространения. Данный алгоритм может быть применен к развернутой байесовской сети, в том случае, если возможно преобразование ее в скрытую Марковскую модель (СММ). СММ состоит из множества наблюдаемых переменных $X = \{X_1, X_2, \dots, X_t\}$ и переменных состояний $Y = \{Y_1, Y_2, \dots, Y_t\}$. При этом каждое значение переменной состояния Y_{t+k} образуют марковскую цепь первого порядка. Распределение вероятностей для переменных наблюдения имеет следующий вид [160]:

$$P(X_{t+k}) = \sum_{Y_{t+k}} P(X_{t+k}|Y_{t+k})\, P(Y_{t+k}). \tag{3.9}$$

В основе алгоритма прямого распространения используется двухшаговый подход. На первом шаге происходит рекурсивное вычисление распределения вероятностей по всем переменным X_t при наличии всех свидетельств вплоть до текущего момента времени $P(X_t|E_{1:t})$. На втором шаге вычисляется распределение вероятностей $P(E_{t+1:t}|X_t)$. В результате получаем распределения вероятностей по всем переменным X_{t+1}

$$P(X_{t+1}|E_{1:t+1}) = \frac{P(E_{t+1:t}|X_t)P(X_t|E_{1:t})}{P(E_{1:t})}. \tag{3.10}$$

Для алгоритма прямого распространения временная сложность пропорциональна числу состояний X_k.

Другим алгоритмом точного вероятностного вывода является граничный алгоритм. Реализация данного алгоритма основывается на распространение марковского покрытия через срезы динамической байесовской сети. Все узлы, принадлежащие марковскому покрытию для момента t, можно объединить в граничное множество F. Обозначим узлы, находящиеся справа и слева, как R и L соответственно. При этом на каждом шаге алгоритма выполняется условие, что

переменная из R условно независима от L при наличии F. Структурно алгоритм можно представить в виде двух основных этапов.

На первом этапе введем следующие обозначения: Y_f – множество скрытых переменных для F, E_f – переменные-свидетельства для F, E_l – свидетельства для L, E_r – свидетельства для R. Вычисляем распределение вероятностей $P(Y_f) = P(Y_f|E_f, E_l)$ за счет добавления узла N к переменной F и перемещаемся в направлении от R к F. Как только все родительские узлы будут добавлены в граничное множество F, получим распределение вероятностей с учетом добавления вершины N

$$P(E_l, E_f, Y_f, N) = P(E_l, E_f, Y_f)P(N|E_f, Y_f). \quad (3.11)$$

Далее удаляем каждый узел N, перемещаясь в противоположном направлении от границы F к L. Тогда все дочерние переменные уже будут принадлежать граничному множеству узлов F. В том случае, если N является скрытой переменной, распределение вероятностей по всем свидетельствам может быть представлено в виде суммы по N

$$P(E_{l+N}, E_{f-N}, Y_{f-N}) = P(E_l, E_f, Y_{f-N}) = \sum_N P(E_l, E_f, Y_f, N), \quad (3.12)$$

где $l + N$ представляет объединение $L \cup \{N\}$ и $f - N$ дополнение граничного множества $F\{N\}$.

Если переменная N является наблюдаемой, процесс маргинализации можно опустить, а выражение (3.12) может быть приведено к следующему виду:

$$P(E_{l+N}, E_{f-N}, Y_{f-N}) = P(E_l, E_f, Y_f). \quad (3.13)$$

На следующем этапе происходит постановка задачи вычисления распределения вероятности $P(E_r|Y_f, E_f)$. После чего выполняется смещение границы F относительно среза t. Данное смещение происходит за счет добавления и удаления узлов в противоположном порядке относительно первого этапа алгоритма. Процедура добавления узла описывает перемещение от L к F, а удаление от F к R. В процессе добавления узла N предполагается вычисление вероятности $P(E_r|E_f, Y_f, N)$. Значение данной вероятности вытекает уз условия, что

на первом этапе алгоритма узел N уже бы удален. Следовательно, граничное множество F содержит все дочерние узлы, связанные с N, а вероятность равна $P(E_r|E_f, Y_f, N) = P(E_r|E_f, Y_f)$. Если вершина N является скрытым узлом, то распределение $P(E_{r+N}|E_{f-N}, Y_{f-N})$ может быть получено за счет его удаления из граничного множества F

$$P(E_{r+N}|E_{f-N}, Y_{f-N}) = \sum_N P(N|E_f, Y_{f-N}) P(E_r|E_f, Y_f) \tag{3.14}$$

Если узел N является наблюдаемым, выражение (3.14) преобразуется в следующий вид [76]:

$$P(E_{r+N}|E_{f-N}, Y_{f-N}) = P(N|E_{f-N}, Y_f) P(E_r|E_N, E_{f-N}, Y_f). \tag{3.15}$$

Для расширения возможностей точного вероятностного вывода используются различные оптимизированные алгоритмы сглаживания и фильтрации. Алгоритмы фильтрации направлены на решение первоочередных задач вероятностного вывода, включая предсказание, так как в общем случае фильтрация включает в себя одношаговое предсказание. Для ДБС с непрерывными переменными и гауссовым распределением вероятностей наибольшее распространение получили алгоритмы на основе фильтра Калмана. При этом ДБС с непрерывными переменными также носит название линейной динамической системы (ЛДС). Классический (дискретный) фильтр Калмана использует гауссовы распределения для задания модели перехода ДБС из состояния X_t в X_{t+1} с добавлением шума, распределенного по нормальному закону. Для получения предсказания (3.3) фильтр Калмана накладывает следующее ограничение – прогнозируемые распределения вероятностей $P(X_{t+1}|E_{1:t})$ и $P(X_{t+1}|E_{1:t+1})$ должны быть гауссовыми. Уравнения для ЛДС, соответствующее ДБС с дискретным временем можно записать в следующем виде [29,31]:

$$X_{t+1} = F_{t,t+1} X_t + w_t,$$
$$\mathbb{E}[w_n, w_t^T] = \begin{cases} Q_t, n = t \\ 0, n \neq t \end{cases}, \tag{3.16}$$

$$E_{t+1} = H_{t+1} X_{t+1} + v_{t+1}, \tag{3.17}$$

$$\mathbb{E}[v_n, v_{t+1}^T] = \begin{cases} R_{t+1}, n = t+1 \\ 0, n \neq t+1 \end{cases},$$

где $F_{\mathrm{t,t+1}}$– матрица модели перехода, принимающая состояние $X_{\mathrm{t+1}}$ при переходе между временными срезами t и $t+1$, $H_{\mathrm{t+1}}$ – матрица модели состояния, $w_t, v_{\mathrm{t+1}}$ – шумовые нормальные процессы с ковариационными матрицами Q_t и R_{t+1} соответственно. При этом шумовой процесс $v_{\mathrm{t+1}}$ не коррелирован с процессом w_t.

Задача фильтрации Калмана может быть сформулирована следующим образов: для каждого состояния $X_{\mathrm{t+1}}$, при условии $t \geq 1$, необходимо получить оценку с наименьшей среднеквадратической ошибкой в том случае, если даны все свидетельства $E_{\mathrm{t+1}}$. Для решения этой задачи необходимо определить распределение вероятностей для моделей перехода и восприятия. Данные распределения вероятностей $P(\mathrm{X_{t+1}}|X_t)$ и $P(\mathrm{E_{t+1}}|X_{\mathrm{t+1}})$ имеют следующий вид:

$$\begin{aligned} P(X_{t+1}|X_t) &= N\big(X_{t+1}\big|F_{t+1,t}X_t, Q_t\big), \\ P(E_{t+1}|X_{t+1}) &= N(E_{t+1}|H_{t+1}X_{t+1}, R_{t+1}). \end{aligned} \tag{3.18}$$

Выражение оптимальной фильтрации, соответствующее уравнению ДБС (3.18), имеет следующий вид [218]:

$$\begin{aligned} P(X_{t+1}|E_{1:t}) &= N\big(X_{t+1}\big|\tilde{X}_{t+1}^-, \Phi_{t+1}^-\big), \\ P(X_{t+1}|E_{1:t+1}) &= N\big(X_{t+1}\big|\tilde{X}_t, \Phi_t\big), \\ P(E_{t+1}|E_{1:t}) &= N(E_{t+1}|H_{t+1}\tilde{X}_{t+1}, S_{t+1}), \end{aligned} \tag{3.19}$$

где $S_{\mathrm{t+1}} = H_{t+1}\Phi_{\mathrm{t+1}}^- H_{t+1}^T + R_{t+1}$, $\Phi_t, \Phi_{\mathrm{t+1}}^-$ – матрицы ковариаций для моментов времени t и $t+1$, включающие в себя оценки дисперсии погрешности (ошибки) для каждого из состояний ДБС.

Распределения вероятностей, описанные в выражении (3.19) можно получить с использованием фильтра Калмана. Данный фильтр состоит из двух основных этапов: предсказание и обновление [201].

На этапе предсказания получаем следующие выражения, определяющие оценку и ошибку предсказания для момента времени $t+1$ с учетом начального состояния параметра X_0:

$$\tilde{X}_0 = \mathbb{E}[X_0], \tag{3.20}$$

$$P_0 = \mathbb{E}[(X_0 - \mathbb{E}[X_0])(X_0 - \mathbb{E}[X_0]^T)],$$
$$\tilde{X}_{t+1}^- = F_{t+1}\,\tilde{X}_t,$$
$$\Phi_{t+1}^- = F_{t,t+1}\mathbb{E}[\tilde{X}_t\tilde{X}_t^T]F_{t,t+1}^T + \mathbb{E}[w_n, w_t^T],$$
$$= F_{t,t+1}\Phi_t F_t^T + Q_t,$$

где $\tilde{X}_0$ оценка параметра X_0, соответствующая начальному состоянию сети (момент времени $t = 0$).

На этапе обновления получаем значение усиления Калмана G_{t+1} и ошибку ковариации Φ_{t+1} [36,130]:

$$K_{t+1} = F_t\,\tilde{X}_t,$$
$$G_{t+1} = \Phi_{t+1}^- H_{t+1} {S_{t+1}}^{-1}, \quad (3.21)$$
$$\Phi_{t+1} = (I - G_{t+1}H_{t+1})\Phi_{t+1}^-. \quad (3.22)$$

Из выражений (3.20) и (3.21), описывающих фильтр Калмана, можно получить полное совместное распределение $P(X_t, X_{t+1}|E_{1:t})$ по всем переменным наблюдения

$$P(X_t, X_{t+1}|E_{1:t}) = P(X_{t+1}|X_t)P(X_t|E_{1:t}) = N(X_{t+1}|F_tX_t, Q_t)N\left(X_t\middle|\tilde{X}_t, B_t\right),$$
$$P(X_{t+1}, E_{t+1}|E_{1:t}) = N(E_{t+1}|X_{t+1})N(X_{t+1}|E_{1:t}), \quad (3.23)$$
$$= N(E_{t+1}|H_{t+1}X_{t+1}, R_{t+1})N\left(X_k\middle|\tilde{X}_{t+1}^- B_{t+1}^-\right).$$

При этом ковариационные матрицы Q_t и R_t могут быть представлены в виде единичной матрицы $Q_t = I, R_t = I$. Данное равенство может быть получено за счет вычисления квадратного корня соответствующих матриц. Исходя из этого, можно определить матрицы Q_t и R_t в виде следующего выражения [100]:

$$Q_t = S_t(S_t)^T,$$
$$R_t = S_t'(S_t')^T, \quad (3.24)$$

где S_t и S_t' – квадратные корни для матриц Q_t и R_t соответственно.

Фильтр Калмана является хорошо апробированным инструментом, однако требует применение дополнительных численных методов оптимизации расчета ковариационных матриц. В первую очередь, такая необходимость связана с тем, что большая часть времени затрачивается на решение разностного уравнения Риккати (3.22) для ковариации Φ_{t+1}. Для решения задач численной оптимизации

наибольший интерес представляют квадратно-корневые алгоритмы фильтрации на основе разложения Холецкого и ортогонализации Грамма-Шмидта. Рассмотрим каждый из данных алгоритмов более детально. В основе метода Холецкого лежит представление положительно определенной матрицы A в виде произведения следующего вида:

$$H = LL^T, (Hx, x) \geq 0, \tag{3.25}$$

где L – нижняя треугольная матрица с положительно определенными элементами, стоящими на главной диагонали и являющиеся коэффициентами Холецкого для матрицы H. Процедура вычисления значений для элементов, входящих в состав матрицы H, имеет следующий вид:

$$H_{ij} = \sum_{k=1}^{n} L_{ik} L_{kj}^T, (Hx, x) \geq 0. \tag{3.26}$$

Из условия симметричности матрицы H получим частный случай, когда $i \leq j$. Преобразуя выражение (3.26), получим

$$H_{ij} = \sum_{k=1}^{i-1} L_{ik} L_{kj}^T + L_{ii} L_{ij}. \tag{3.27}$$

Из выражения (3.27) следует, что значения элементов матрицы L вычисляются следующим образом:

$$\begin{gathered} L_{ii} = \sqrt{H_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}, i = j, i = 1,2, \dots, n, \\ L_{ji} = \frac{\left(H_{ii} - \sum_{k=1}^{i-1} L_{ik} L_{jk}\right)}{L_{ii}}, i < j, j = i + 1, i + 2, .. n. \end{gathered} \tag{3.28}$$

Применение метода Холецкого в фильтрации Калмана [29] направлено на разложение ковариационных матриц Q_t и R_{t+1}, характеризующих шумовые процессы, протекающие внутри фильтра. В тоже время использование данного метода позволяет упростить процедуру вычисления значения усиления Калмана (3.21). Определим алгоритм вычисления симметричной, положительно определенной матрицы L с применением разложения Холецкого. Алгоритм

представляет собой совокупность выражений, вычисляемых циклически [123] для каждого из элементов матрицы L

$$\begin{gathered} H_{ik} = H_{ik} - L_{ik}L_{jk}, \\ L_{ij} = \sqrt{H_{jj}}, i = j, \\ L_{ij} = \frac{H_{k,j}}{L_{j,j}}, i \neq j, \end{gathered} \tag{3.29}$$

где $j = 1,2,..,n; i = 1,2,..,j; k = 1,2,\dots,j-1$ для нижнего треугольного разложения и $j = m, m-1,\dots,1; i = j, j-1,..,1; k = j+1,\dots,m$ для верхнего треугольного разложения.

В свою очередь для оптимизации процедуры разложения матриц можно использовать оптимизированный метод Холецкого (UD-алгоритм). Сущность данного подхода заключается в формировании нижнего (верхнего) треугольного разложения для матрицы H. При этом подразумевается, что единичной верхней (нижней) треугольной будет считаться матрица, содержащая единицы на своей главное диагонали. Исходя из этого, разложение Холецкого (3.25) может быть приведено к следующему виду:

$$H = \overline{L}D\overline{L}^{T} = UDU^{T}, \tag{3.30}$$

где $\overline{L}$ – нижняя единичная треугольная матрица, U – верхняя единичная треугольная матрица, D – главная диагональ.

Тогда алгоритм разложения Холецкого может быть преобразован к UD-алгоритму с учетом формулировки (3.30)

$$\begin{gathered} H_{ik} = H_{ik} - U_{ik}D_{kk}U_{jk}, \\ D_{jj} = H_{ij}U_{jj} = 1;\ i = j \\ U_{ij} = \frac{H_{ij}}{D_{jj}}; i \neq j, \end{gathered} \tag{3.31}$$

где $j = n, n-1,\dots,1; i = j, j-1,\dots,1; k = j+1, j+2,\dots,n.$

Основное отличие UD-разложения от классического разложения Холецкого заключается в упрощения процедуры вычисления значений элементов матрицы и

отсутствие операции вычисления квадратного корня, что также позволяет сократить время выполнения алгоритма.

Далее рассмотрим алгоритмы на основе ортогонализации Грамма-Шмидта. Матрица H будет являться ортогональной, если выполняется следующее условие:

$$\begin{gathered} HH^T = I, \\ H^{-1} = H^T. \end{gathered} \tag{3.32}$$

При этом получаем, что матрица H^T будет обратной к матрице H.

Рассмотрим процесс формирования ортогональной матрицы на основе метода Грамма-Шмидта. Сущность данного метода заключается в нахождении множества взаимно ортогональных векторов $q = (q_1, q_2, \dots, q_m)$ из подпространства $\Omega \in \mathbb{R}$, исходя из линейно-независимого множества векторов $h' = (h_1, h_2, \dots, h_n)$, h' – i-й столбец матрицы H

$$\begin{gathered} q_i^T q_i = \|q_i\|^2 = 1, i = j, q \in \mathbb{R}_i, \\ q_i^T q_j = 0, i \neq j. \end{gathered} \tag{3.33}$$

Каждый вектор $h_i \in H$ может быть представлен в виде суммы произведений векторов q_j и b_{ij}

$$h_i = \sum_{j=1}^{n} q_j b_{ij} \tag{3.34}$$

В том случае, если матрица Q состоит из набора ортонормированных векторов q_j, матрица H может быть представлена в виде произведения

$$H = QL = [q_1, q_2, \dots, q_n] \begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n2} & 1 \end{pmatrix}, \tag{3.35}$$

где L – матрица перехода от базиса $q = (q_1, q_2, \dots, q_n)$ к базису $h' = (h_1, h_2, \dots, h_n)$, Q – ортогональная матрица

$$\begin{gathered} Q^T Q = (q_1, q_2, \dots, q_n) \begin{pmatrix} \|q_1\|^2 & 0 & \dots & 0 \\ 0 & \|q_2\|^2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \|q_n\|^2 \end{pmatrix} = diag\{\|q_i\|^2\}, \\ i = 1,2, \dots, n. \end{gathered} \tag{3.36}$$

Выражение ортогонализации Грамма-Шмидта, приведенное в формуле (3.35), носит также название QL-разложения. В свою очередь, если нижнюю треугольную матрицу L заменить на верхнюю треугольную матрицу R, выражение может быть преобразовано к QR-разложению. Одним из разновидностей QR-разложений является метод Хаусхолдера. Для формулировки алгоритма Хаусхолдера необходимо определить фиксированный вектор w, взятый из пространства $\Omega \in \mathbb{R}$ и для которого справедливо следующее равенство:

$$\|w\|^2 = \sqrt{(w,w)} = \sqrt{w^T w} = 1. \tag{3.37}$$

Основой данного подхода является приведение исходной квадратной матрицы H размерностью $n \times n$ к трехдиаганальной форме за $n-2$ шагов. Матрица, образованная из вектора w, называется матрицей Хаусхолдера и имеет следующий вид [123]:

$$D = I - 2ww^T, \tag{3.38}$$

где w – вектор Хаусхолдера.

В свою очередь докажем, что матрица D является ортогональной. Для этого определим произведение вектора w на его транспонированное представление w^T

$$ww^T = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} (w_1 \quad w_2 \quad \cdots \quad w_n) = \begin{pmatrix} w_1^2 & w_1 w_2 & \cdots & w_1 w_n \\ w_2 w_1 & w_2^2 & \dots & w_2 w_n \\ \cdots & \cdots & \cdots & \cdots \\ w_n w_1 & w_n w_2 & \dots & w_n^2 \end{pmatrix}. \tag{3.39}$$

Из выражения (3.39) следует симметричность матрицы ww^T и, как следствие, симметричность матрицы D. Исходя из этого, можно получить свойство ортогональности для матрицы D

$$DD^T = D^2 = I - 4ww^T + 4w(w^T w)w^T = E, (w^T w) = 1. \tag{3.40}$$

Рассмотрим преобразование матрицы H к трехдиаганальной форме на основе разложения Хаусхолдера, а также докажем, что данное разложение является разновидностью QR-разложения. Для этого построим отражение Хаусхолдера $D_1 = E - w_1 w_1^T$ и применим ее к матрице H

$$H_1 = D_1 H, \tag{3.41}$$

где H_1 представляет собой матрицу, в которой первый столбец заполнен нулями, за исключением первого диагонального элемента $h_{11}^{(1)}$.

Данная матрица имеет следующий вид:

$$H_1 = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ 0 & h_{22} & \dots & h_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & h_{n2} & \dots & h_{nn} \end{pmatrix}. \quad (3.42)$$

На следующем этапе вычислим значение матрицы H_2

$$H_2 = D_2 H_1 = D_2 D_1 H. \quad (3.43)$$

Из формул (3.42) и (3.43) следует, что для приведения матрицы H к треугольной форме необходимо выполнить $n-1$ итераций, где на каждой итерации необходимо вычислить отражение Хаусхолдера $D_i = E - w_i w_i^T, i = 1,2,\dots,n-1$. Тогда результирующая треугольная матрица D_{n-1} будет определяться на основе следующего выражения [114]:

$$R = D_{n-1} = D_{n-1}, D_{n-1}, \dots, D_2 D_1 H = Q^T H, \quad (3.44)$$

где $Q^T = D_{\mathrm{n}-1}, D_{\mathrm{n}-2}, \dots, D_2 D_1$ – матрица произведений ортогональных отражений Хаусхолдера, полученных за $n-1$ итераций. Принимая во внимание, что результат произведения ортогональных матриц Хаусхолдера является также ортогональная матрица, выражение (3.44) может быть преобразовано к следующему виду за счет умножения $(Q^T)^{-1} = (Q^T)^T = Q$. В результате получим искомое выражение факторизации для QR-разложения $H = QR$.

Использование рассмотренных подходов к оптимизации матричных вычисление в процессе фильтрации Калмана позволяет повысить эффективность расчета матриц ковариаций Φ_t и $\Phi_{\mathrm{t}+1}^{-}$, формируемых в процессе вычисления вероятностей и характерных для каждого среза ДБС. Использование алгоритмов QL и QR разложений естественно накладывает ограничения, связанные с размером исходной матрицы и выполнением условия ортогональности. Однако в рамках решения задачи фильтрации они являются наиболее применимыми и адаптированными алгоритмами.

Процедура сглаживания позволяет получить вероятностные распределения для моментов времени $t-1, t-2, \dots, t-k$. Под потоком свидетельств в процессе сглаживания будем понимать множество $E = E_{t-k}, \dots, E_t$. Наибольшее распространение получили алгоритмы сглаживания на основе фиксированной задержкой [39]. В основе алгоритмов используется предположение о марковости для переменной X_k, которое заключается в том, что переменная X_k будет условно независима от $E_{k+1:t}$ при наличии X_{k+1} для среза $k+1$. Тогда из формулы Байеса получаем распределение вероятностей для переменной X_k, условно-зависимой от переменной X_{k+1} и свидетельств $E_{1:t}$

$$P(X_k|X_{k+1}, E_{1:t}) = P(X_k|X_{k+1}, E_{1:k}) = \frac{P(X_{k+1}|X_k, E_{1:k})P(X_k|E_{1:k})}{P(X_{k+1}|E_{1:k})} = \frac{P(X_{k+1}|X_k)P(X_k|E_{1:k})}{P(X_{k+1}|E_{1:k})}. \quad (3.45)$$

Полное совместное распределение по переменным X_k и X_{k+1} при наличии свидетельств $E_{1:t}$ можно записать в следующем виде:

$$P(X_k, X_{k+1}|E_{1:t}) = P(X_k|X_{k+1}, E_{1:k})P(X_{k+1}|E_{1:t}) = \frac{P(X_{k+1}|X_k)P(X_k|E_{1:k})P(X_{k+1}|E_{1:t})}{P(X_{k+1}|E_{1:k})}, \quad (3.46)$$

где $P(X_{k+1}|E_{1:t})$ – сглаживание для момента времени $k+1$.

Распределение $P(X_k|E_{1:t})$ может быть получено из выражения (3.46) путем суммирования по X_k

$$P(X_{k+1}|E_{1:k}) = \sum_{X_k} P(X_{k+1}|X_k)\, P(X_k|E_{1:k}),$$
$$P(X_k|E_{1:t}) = P(X_k|E_{1:k}) \sum_{X_{k+1}} \frac{P(X_{k+1}|X_k)P(X_{k+1}|E_{1:t})}{P(X_{k+1}|E_{1:k})}. \quad (3.47)$$

Выражение (3.47) описывает алгоритм сглаживания с фиксированной задержкой. Основная сущность данного алгоритма заключается в оценке вероятности для временного среза, отстоящего от текущего t на k значений. Это приводит к тому, что сглаживание будет выполняться для каждого временного среза $t+k$ по мере добавления новых свидетельств. Использование данного

алгоритма при большом количестве узлов ДБС становится затруднительно, так как требуется многократный перерасчет вероятностей $P(X_{k+1}|E_{1:t})$ при поступлении каждого нового свидетельства.

Для обхода данных ограничений используется алгоритм сглаживания, основанный на двухэтапной фильтрации [27]. Применение данного алгоритма основывается на вычислении следующего распределения вероятностей:

$$P(X_k|E_{1:n}) = P(X_k|E_{1:k-1})P(E_{k:n}|X_k), \tag{3.48}$$

где первый множитель $P(X_k|E_{1:k-1})$ описывает процесс фильтрации для временного шага $k-1$, а второй может быть получен рекурсивным способом на основе следующего выражения:

$$\begin{aligned} P(E_{k+1:n}|X_k) &= \sum_{X_{k+1}} P(E_{k+1:n}, X_{k+1}|X_k) \\ &= \sum_{X_{k+1}} P(E_{k+1:n}|X_{k+1}, X_k)P(X_{k+1}|X_k) \\ &= \sum_{X_{k+1}} P(E_{k+1:n}|X_{k+1})P(X_{k+1}|X_k), \end{aligned} \tag{3.49}$$

$$\begin{aligned} P(E_{k:n}|X_k) &= P(E_{k+1:n}, E_k|X_k) = P(E_{k+1:n}|X_k)P(E_k|E_{k+1:n}, X_k) \\ &= P(E_{k+1:n}|X_k)P(E_k|X_k). \end{aligned} \tag{3.50}$$

Наряду с алгоритмами сглаживания с фиксированной задержкой особый интерес представляют алгоритмы сглаживания на основе фильтра Калмана. Одним из наиболее адаптивных алгоритмов сглаживания применительно к семантике динамических байесовских сетей является алгоритм Рауча-Тюнга-Штрибеля (РТШ). Данный алгоритм сглаживания подходит для анализа непрерывных моделей, при этом временные срезы могут иметь как дискретное представление, так и непрерывное. Исходя их этого, задача сглаживания применительно к алгоритму РТШ может быть приведена к следующему виду [71]:

$$P(X_k|E_{1:t+k}) = N\left(X_t\middle|\tilde{X}_t, \Phi_t\right). \tag{3.51}$$

Основное отличие формулы (3.51) от модели фильтрации Калмана (3.19) заключается в том, что для решения задачи сглаживание необходимо определить все свидетельства $E_{1:t+\mathrm{k}}$, в то время как для фильтрации необходимо

существование свидетельств до заданного состояния $E_{1:t+1}$, включая текущее состояние $t+1$. Рассмотрим процедуру сглаживания на основе подхода РТШ. Структурно процедура сглаживания на основе РТШ может быть разделена на две основные фазы: фильтрация и сглаживание. Первая фаза характеризуется получением оценки параметров ДБС $\tilde{X}_t^f$ на временном интервале $0 < t \leq k$ на основе фильтра Калмана (фаза фильтрации) – выражения (3.21) и (3.22). На второй фазе вычисляется оценка $\tilde{X}_{t+1}^s$ на основе алгоритма обратной фильтрации (фаза РТШ). Для построения алгоритма РТШ определим оценку $\tilde{X}_{t+1}$, соответствующую временному срезу $t+1$, на основе выражения (3.16)

$$\tilde{X}_{t+1} = F_{t,t+1}^{-1} X_{\mathrm{t+1}} - F_{t+1,}^{-1} E'_{t+1} \tag{3.52}$$

где E'_{t+1} – наблюдаемые переменные, $F_{t,t+1}^{-1}$ – обратная матрица по отношению к матрице модели перехода $F_{t,t+1}$. Использование обратной матрицы обусловлено тем, что выполнение алгоритма производится в обратном порядке, начиная с конечного момента времени $t+k$ и завершается состоянием t.

Для описания алгоритма введем следующие обозначения:

$$\begin{gathered} \Theta_t = \Phi_t^{-1}, \\ \Theta_t^- = (\Phi_t^-)^{-1}, \\ \tilde{z}_t = \Theta_t \tilde{X}_t^s, \\ \tilde{z}_t^- = \Theta_t^- \tilde{X}_t^{s-}. \end{gathered} \tag{3.53}$$

В основе алгоритма оптимального сглаживания лежит процедура вычисления оценки $\tilde{X}_{t+1}$ для каждого из параметров ДБС и связанных с ними ковариационных матриц ошибок. С учетом того, что шумовые процессы w_k и v_k являются некоррелированными, то можно определить выражение для ковариационной матрицы ошибок, соответствующей каждой из оценок $\tilde{X}_t$

$$\begin{gathered} \Phi_t = [(\Phi_{\mathrm{e}}^{s-})^{-1} + H^T R^{-1} H]^{-1} \\ = \left[\left(\Phi_t^f\right)^{-1} + (\Phi_t^s)^{-1}\right]^{-1} = \left[\left(\Phi_t^f\right)^{-1} + \Theta_t^-\right]^{-1}. \end{gathered} \tag{3.54}$$

Используя лемму об инверсии матриц (тождество Шермана-Моррисона-Вудбери), выражение для ковариационной матрицы ошибок Φ_t может быть переписано в следующей форме:

$$\Phi_t = \Phi_t^f - \Phi_t^f\left(\Phi_t^{s-} + \Phi_t^f\right)^{-1}\Phi_t^f = \Phi_t^f - \Phi_t^f\Theta_t^-\left(I + \Phi_t^f\Theta_t^-\right)\Phi_t^f. \quad (3.55)$$

Из выражения (3.55) получаем апостериорное значение ковариационной матрицы ошибок Φ_t, полученное в результате выполнения процедуры сглаживания [50]. Важно отметить, что значение Φ_t не может превышать соответствующее значение ковариационной матрицы ошибок Φ_t^f, полученное в результате выполнения фазы фильтрации на основе алгоритма Калмана. Это связано с тем, что в процессе сглаживания получаем дополнительные данные из состояний $t+k$. Тогда апостериорное значение оценки сглаживания может быть получено на основе следующего выражения:

$$\tilde{X}_t = \Phi_t\left(\left(\Phi_t^f\right)^{-1}\tilde{X}_t^f + (\Phi_t^{s-})^{-1}\tilde{X}_t^{s-}\right),$$
$$\tilde{X}_t = \tilde{X}_t^f + \left(\Phi_t\tilde{z}_t^- - G_t\tilde{X}_t^f\right), \quad (3.56)$$
$$G_t = \Phi_t^f\Theta_t^-\left(I + \Phi_t^f\Theta_t^-\right)^{-1},$$

где G_t – коэффициент сглаживания.

С учетом того, что алгоритм РТШ использует процедуру обратной фильтрации, расчет ковариационных матриц для фазы фильтрации и сглаживания может быть получен из выражения (3.54) и имеет следующий вид:

$$\Phi_t^f + \Phi_t^{s-} = F_{t,t+1}^T\left(\Phi_{t+1}^{f-} + \Phi_{t+1}^s\right)^{-1}F_{t,t+1} = F_{t,t+1}^T\left(\Phi_{t+1}^{f-} + \left[\Phi_{t+1}^{-1} - \left(\Phi_{t+1}^{f-}\right)^{-1}\right]^{-1}\right)^{-1}F_{t,t+1}. \quad (3.57)$$

Применяя лемму об инверсии матриц, выражение (3.57) может быть приведено к следующему виду:

$$\Phi_t^f + \Phi_t^{s-} = F_{t,t+1}^T\left(\Phi_{t+1}^{f-}\right)^{-1}\left[\Phi_{t+1}^{f-} - \Phi_{t+1}\right]\left(\Phi_{t+1}^{f-}\right)^{-1}F_{t,t+1}. \quad (3.58)$$

Учитывая выражения (3.55) и (3.56), можно получить искомое значение ковариационной матрицы ошибки, соответствующее алгоритму РТШ

$$\Phi_t = \Phi_t^f - \Phi_t^f F_{t,t+1}^T\left(\Phi_{t+1}^{f-}\right)^{-1}\left(\Phi_{t+1}^{f-} - \Phi_{t+1}\right)\left(\Phi_{t+1}^{f-}\right)^{-1}F_{t,t+1}\Phi_t^f. \quad (3.59)$$

Далее определим матрицу усиления как произведение следующих величин:

$$A_t = \Phi_t^f F_{t,t+1}^T \left[\Phi_{t+1}^{f-}\right]^{-1}. \quad (3.60)$$

С учетом выражения (3.60) для матрицы усиления, получим обобщенное значение ковариационной матрицы ошибок

$$\Phi_t = \Phi_t^f - A_t\left(\Phi_{t+1}^{f-} - \Phi_{t+1}\right)A_t^T. \quad (3.61)$$

С учетом того, что Φ_t имеет непосредственную зависимость от ковариационных матриц Φ_t^{f-} и Φ_t^f, соответствующих фазе фильтрации, оценка каждого параметра ДБС $\tilde{X}_t$ может быть вычислена на основе следующего выражения:

$$\tilde{X}_t = \tilde{X}_t^f + A_t\left(\tilde{X}_{t+1} - \tilde{X}_{t+1}^{f-}\right). \quad (3.62)$$

Важно отметить, что процесс вычисления выражений (3.60) и (3.62) носит рекурсивный характер. За начальное время отчета берется значение $t = n$, после чего выполнение фазы фильтрации повторяется до того момента, пока значение t не будет равно 0. Исходное значение для ковариационной матрицы ошибок $\Phi_{t=n}$ и оценки параметров динамической байесовской сети $\tilde{X}_{t=n}$ можно записать в следующей форме:

$$\begin{aligned} \tilde{X}_n &= \tilde{X}_n^f, \\ \Phi_n &= \Phi_n^f. \end{aligned} \quad (3.63)$$

Использование механизмов сглаживания на базе алгоритма РТШ является достаточно эффективным инструментов для оценки параметров ДБС с непрерывными параметрами.

В первую очередь, в отличие от классических алгоритмов, РТШ позволяет скорректировать процесс определения ковариационных матриц ошибок за счет выполнения повторной фильтрации. Использование данного подхода также позволяет определить коэффициент сглаживания G_t, представленный в формуле (3.56), что дает возможность оценить динамику изменения ковариационных матриц ошибок Φ_t и оценок параметров $\tilde{X}_t$ динамической байесовской сети в процессе выполнения процедур прямой и обратной фильтрации на заданном временном срезе $t \in [0; n]$. Подход, предложенный в алгоритме РТШ является достаточно универсальным и в полной мере адаптированным к ДБС.

3.2. Стохастические гибридные методы повышения эффективности процедур вероятностного вывода

В процессе решения задач вероятностного вывода для анализа состояний переменных ДБС особый интерес представляют стохастические подходы для реализации процедур вероятного вывода, построенные на основе метода Монте-Карло.

Использование данного метода в рамках вероятностного вывода является достаточно эффективным. Общий подход, используемый в методе Монте-Карло, был представлен в параграфе 2.2. Рассмотрим разновидности процедур вероятностного вывода в ДБС на основе метода Монте-Карло. Структурно данные методы можно разделить на два основных типа: автономные и онлайн. Основное отличие данных типов заключается в том, что автономные предусматривают развертывание сети за счет повторения временных срезов до того момента, пока не будут учтены все наблюдения, необходимые для решения поставленной задачи. Тем самым происходит преобразование ДБС к статической БС, после чего выполняется процедура логического вывода внутри такой сети. Онлайн ,напротив, не требуют полного развертывания сети и наиболее адаптированы для решения задач стохастического вывода в ДБС с большим числом временных срезов и параметров. Особенность данных методов обоснована тем, что в рассматриваемый момент времени достаточно хранить информацию о двух соседних срезах данной сети. Рассмотрим каждый из методов по отдельности. К автономным методам относятся: выборка по значению, выборка с исключение, оценка веса с учетом правдоподобия. Приведем более детально описание каждого из данных подходов для реализации процедуры логического вывода.

Алгоритм поиска выборки с исключением (ВИ) является одним из базовых подходов для вероятностного вывода и направлен на упрощение получения искомого распределения вероятностей $P(X|E)$ при поступлении свидетельств E. На начальном этап алгоритм выполняет формирование выборок на основе начального распределения $P(X_0)$, после чего отбрасываются выборки несогласующиеся со

свидетельствами E. В завершение для каждой переменной сети формируется оценка $\tilde{P}(X = x|E)$ на основе частотного анализа оставшихся выборок, содержащих значение переменной $X = x$. Исходя из того, что распределение $\tilde{P}(X|E)$ является результирующим для ВИ, можно получить искомое распределение $P(X|E)$

$$\tilde{P}(X|E) = \frac{N_s(X,E)}{N_s(E)},$$

$$\lim_{N\to\infty} \frac{N_s(X_1, X_2, \dots, X_n)}{N} = P(X_1, X_2, \dots, X_n), \tag{3.64}$$

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^{n} P\big(X_i \big| Parents(X_i)\big),$$

где S – выборки, сформированные из начального распределения вероятностей $P(X_0)$, $N_S(X_1, X_2, \dots, X_n)$ – количество раз, которое встречается событие $X_1, X_2, \dots, X_n$ в выборке S, N – общее число выборок S.

В результате из выражения (3.64) можно получить следующее соотношение:

$$P(X_1, X_2, \dots, X_n) \approx \frac{N_s(X_1, X_2, \dots, X_n)}{N},$$

$$\tilde{P}(X|E) \approx \frac{P(X,E)}{P(E)} = P(X|E). \tag{3.65}$$

Математическое описание алгоритма ВИ является простым, однако дает возможность получить согласованную с множеством свидетельств оценку для распределения $P(X|\mathrm{E})$. Основные недостатки данного алгоритма проявляются при увеличении числа свидетельств E. В результате алгоритм исключает большинство несогласованных со свидетельством E выборок S, что приводит к получению некорректной оценки $\tilde{P}(X|E)$.

Для обхода ограничений подхода, используемого в ВИ, наибольший интерес представляет алгоритмы на основе взвешивания с учетом правдоподобия (ВП). Данные алгоритмы позволяют изначально производить генерация только тех выборок S , которые полностью согласуются со свидетельствами $\mathrm{E} = (e_1, e_2, \dots, e_n)$. Данное условие выполнимо в том случае, если в процессе вероятностного вывода

на основе ВП фиксируем переменные E, а генерация выборок производится лишь для остальных переменных $Z = X \cup Y$, где X переменные запроса и Y переменные состояния ДБС. По результатам работы алгоритма получаем набор выборок S_{ws} для всех переменных состояния. Каждая выборка $s_{ws}^{i} \in S_{ws}$ взвешивается в соответствии со своим правдоподобием, относительно свидетельств.

Рассмотрим процедуры генерации выборок, а также оценивания весов данных выборок [208]. С учетом того, что переменная $Z = (\mathrm{Z}_1, \mathrm{Z}_2, \ldots, \mathrm{Z}_n)$ имеет непосредственную зависимость лишь от множества своих родителей $Parents(Z)$, распределение вероятностей по всем выборкам S_{ws} будет иметь следующий вид:

$$P_{ws}(Z, E) = \prod_{i=1}^{m} P\big(Z_i \big| Parents(Z_i)\big), E \subset Parents(Z_i). \tag{3.66}$$

Из формулы (3.66) следует, что каждое из свидетельств $e_i \in E$ может вносить вклад в апостериорное распределение по всем выборкам S_{ws}, в том случае, если оно принадлежит родительским вершинам $e_i \in Parents(Z_i)$. Значения весов выборок $w(Z, E)$, формируемых на основе оценки правдоподобия выборок, представляют собой величину, характеризующую разницу между ожидаемым и реальным распределениями вероятностей, полученными для переменной Z и выборки S_{ws}. Каждый вес $W_{ws}(Z, E)$ представляет собой произведение логарифма правдоподобия для свидетельств $E_i \in E$ и имеет место только в том случае, если для каждого E_i существует множество родительских вершин $Parents(E_i) \neq \emptyset$

$$W_{ws}(Z, E) = \prod_{i=1}^{m} P\big(e_i \big| Parents(E_i)\big). \tag{3.67}$$

Производим перемножение выражений (3.66) и (3.67). В результате получим требуемое соотношение для расчета $P(Z|E)$ на основе алгоритма ВП [197]

$$\begin{aligned} P_{ws}(Z, E) W_{ws}(Z, E) &= P(Z|E) \\ &= \prod_{i=1}^{m} P\big(Z_i \big| Parents(Z_i)\big) \prod_{i=1}^{n} P\big(e_i \big| Parents(E_i)\big) \end{aligned} \tag{3.68}$$

При этом несложно доказать условие согласованности весов, полученных в процессе взвешивания с учетом правдоподобия [30] для каждой рассматриваемой переменной $X_i \in X$

$$P(X|E) = \sum_Y N(X,Y,E)W(X,Y,E)$$
$$\approx \sum_Y P_{ws}(X,Y,E)W(X,Y,E) = \sum_Y P(X,Y,E) = P(X|E). \quad (3.69)$$

По сравнению с ВИ алгоритм ВП обладает главным преимуществом – используются все выборки, генерируемые в процессе выполнения алгоритма. Это позволяет снизить общее число выборок для получения искомого апостериорного распределения вероятностей $P(X|E)$ с допустимым уровнем точности. В свою очередь, алгоритм ВП не лишен недостатков. Главный из них заключается в том, что по мере усложнения структуры ДБС и, как следствие, роста числа свидетельств E, алгоритм начинает порождать выборки, имеющие достаточно низкие весовые значения, поэтому их вклад в результирующее распределение будет ограниченным, а общее число выборок, в наибольшей степени согласующихся со свидетельствами E, будет постоянно снижаться.

Применение алгоритмов, формируемых на основе метода Монте-Карло и цепей Маркова (МКМЦ) позволяет преодолеть недостатки классических алгоритмов стохастического вывода на основе метода Монте-Карло. Сущность общего подхода, применяемого в методе МКМЦ, описана формулами (2.113) и (2.114). В свою очередь, применение метода МКМЦ позволяет преодолеть недостатки классических подходов вероятностного вывода за счет того, что процесс генерации выборок будет происходить путем внесения случайного изменения в выборку одной из переменный X_k, вязтой из предыдущего состояния.

Процесс формирования S осуществляется за счет построения цепи Маркова для каждой переменной запроса X, ассоциируемой с выборкой. Каждая последующая выборка S' формируется путем внесения случайных изменений в текущую выборку S. Простейшими реализациями метода МКМЦ являются алгоритмы Метрополиса-Гастингса (МГ) и Гиббса (ГБ). Данные алгоритмы

подробно описаны нами в параграфе 2.3. Далее рассмотрим процесс адаптации данных алгоритмов к решению задач вероятностного вывода в ДБС. Для простоты введем следующие обозначения: X – текущая (наблюдаемая) переменная, соответствующая временному срезу t, для которой производится формирование выборки, Z – множество ненаблюдаемых переменных, E – свидетельства, характерные для события X, X' – значение параметра, полученное для временного среза $t+1$. Тогда переходная вероятность для алгоритма МГ может быть получена путем использования цепного правила и преобразования формул (2.118) и (2.119) к следующему виду [17,102]:

$$\begin{gathered} P(X,X') = Q(X,X')\varphi(X,X'), \\ P(X)Q(X,X')\varphi(X,X') = P(X|E)P(X'|X,E)\varphi(X'|X,E) \\ = P(X|Z,E)P(Z|E)P(X'|Z,E) = P(X|Z)P(X',Z|E) \\ = P(X')Q(X',X), \end{gathered} \tag{3.70}$$

$$\varphi(X,X') = \frac{P(X')Q(X',X)}{P(X)Q(X,X')}. \tag{3.71}$$

Из алгоритма Гиббса (формула (2.121)) видно, что в процессе выполнения алгоритма берется лишь одна текущая переменная X', а остальные фиксируются. Из этого следует, что для переменной X' выполняется свойство условной независимости. Для определения факта условной независимости X' необходимо задать ее марковское покрытие $\xi(X)$. Исходя из этого, формула (2.121) может быть преобразована с учетом особенностей вероятностного вывода в ДБС и ее можно переписать в следующем обобщенном виде:

$$\begin{gathered} P(X'|Z) = P\big(X'\big|\xi(X)\big), \\ = P\big(X'\big|Parents(X)\big) \times \prod_{C_j \in Children(X)} P(C_j|Parents(C_j)), \end{gathered} \tag{3.72}$$

где C – множество дочерних вершин по отношению к переменной X.

Применение метода МКМЦ и его разновидностей впервые было апробировано Д. Перлом в своей работе по исследования особенностей оптимизации процедуры вероятностного вывода [59]. Алгоритмы на основе МКМЦ доказали свою эффективность в рамках решения задач вероятностного вывода

статических байесовских сетей, однако их применение к ДБС ограничивается процедурой развертывания сети, что накладывает дополнительные временные и ресурсные затраты. Формирования выборок в процессе выполнения алгоритмов на основе метода МКМЦ экспоненциально зависит от числа временных срезов. В первую очередь это связано с тем, что формирование выборок происходит сразу для всех срезов одновременно, как следствие, формируется достаточно большое число выборок, несогласованных со свидетельствами. Для преодоления ограничений классических алгоритмов на основе МКМЦ целесообразно использовать алгоритмы на основе последовательного метода Монте-Карло (ПМК), в частности многочастичный фильтр (МЧФ). Классическая схема МЧФ представлена на рисунке 3.1

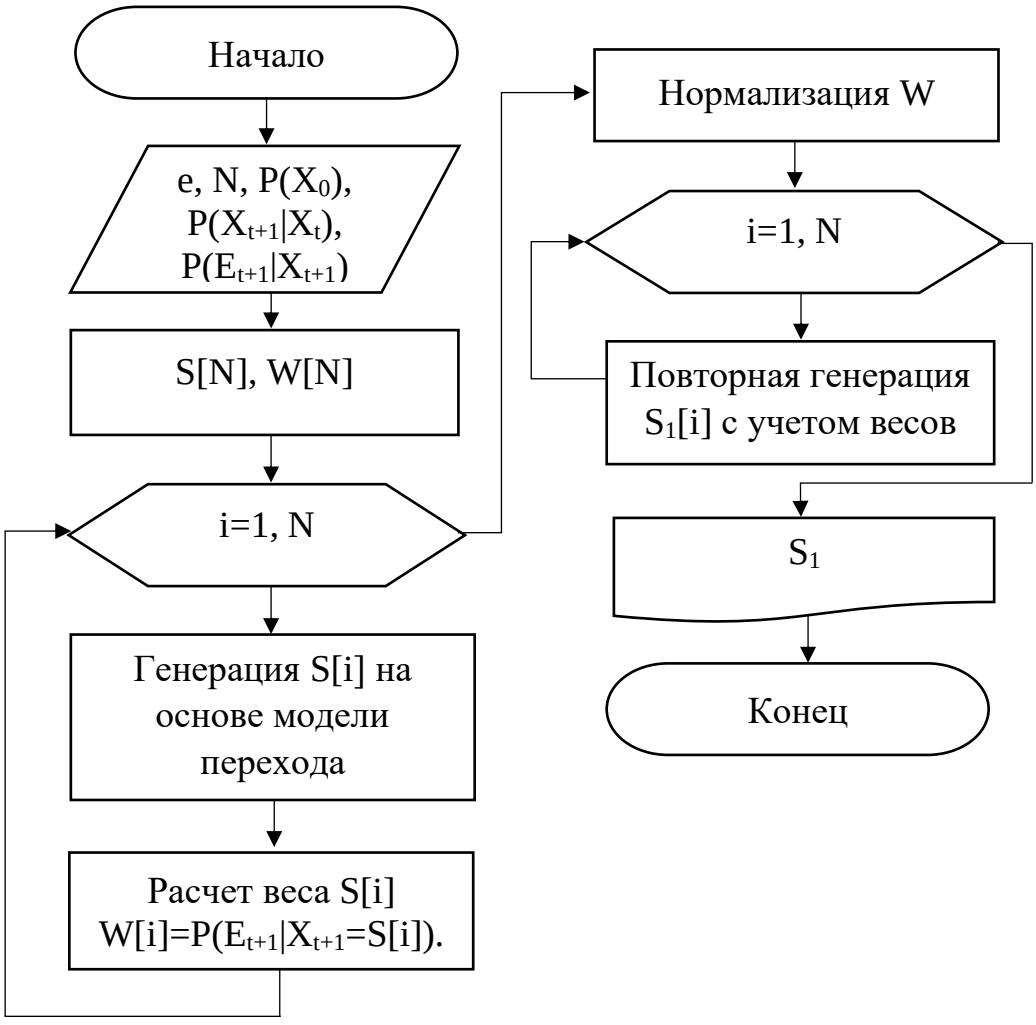

Рисунок 3.1. Структурная схема фильтра МЧФ.

Для получения апостериорного распределения вероятностей в МЧФ используется алгоритм формирования выборок по значимости (ВЗ). Основная

сущность данного алгоритма заключается в представлении плотности апостериорного распределения вероятностей для величины X_{t+1} в виде совокупности точек $\{X_{t+1}^i\}_{i=1}^n$ и их весов $\{W_{t+1}^i\}_{i=1}^n$. Интеграл, получаемый с помощью метода Монте-Карло будет иметь следующий вид [2,28]:

$$I = \int_G F(X)P(X) = \int_G F(X)Q(X)W(X) \approx \frac{1}{N}\sum_{i=1}^{N} F(X^i)W_i, \qquad W(X) = \frac{P(X)}{Q(X)}, \tag{3.73}$$

где $Q(X)$ – распределение по значимости. $W_i = W(\mathrm{X^I})$ – вес, соответствующий выборке $\mathrm{X^i}$.

Тогда аппроксимацию распределения $P(X_{t+1}|E_{t+1})$, характерную для среза $t+1$, можем выразить через следующую δ-функцию [26]

$$P(X_{t+1}|E_{t+1}) \approx \sum_{i=1}^{N} W_{t+1}^i \,\delta\big(X_{t+1} - X_{t+1}^i\big). \tag{3.74}$$

Из выражения (3.74) следует:

$$Q(X_{t+1}|E_{t+1}) \approx \sum_{i=1}^{N} \delta\big(X_{t+1} - X_{t+1}^i\big). \tag{3.75}$$

Распределение $P(X_{t+1}|E_t)$ и $Q(X_{t+1}|E_{t+1})$ запишем с помощью уравнения Чепмена-Колмогорова

$$P(X_{t+1}|E_t) = \int P(X_{t+1}|X_t)\, P(X_t|E_t)dX_t, \tag{3.76}$$

$$Q(X_{t+1}|E_{t+1}) = Q(X_{t+1}|E_{t+1}, E_t) = \int P(X_{t+1}|X_t, E_{t+1})\, P(X_t|E_t)dX_{t-1}. \tag{3.77}$$

Применяя формулу Байеса, определим распределение $P(X_{t+1}|E_{t+1})$ в виде следующего выражения [169]:

$$P(X_{t+1}|E_{t+1}) = P(X_{t+1}|E_{t+1}, E_t) = \frac{P(E_{t+1}|X_{t+1}, E_t)P(X_{t+1}|E_t)}{P(E_{t+1}|E_t)} = \mathrm{c}P(E_{t+1}|X_{t+1})P(X_{t+1}|E_t), \tag{3.78}$$

где c – нормализующая константа.

Перепишем выражение (3.74) с учетом (3.76) и (3.78):

$$P(X_{t+1}|E_{t+1}) \approx cP(E_{t+1}|X_{t+1}) \sum_{i=1}^{N} W_{t+1}^{i} P\left(X_{t+1}^{i}|X_{t}^{i}\right). \quad (3.79)$$

В соответствии с выражением (3.73) имеем

$$P(X_{t+1}|E_{t+1}) = W_{t+1} Q(X_{t+1}|E_{t+1}). \quad (3.80)$$

Приравнивая (3.79) и (3.80) и подставляя выражения (3.75) и (3.77), получим следующее выражение для расчета весов W_{t+1}:

$$W_{t+1} = cW_t \frac{P(E_{t+1}|X_{t+1})P(X_{t+1}|X_{t+1})}{Q(X_{t+1}|X_t, E_{t+1})}. \quad (3.81)$$

С учетом того, что константу c можно исключить из расчета. Получим [54,55]

$$\widetilde{W}_{t+1} = W_t \frac{P(E_{t+1}|X_{t+1})P(X_{t+1}|X_{t+1})}{Q(X_{t+1}|X_t, E_{t+1})}, \quad W_{t+1} = \frac{\widetilde{W}_{t+1}}{\sum_{i=1}^{N} \widetilde{W}_{t+1}}. \quad (3.82)$$

В процессе определения весов по значимости часто предполагается, что формирование распределения по значимости для каждой из сгенерированных выборок производится из априорного распределения

$$Q(X_{t+1}|X_t, E_{t+1}) = P(X_{t+1}|X_t) \quad (3.83)$$

Тогда на основании формулы (3.83) выражение (3.82) можно привести к следующему упрощенному виду:

$$\widetilde{W}_{t+1} = W_t P(E_{t+1}|X_{t+1}) \quad (3.84)$$

На практике в процессе применения ВЗ возникает рост дисперсии весов для элементов выборки, что приводит к тому, что большинство весов W_{t+1} попадают в область наименьших значений, в следствие чего плотность распределения становится некорректной, а сама выборка становится вырожденной. Для оценки доли вырожденных выборок можно определить параметр эффективного размера выборки $\widehat{N}_{eff}$

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_s} \left(W_{t+1}^{i}\right)^2}, \quad (3.85)$$

где W_{t+1}^{i} – нормализованные веса, полученные из выражения (3.82).

Алгоритм ВЗ поможет быть дополнен фазой повторного формирования выборки. Критерием останова данной фазы является достижение некоторого порога N_{max}. Сама процедура формирования выборок выполняется на основе следующих шагов [177]. На первом шаге происходит определение весов выборок и их принятие с вероятностью $P_a = \min\{1, W_{t+1}^i / N_{max}\}$. На следующем шаге происходит переопределение весов для всех выборок, которые не были отброшены на первом этапе, $W_i^{t+1} = \max\{N_{max}, W_{t+1}^i\}$.

В большинстве случаев применение классических алгоритмов на основе МКМЦ не всегда позволяют получить апостериорное распределение с соответствующей точностью. В первую очередь это связано с тем, что необходимо определить веса генерируемых выборок. Вычисление весов позволяет определить, насколько выборка согласуется со свидетельством, как следствие можно получить распределение по выборкам, обладающее требуемым уровнем точности. Для решения таких задач используются гибридные методы. Наибольший интерес среди гибридных подходов на основе ПМК представляет МЧФ в сочетании алгоритмами Боена-Коллера (БК) и ВП. Использование гибридных методов вероятностного вывода в первую очередь направлено на оптимизацию расчета и оценки весов генерируемых выборок.

В основе вероятностного алгоритма Боена-Коллера (БК) используется интерфейсный алгоритм (ИА). ИА использует дерево сочленений (параграф 2.2) для построения сильно связного дерева и определения так называемых интерфейсных узлов. В ИА в качестве критерия используется условие d-разделенности, гарантирующее условную независимость вершин, имеющих связи между временными срезами. Узлы, для которых выполняется такое условие, формируют прямое множество V'. Использование сильно связного дерева обусловлено необходимостью хранения промежуточных результатов вычисления алгоритма, в частности, в рамках определения интерфейсов между прошлыми и будущими состояниями ДБС. Для описания алгоритма введем следующие обозначения: G_t , G_{t+1} – графы, созданные для соответствующих моментов времени t и $t+1$, DBN_t – динамическая байесовская сеть из двух временных срезов и

созданная путем удаления из графа G_{t+1} всех узлов и ребер, не входящих в множество интерфейсных узлов графа G_{t+1} ($DBN_t = I_t \cup I_{t+1}$ для $t = 1, DBN_1 = I_1$). Алгоритм ИА в процессе построения дерева J_t накладывает ограничения, связанные с тем, что получаемые интерфейсы I_t и I_{t+1} для временных срезов t и $t+1$ должны использоваться в процессе формировании графа всех кликов J_t, участвующего в вычислении распределений вероятностей $P(I_t)$ и $P(I_{t+1})$ по заданным интерфейсам. Искомый результат получается за счет определения ребер внутри графа G_{t+1}, находящихся между узлами I_t. Определение ребер для I_{t+1} выполняется по такому же принципу. По завершению выполнения алгоритма все сильно связные деревья $J_{t+1} = (J_{t+1}^1, J_{t+1}^2, \dots, J_{t+1}^n)$ будет объединяться между собой через соответствующие интерфейсы. Следовательно, вероятностный вывод можно реализовать независимо для каждого дерева. Затем необходимо осуществить тиражирование запросов между всеми интерфейсными узлами сети, располагающимися на стыке каждого среза J_{t+1}^i в прямом и обратном направлении. Распространение в прямом направлении позволяет получить априорное значение доверительного состояния для каждого из интерфейсов $P(I_t|E_{1:t})$, при условии поступления свидетельств в процессе прохождении пути от C_{t-1} до C_t, C_{t-1} – клики в J_{t-1}, C_t – клики в J_t.

Алгоритм БК является стохастической надстройкой поверх алгоритма ИА и наиболее адаптирован к применению в рамках решения задач логического вывода во временных моделях. Он наиболее полезен в тех ситуациях, когда число интерфейсных кликов, полученных на основе применения алгоритма ИА, достаточно велико. Наиболее применимый метод повышения точности логического вывода напрямую завязан с аппроксимацией объединения интерфейсов, формируемых по результатам маргинализации термов небольшого размера. Реализация данного подхода является основополагающей для алгоритма БК. В процессе выполнения алгоритма производится создание сильно связного дерева ДБС DBN_{t+1}, состоящего из двух и более временных срезов. Данный алгоритм накладывает ряд ограничений. Все узлы, принадлежащие одному

интерфейсу, должны быть в одной клике. Вероятность $P(I_{t+1}|E_{1:t+1})$ нельзя представить в виде потенциала одной клики.

Последовательно выполняется объединение узлов в кластер. Точность алгоритма напрямую зависит от сформированных кластеров, используемых для определения стохастического значения доверительного состояния. В свою очередь для обеспечения максимальной точности вероятностного вывода необходимо иметь только один кластер, который будет содержать все узлы интерфейса. При использовании D кластеров для каждой из переменных можно получить аппроксимированное значение вероятности с полной факторизацией. Алгоритм БК является оптимизированной версией интерфейсного алгоритма.

Другим гибридным подходом к вероятностному выводу является МЧФ в сочетании с ВП. МЧФ с ВП в первую очередь характеризуется возможностью порождения выборок, обладающих наибольшим весом, вычисляемым на основе алгоритма ВП. Это приводит к тому, что с каждой итерацией алгоритма МЧФ доля выборок, обладающих большим весом, увеличивается. Как следствие имеем достаточно большую долю выборок близких к истинным значениям. Это дает возможность получения апостериорного распределения по всем выборкам, согласующимся со свидетельствами с требуемым уровнем точности. Применительно к ДБС алгоритм МЧФ первоначально формирует выборки заданной размерности N из априорного распределения $P(X_0)$. Следует отметить, что точность алгоритма прямо пропорциональна числу выборок. Тем самым, при $N \to \infty$ алгоритм показывает наилучшую точность. Далее происходит циклическое выполнение следующих основных фаз. На первой фазе производится генерация и распространение выборки внутри ДБС в прямом направлении для некоторого следующего состояния X_{t+1} при наличии текущего состояния переменной X_t. Формирование таких выборок производится на основе использования модели перехода $P(X_{t+1}|X_t)$. Получение первичной выборки для момента времени t происходит на основе распределения $P(X_t|E_{1:t})$. В таком случае количество из N выборок $N'(X_t|E_{1:t})$, соответствующих состоянию X_t при получении свидетельств $E_{1:t}$ может быть получено из следующего выражения:

$$N'(X_t|E_{1:t}) = N \times P(X_t|E_{1:t}). \quad (3.86)$$

После этого происходит распространение свидетельств для следующего момента времени $t+1$, при условии наличия выборок для среза t. Можно получить общее количество выборок, достигающих состояние X_{t+1} при переходе из предшествующего состояния X_t, как произведение вероятности перехода на выборку, полученную для состояния X_t

$$N'(X_{t+1}|E_{1:t}) = \sum_{X_t} P(X_{t+1}|X_t)\, N'(X_t|E_{1:t}). \quad (3.87)$$

На следующей фазе происходит формирование весов выборок на основе алгоритма ВП. Общий вес выборок конкретной популяции, соответствующей состоянию X_{t+1} при получении свидетельств E_{t+1} для среза $t+1$, можно получить из следующего выражения:

$$W(X_{t+1}|E_{1:t+1}) = P(E_{t+1}|X_{t+1})\mathrm{P}'(X_{t+1}|E_{1:t}). \quad (3.88)$$

На завершающей фазе происходит повторная генерация N выборок для формирования новой популяции. Каждая следующая выборка использует предыдущую популяцию в качестве первоначальной точки генерации. Вероятность того, что будет выбрана конкретная популяция, пропорциональная ее весу. Из этого следует, что общее число выборок X_{t+1} после повторной генерации будет пропорционально весу, сформированному исходя из формулы (3.88). Тогда окончательное выражение для числа сформированных выборок будет иметь вид

$$N'(X_{t+1}|E_{1:t+1}) = N \times W(X_{t+1}|E_{1:t+1}). \quad (3.89)$$

Используя формулы (3.86), (3.87) и (3.88), выражение (3.89) может быть приведено к следующему виду:

$$\begin{aligned} N'(X_{t+1}|E_{1:t+1}) &= N \times P(E_{t+1}|X_{t+1})P'(X_{t+1}|E_{1:t}) \\ &= N \times P(E_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)\, P'(X_t|E_{1:t}) \\ &\approx N \times P(E_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)\, P(X_t|E_{1:t}) \\ &= N \times P(X_{t+1}|E_{1:t+1}). \end{aligned} \quad (3.90)$$

Применение МЧФ в сочетании с ВП для ДБС позволяет получить достаточно точные результаты, при этом дает возможность избежать полного развертывание сети, что существенно снижает ресурсные и вычислительные затраты по отношению к классическим алгоритмам на основе метода МКМЦ. Однако наиболее существенный недостаток МЧФ с ВП заключается в том, что для достижения требуемой точности результатов необходимо сгенерировать большое число выборок, что, в свою очередь, накладывает дополнительные временные затраты. В настоящее время использование высокопроизводительных вычислительных систем в целом позволяет решить этот недостаток, однако делает невозможным применение данного алгоритма на слабой аппаратной платформе. В связи с этим возникает задача оптимизации процедуры формирования выборок и определения ограниченного числа функций весов таких выборок. В качестве данных функций будут рассматриваться достаточные статистики.

Использование полных выборок достаточно проблематично, особенно если ДБС имеет сложную разветвленную структуру. Применение статистических методов позволяет оптимизировать решение задач вероятностного вывода в ДБС без потери необходимой информации, содержащейся в выборках, участвующих в формировании искомого апостериорного распределения вероятностей для ДБС. Наибольший интерес представляет подход, предложенный К. Рао, Д. Блэквеллом и А. Колмогоровым. Рассмотрим метод оптимизации МЧФ на основе теоремы Рао-Блэквелла-Колмогорова (РБК).

Теорема РБК является основополагающим утверждением математической статистики, направленным на улучшение статистических показателей оценки параметров с точки зрения среднеквадратичного отклонения. Под оценкой случайной величины X из некоторой выборки $X = (X_1, X_2, \dots, X_n)$ будем понимать функцию $T_1(X)$, зависящую от X, но независящую от параметра θ.

В рамках исследования достаточной статистикой для параметра $\theta \in \Theta$ будем называть функцию $T(X) = \big(T_1(X), T_2(X), \dots., T_n(X)\big)$ для которой распределение случайной величины X относительно $P\big(X\big|T(X)\big)$ не зависит от θ. Тогда имеем следующее равенство:

$$P\left(X \in \widetilde{X} | T(X) = t, \theta\right) = P\left(X \in \widetilde{X} | T(X) = t\right). \quad (3.91)$$

Данное выражение свидетельствует о том, что использование выборочного значения достаточной статистики $T(X)$ позволяет получить все необходимые данные для формирования эквивалентной выборки (ЭВ). Основная идея использования достаточных статистик заключается в снижении сложности решении статистической задачи без уменьшения информативности генерируемых выборок. Особый интерес представляет задача нахождения минимальной статистики. Минимальной достаточной статистикой $S(X)$ называется такая достаточная статистика, что для любой другой достаточной статистики существует измеримая функция g, что $S(X) = g\left(T(X)\right)$ почти всюду. Рассмотрим основные способы нахождения минимальных достаточных статистик.

Процедура формирования ЭВ может быть получена с использованием методов Монте-Карло, в частности, рассмотренного ранее алгоритма МГ. В таком случае, если известно множество случайных величин $X_1, X_2, \dots, X_n$, независящих от θ, при заданной достаточной статистике $T(X)$, можно определить выборку $Y_1, Y_2, \dots, Y_m$, распределение которой будет эквивалентно распределению величин $X_1, X_2, \dots, X_n$ при определенной достаточной статистике $T(X)$. Для установления факта, что $T(X)$ является достаточной статистикой [145] для случайных величин $X_1, X_2, \dots, X_n$, необходимо определить условное распределение параметров $X_1, X_2, \dots, X_n$ при заданном значении $T(X)$.

Задача теоремы РБК сводится к уменьшению среднеквадратической ошибки оценки $T_1(X)$ параметра θ, которая определяется как $\mathbb{E}_\theta(T_1(X) - \theta)^2$. Сущность теоремы Рао-Блэквелла-Колмогорова заключается в том, что если $\mathrm{T}(X)$ есть достаточная статистика и $T_1(X)$ – оценка параметра θ, то для оценки $T_2(X) = \mathbb{E}_\theta\left(T_1(X) \middle| T(X)\right)$ при любом векторе z справедливо неравенство [196]

$$z\mathbb{E}_\theta(T_2(X) - \theta)^T(T_2(X) - \theta)z \leq z\mathbb{E}_\theta(T_1(X) - \theta)^T(T_1(X) - \theta)z. \quad (3.92)$$

По формуле условной дисперсии можно установить следующее соотношение, исходя из теоремы РБК:

$$\begin{aligned}\mathbb{D}\big(T_1(X)\big) &= \mathbb{E}\left(\mathbb{D}\big(T_1(X)\big|T(X)\big)\right) + \mathbb{D}\left(\mathbb{E}\big(T_1(X)\big|T(X)\big)\right)\\ &= \mathbb{E}\left(\mathbb{D}\big(T_1(X)\big|T(X)\big)\right) + \mathbb{D}\big(T_2(X)\big).\end{aligned} \tag{3.93}$$

Из выражения (3.93) получаем неравенство дисперсий [147]

$$\mathbb{D}(T_2(X)) \leq \mathbb{D}(T_1(X)). \tag{3.94}$$

Из неравенства (3.94) следует, что оптимальная оценка может является функцией от достаточной статистики [131]. Несмещенной будем называть статистическую оценку $T_1(X)$ для которой. математическое ожидание будет равно значению оцениваемого параметра θ. Критерий несмещенности оценки можно записать в следующем виде [37]:

$$\begin{aligned}\mathbb{E}_\theta\big(T_1(X)\big) &= \theta, \forall\theta \in \Theta,\\ b(\theta) &= \mathbb{E}_\theta\big(T_1(X)\big) - \theta,\end{aligned} \tag{3.95}$$

где $b(\theta)$ – смещение оценки $T_1(X)$. Если $T_1(X)$ является несмещенной оценкой, то $b(\theta) = 0$. Значение среднеквадратической ошибки для несмещенной оценки будет иметь следующий вид [6,7]:

$$\delta(T_1(X);\theta) = \big(\mathbb{E}_\theta\big(T_1(X)\big) - \theta\big)^2 \tag{3.96}$$

Из формулы (3.96) следует, что оценку $T_1(X)$ можно выразить по формуле

$$\delta(T_1(X);\theta) = \mathbb{D}_\theta\big(T_1(X)\big) + b^2(\theta). \tag{3.97}$$

Из выражения (3.97) следует, что для несмещенной оценки ее среднеквадратическая ошибка будет совпадать с дисперсией. В таком случае значение несмещенной оценки $T_1(X)$ будет напрямую зависеть от достаточной статистики и иметь дисперсию всегда меньшую, чем первоначальная оценка. В классической формулировке теорема РБК позволяет определить критерий формирования наиболее оптимальных несмещенных оценок. Данный критерий заключается в том, что если существует полная достаточная статистика, то необходимо взять любую несмещенную оценку, а затем произвести ее усреднение по достаточной статистике. В таком случае оптимальную оценку можно однозначно определить из следующего выражения:

$$\mathbb{E}_\theta T_2(X) = \tau(\theta). \tag{3.98}$$

Оптимальную оценку τ' можно определить исходя из несмещенной оценки $T_2(X)$ параметрической функции $\tau(\theta)$

$$\tau' = T_2(X) = \mathbb{E}_\theta\big(T_1(X)\big|T(X)\big). \tag{3.99}$$

Однако вычисление оптимальной оценки по формуле (3.99) используется редко. В основном это связано со сложностью получения условного математического ожидания. Для упрощения данной процедуры применяется следующая математическая запись для функции $\tau(\theta)$, представляющая собой уравнение несмещенности оценки $T_2(\mathrm{X})$ [8,137]

$$\sum T_2(X)g(s;\theta) = \tau(\theta), \forall\theta \in \Theta, \tag{3.100}$$

где $g(s;\theta)$ – плотность распределения полной достаточной статистики $T(X)$.

Для решения уравнения несмещенности (3.100) можно использовать так называемый метод подгонки. Пусть $\tau(\theta) > 0$, тогда можно сформулировать и записать следующее тождество [134]:

$$\begin{gathered}\mathbb{E}_\theta\left[\frac{T_2(X)}{\tau(\theta)}\right] \equiv 1,\\ \sum\frac{T_2(X)g(s;\theta)}{\tau(\theta)} \equiv 1.\end{gathered} \tag{3.101}$$

Для определения искомого выражения для метода подгонки введем следующую факторизацию:

$$\frac{g(s;\theta)}{\tau(\theta)} = u(s)g_1(s;\theta). \tag{3.102}$$

С учетом факторизации (3.102) тождество (3.101) может быть обобщено и переписано в следующем виде:

$$\sum T_2(X)\,u(s)g_1(s;\theta) \equiv 1. \tag{3.103}$$

Тождеству (3.103) будет удовлетворять единственная функция $T_2(X) = 1/u(s)$. Если имеет место факторизация (3.102), то можно определить вид оптимальной оценки $T'(X)$.

Рассмотрим специфику применения теоремы РБК в процессе решения задач стохастического вывода в ДБС: фильтрация, прогнозирование, сглаживание. Для

этого будем использовать МЧФ с выборкой по значимости. Основная идея фильтрации частиц с применением теоремы РБК заключается в разделение множества всех ненаблюдаемых переменных X_t на U_t и V_t, при этом модель перехода будет определяться следующим образом:

$$P(X_{t+1}|X_t) = P(\ U_{t+1}|V_{t:t+1}, V_t)P(V_{t+1}|V_t). \tag{3.104}$$

Учеными А. Дусетом и С. Расселом [25] впервые представлена модель МЧФ с ВЗ на основе применения теоремы РБК. Предполагается, что условное распределение вероятностей $P(U_{t+1}|E_{1:t+1}, V_{t+1})$ может быть найдено аналитическим способом. В таком случае достаточно вычислить вероятность $P(V_{t+1}|E_{1:t+1})$. Для этого определим апостериорное распределение вероятностей для временного среза $t+1$ на основе цепного правила

$$P(V_{0:t+1}, U_{0:t+1}|E_{1:t+1}) = P(U_{0:t+1}|E_{1:t+1}, V_{0:t+1})P(V_{0:t+1}|E_{1:t+1}) \tag{3.105}$$

Тогда распределение $P(V_{0:t+1}|E_{1:t+1})$ будет определяться на основе следующего выражения:

$$P(V_{0:t+1}|E_{1:t+1}) = \frac{P(E_{t+1}|E_{1:t}, V_{0:t+1})P(V_{t+1}|V_t)P(V_{0:t}|E_{1:t})}{P(E_{t+1}|E_{1:t})} \tag{3.106}$$

Введем распределение по значимости $Q(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})$. Тогда оценка любой функции f_{t+1} от всех переменных запроса $V_{0:t+1}, U_{0:t+1}$ может быть получена на основе метода Монте-Карло

$$\hat{p}(V_{0:t+1}, U_{0:t+1}|E_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} \delta\left(X_{t+1} - X_{t+1}^i\right) \tag{3.107}$$

$$\begin{aligned} F(f_{t+1}) = & \iint f_{t+1}(V_{0:t+1}, U_{0:t+1})\hat{p}(V_{0:t+1}, U_{0:t+1}|E_{1:t})\, dV_{0:t+1}dU_{0:t+1} \\ & = \frac{1}{N}\sum_{i=1}^{N} f_{t+1}\left(V_{0:t+1}^i, U_{0:t+1}^i\right). \end{aligned} \tag{3.108}$$

Введем распределение по значимости $Q(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})$. В таком случае веса выборок по значимости можно определить в следующем виде:

$$W(V_{0:t+1}, U_{0:t+1}) = \frac{P(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})}{Q(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})}. \tag{3.109}$$

Нормализуя веса $W\left(V_{0:t+1}^i, U_{0:t+1}^i\right)$, получим

$$\hat{F}_{t+1} = \frac{\hat{A}^1_{t+1}}{\hat{B}^1_{t+1}} = \frac{\sum_{i=1}^{N} f_t\left(V^i_{0:t+1}, U^i_{0:t+1}\right) W\left(V^i_{0:t+1}, U^i_{0:t+1}\right)}{\sum_{i=1}^{N} W\left(V^i_{0:t+1}, U^i_{0:t+1}\right)} = \sum_{i=1}^{N} \widetilde{W}^i_{0:t+1} f_{t+1}\left(V^i_{0:t+1}, U^i_{0:t+1}\right). \quad (3.110)$$

Полагая что $Q(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})$ определено из усиленного закона больших чисел при $N \to +\infty$ получим обобщенную оценку $\hat{F}^1_{t+1}$

$$\hat{F}^1_{t+1} = \frac{\mathbb{E}_Q\left(f_{t+1}(V_{0:t+1}, U_{0:t+1}) W(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})\right)}{\mathbb{E}_Q\left(W(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})\right)}.. \quad (3.111)$$

Маргинализируя переменные $U_{0:t+1}$, получим альтернативную оценку для редуцированного набора для $V_{0:t+1}$ в соответствии с методом Монте-Карло

$$\hat{F}^2_{t+1} = \frac{\hat{A}^2_{t+1}}{\hat{B}^2_{t+1}} = \frac{\sum_{i=1}^{N} f_t\left(V^i_{0:t+1}, U^i_{0:t+1}\right) W\left(V^i_{0:t+1}\right)}{\sum_{i=1}^{N} W\left(V^i_{0:t+1}\right)}. \quad (3.112)$$

Тогда для весов $W(V_{0:t+1}, U_{0:t+1}), W\left(V^i_{0:t+1}\right)$ и оценок $\hat{F}^1_{t+1}, \hat{F}^2_{t+1}$ будет справедлива теорема Рао-Блэквелла-Колмогорова

$$\mathbb{D}(W(V_{0:t+1})) \le \mathbb{D}(W(V_{0:t+1}, U_{0:t+1})), \quad \mathbb{D}\left(\hat{A}^1_{t+1}\right) \le \mathbb{D}(A^2_{t+1}), \mathbb{D}\left(\hat{B}^1_{t+1}\right) \le \mathbb{D}(B^2_{t+1}). \quad (3.113)$$

Предполагая, что распределение ВП является разновидностью распределения по значимости, следует, что выражения (3.109)-(3.113) будет справедливы для МЧФ с применением алгоритма ВП. Использование такого подхода для МЧФ с ВП позволяет вычислить апостериорное распределении вероятностей для U_t и V_t эмпирическим путем. Это дает возможность применять алгоритм без предварительного анализа структуры ДБС, а необходимым входными данными для алгоритма будет являться набор свидетельств $E_{1:t+1}$.

3.3. Оптимизация процедур формирования и обработки выборок в гибридных методах вероятностного вывода

В процессе разработки гибридных методов вероятностного вывода, а также выработки подходов к оценке эффективности данных методов особую роль играет

оптимизация процедур формирования выборок. Зачастую классические подходы не всегда могут дать требуемую точность, а процесс генерации выборок становиться достаточно трудоемким. В параграфе 3.2 нами был рассмотрен МЧФ как с применением ВЗ и ВП, так и с применением теоремы РБК. В процессе научного исследования прилагается оптимизация методов вероятностного вывода с целью улучшения их показателей, повышения производительности, а также снижения накладных расходов, связанных с обработкой ДБС со сложной структурой и большим количеством значений, которые могут принимать вершины данной сети.

Вначале рассмотрим оптимизацию классического алгоритма МЧФ с ВЗ. Основная цель оптимизации заключается во включении в него алгоритма МЦМП на завершающем этапе алгоритма, так как вклад различных выборок в апостериорное распределение вероятностей достаточно неоднозначен. Необходимость выработки такого подхода связана с тем, что для достижения требуемого уровня точности алгоритма число сгенерированных выборок должно быть достаточно велико $N \to \infty$. Это накладывает как временные, так и ресурсные ограничения в процессе выполнения процедуры вероятностного вывода. Для преодоления данных ограничений целесообразно разработать гибридный метод для выполнения задач стохастического вывода в ДБС на основе сочетания МЧФ с выборкой по значимости (МЧФиВЗ) и алгоритма Метрополиса-Гастингса (МГ). Наиболее целесообразно использовать алгоритм МГ [140] в процессе повторного формирования выборок. Целью повторного формирования выборок является получение аппроксимации искомого распределения путем вторичной генерации N выборок из распределения по значимости $P(X_{t+1}|X_t, E_{t+1})$. Каждая выборка отбирается в соответствии с распределением по значимости и пропорционально связанному с ней весу W_i^{t+1}. В процессе перегенерации учитываются только выборки с наибольшими весами, остальные отбрасываются. В процессе выполнения алгоритма необходимо определить критерий завершения выполнения алгоритма в виде некоторого порогового значения N_{max}. Процедура формирования выборок может быть представлена в виде совокупности следующих этапов.

На первом этапе осуществляется формирование весов выборок W_i^{t+1}. Принятие или отбрасывание таких весов происходит на основе величины $P_a = \min\{1, W_i^{t+1}/N_{max}\}$. На следующем этапе происходит переназначение весов выборкам S^{t+1}, которые не были отброшены на предыдущем этапе в соответствии со следующим выражением $W_i^{t+1} = \max\{N_{max}, W_i^{t+1}\}$. Использование подхода на основе метода Монте-Карло, в частности МГ, позволяет определить переходное распределение вероятностей $P(X, X_{t+1})$, полученное путем развертывания марковской цепи. С учетом этого, искомое распределение вероятностей, описывающее нахождение переменной в состоянии X_{t+1}, имеет вид

$$P(X_{t+1}) = \sum_X P(X)P(X, X_{t+1}). \tag{3.114}$$

Формирование выборок на этапе выполнения алгоритма МГ происходит из распределения по значимости $Q(X, X_{t+1})$, сходящегося к распределению $P(X)$. В связи с этим, вероятность перехода из состояния X в X_{t+1} может быть получена путем преобразования выражения (2.119) на основе обобщения, приведенного в работах Гастингса

$$\varphi(X, X_{t+1}) = min\left[1, \frac{P(E|X_{t+1})P(X_{t+1})Q(X|X_{t+1})}{P(E|X)P(X)Q(X_{t+1}|X)}\right]. \tag{3.115}$$

Из алгоритма МЧФиВЗ следует, что распределение $Q(X|X_{t+1})$ является симметричным. Следовательно, выражение (3.115) может быть преобразовано в следующий вид [15]:

$$\varphi(X, X_{t+1}) = min\left[1, \frac{P(E|X_{t+1})P(X_{t+1})}{P(E|X)P(X)}\right]. \tag{3.116}$$

Из формулы (3.116) можно определить выражение, соответствующее марковской цепи [63] с переходной вероятностью $P(X_{t+1}|X)$

$$P(X_{t+1}|X) = Q(X_{t+1})min\left[1, \frac{W(X_{t+1})}{W(X)}\right], \tag{3.117}$$

где веса $W(X_{t+1}) = \frac{P(X_{t+1})}{Q(X_{t+1})}$ и $W(X_t) = \frac{P(X)}{Q(X)}$ получены из распределения по значимости $Q(X|X_{t+1})$.

Использование алгоритма МГ применительно к МЧФиВЗ недостаточно эффективно, если использовать МГ в процессе формирования ВЗ, и не дает возможность упростить и оптимизировать процедуру вероятностного вывода на основе МЧФиВЗ. В процессе исследования выявлено, что использование МГ наиболее применимо на этапе повторного формирования выборок, что особенно важно, если по результатам алгоритма МЧФиВЗ получено недостаточно количество согласованных выборок. Применение МГ на этапе повторной генерации дает возможность получить выборки с достаточно высоким разбросом значений, которые в свою очередь, не вносят существенный вклад в апостериорное распределение по всем скрытым переменных $P(X_{t+k}|E_{t+k})$ для момента времени $t+k$. Алгоритм МЧФиВЗ с МГ наиболее применим для ДБС с набором n временных состояний, где требуется генерация достаточно большого числа выборок. Однако даже в классическом алгоритме МЧФиВЗ для получения требуемого уровня точности число выборок должно быть достаточно большим. В таком случае разработанный алгоритм позволяет не только оптимизировать процесс генерации выборок, но и повысить точность формирования таких выборок. В итоге распределение по значимости приближается к искомому апостериорному распределению вероятностей $P(X_{\mathrm{t+1}}|E_{t+1})$. Еще одной важной особенностью использования алгоритма МГ является возможность его применения в рамках реализации процедуры перераспределения весов выборок для временных срезов t и $t+k$, полученных в процессе выполнения МЧФиВЗ.

Для оценки эффективности генерации выборок на каждом шаге алгоритма МЧФиВЗ с применение МГ, а также оценки близости распределения значимости $Q(X_{t+\mathrm{k}})$ и искомого распределения $P(X_{t+\mathrm{k}})$ можно использовать метрику Кульбака-Лейблера (КЛ). Метрика КЛ является достаточно эффективной характеристикой степени схожести двух распределений. В общем случае при определении КЛ, распределения могут быть как дискретными, так и непрерывными. Применительно к рассматриваемому алгоритму, метрика КЛ может быть получена из формулы (2.61) с учетом добавления распределения по значимости $Q(X_{t+\mathrm{k}})$

$$D_{KL}\big(Q(X_{t+k}),P(X_{t+k})\big)=\mathbb{E}_Q\left[ln\frac{Q(X_{t+k})}{P(X_{t+k})}\right]\approx\frac{1}{N}\sum_{i=1}^{N}ln\frac{Q\big(X_{t+k}^i\big)}{P\big(X_{t+k}^i\big)}$$
$$=-\frac{1}{N}\sum_{i=1}^{N}ln\,W\big(X_{t+k}^i\big) \quad (3.118)$$

где N – общее число выборок, сгенерированных для ДБС, развернутой вплоть до среза $t+\mathrm{k}$.

Из выражения (3.118) следует, что при равенстве искомого распределения и распределения по значимости $Q(X_{t+k})=P(X_{t+k})$ значение весов выборок $W\big(X_{t+k}^i\big)$ будет равно 1, а метрика КЛ $D_{KL}\big(Q(X_{t+k}),P(X_{t+k})\big)$ будет равна 0. Также данное выражение может быть переопределено для оценки вероятности нахождения переменных X в состоянии $t+k$ при переходе из состояния $t+k-1$ при поступлении свидетельств E. При этом подразумевается, что метрика КЛ зависит лишь от логарифма модели состояния $\sum_{i=1}^{N}\ln P\big(E_{t+k+1}\big|X_{t+k+1}^i\big)$, соответствующей моменту времени $t+k+1$. Можно определить логарифм по всем весам выборок, генерируемых на этапе выполнения алгоритма фильтрации частиц для среза $t+k$

$$\sum_{i=1}^{N}ln\,W\big(X_{t+k}^i\big)=\sum_{i=1}^{N}ln\,W\big(X_{t+k-1}^i\big)+\sum_{i=1}^{N}ln\,P\big(E_{t+k}\big|X_{t+k}^i\big). \quad (3.119)$$

Из формул, используемых для определения метрики КЛ алгоритма МЧФиВЗ с применением МГ следует, что значение КЛ прямо пропорционально числу выборок N. Если значение метрики КЛ является неотрицательным $D_{KL}\big(Q(X_{\mathrm{t}+k}),P(X_{t+k})\big)\geq 0$, тогда получим нормализованные веса $\widetilde{W}_{t+\mathrm{k}}$. Выражение (3.118) можно привести к следующему обобщенному виду:

$$D_{KL}\big(Q(X_{t+k}),P(X_{t+k})\big)\approx-\frac{1}{N}\sum_{i=1}^{N}ln\,\widetilde{W}\big(X_{t+k}^i\big)=N_{KL}. \quad (3.120)$$

Из выражения (3.120) следует, что значение $N_{KL}=\ln N$ достижимо, когда значения нормализованных весов обратно пропорционально числу сгенерированных выборок $\widetilde{W}_{t+\mathrm{k}}=1/N$. Формула расчета расстояния КЛ (3.120)

может быть использована как в классическом алгоритме МЧФиВЗ, так и в процессе оценки выборок на этапе их повторного формирования на основе алгоритма МГ. Можно получить оценку разброса распределения по всем выборкам относительно искомого распределения, а также оценить какой вклад вносит такое распределение. Для получения оптимальных значений весов $W(X_{t+k})$, соответствующих времени $t + k$, необходимо рассчитать метрику КЛ с учетом распределения по значимости $Q(X)$, формируемого для каждого временного среза вплоть до $t + k$. Наилучшие веса достигаться при максимальной схожести распределений $Q(X_{t+k})$ и $P(X_{t+k})$. Исходя из этого, можно скорректировать процедуру оценки наиболее согласованных выборок в процессе перехода от одного временного среза к другому. Важно отметить, что увеличение числа выборок в алгоритме МЧФиВЗ с МГ в меньшей степени влияет на метрику КЛ по сравнению с классическим алгоритмом МЧФиВЗ. Это связано с тем, что после повторного формирования выборок распределение по значимости будет содержать лишь согласованные выборки, а получить необходимую точность можно даже при достаточно ограниченном числе выборок. Постепенное увеличение числа свидетельств незначительно сказывается на изменении метрики КЛ, так как несмотря на необходимость формирования выборок большего объема, доля выборок, вносящих существенных вклад в формирования искомого распределения, увеличивается незначительно.

Далее рассмотрим различные подходы к оптимизации алгоритма МЧФ с применение теоремы РБК. Как было сказано ранее применение РБК является достаточно эффективным способом для уменьшения числа генерируемых выборок, однако в явном виде не позволяет произвести оценку качества данных выборок. Для этого существует несколько способов. Одним из таких способов является построение минимальных достаточных статистик. Данный подход был впервые предложен Леманом и Шеффе [48]. В свою очередь, в работе предлагается оптимизировать формирования оценок на основе использования теоремы Лемана-Шеффе (ЛШ). Теорема ЛШ позволяет сформулировать критерий получения

наилучшей несмещенной оценки (ННО). В частности, ННО согласно теореме ЛШ могут быть получены из теоремы РБК.

Для определения теоремы ЛШ необходимо ввести некоторые формулировки и условия существования достаточных статистик и несмещенных оценок. Несмещенной оценкой для функции $\tau(\theta)$ будет являться функция $\hat{\theta}(\mathrm{X})$, если справедливо следующее выражение [42,70]:

$$\mathbb{E}\hat{\theta}(X) = \tau(\theta). \tag{3.121}$$

Можно рассматривать верхнюю $\hat{\theta}(X)^+$ и нижнюю $\hat{\theta}(X)^-$ оценки для $\tau(\theta)$, для которых выполняются неравенства для их математических ожиданий [147]

$$\begin{aligned} \mathbb{E}\hat{\theta}(X)^+ \geq \tau(\theta), \\ \mathbb{E}\hat{\theta}(X)^- \leq \tau(\theta). \end{aligned} \tag{3.122}$$

В некоторых простейших случаях в качестве несмещенной оценки может выступать линейная форма $\xi = \mathrm{c}_1 X_1 + \mathrm{c}_2 X_2 + \cdots + \mathrm{c}_n X_n$, где $\sum_{i=1}^{n} c_i = 1$. На практике получается достаточно большое разнообразие несмещенных оценок для θ. Для сокращения числа несмещенных оценок целесообразно использовать только те оценки, которые можно выразить через соответствующие им достаточные статистики.

Один из критериев существования достаточных статистик, впервые предложенный Фишером, основывается на определении функции правдоподобия. Согласно Фишеру функцию правдоподобия можно представить в виде произведения вероятностей для каждого элемента выборки X_n

$$\begin{aligned} L(\theta|X) &= P(X_1 = x_1) \times \ldots \times P(X_n = x_n) \\ &= P(X_1 = x_1, .., X_n = x_n). \end{aligned} \tag{3.123}$$

Впервые описание критерия существования достаточных статистик было определено в теореме факторизации Неймана-Фишера [104]. Основная сущность данной теоремы заключается в том, что статистика $T(X)$ будет являться достаточной, если для функции правдоподобия $L(X) = \sum_{i=1}^{n} L(\theta|X_i)$ будет справедливо следующее выражение:

$$L(X|\theta) = g(X|\theta) = \varphi(T(X), \theta)h(X), \tag{3.124}$$

где функции $\varphi \geq 0$ и $h \geq 0$ зависят от соответствующих параметров, функция $\varphi(T(X), \theta)$ зависит от θ, а $h(X)$ только от X.

Выражение (3.124) является критерием достаточности (факторизации). Если имеет место равенство (3.124), то можно показать, что статистика $T(X)$ будет являться достаточной для параметра θ. Определим распределение вероятностей для произвольного θ при фиксированных значениях $x \in X$ и $T(x) = t \in Y$

$$P(X = x|T(X) = T(x)) = \frac{P\big(X = x, T(X) = T(x)\big)}{P\big(T(X) = T(x)\big)}, \tag{3.125}$$

$$P(X = x|T(X) = T(x)) = \frac{P(X = x)}{P\big(T(X) = T(x)\big)} = \frac{L(X)}{\sum_{y:T(y)=T(x)} f(y)} = \\ = \frac{\varphi(T(x), \theta)h(x)}{\sum_{y:T(y)=T(x)} \varphi(T(y), \theta)h(y)} = \frac{h(x)}{\sum_{y:T(y)=T(x)} h(y)}. \tag{3.126}$$

Из формулы (3.126) получим, что вероятностное распределение $P(X = x|T(X) = T(x))$ не зависит от θ. В таком случае искомое выражение имеет следующий вид:

$$P_\theta(X = x) = f_\theta(x) = P_\theta\big(X = x, T(X) = T(x)\big) = h(x)P\big(T(X) = T(x)\big). \tag{3.127}$$

Условное распределение $P\big(T(X) = T(x)\big) = \varphi(T(X), \theta)$ имеет зависимость только от статистики $T(x)$ и θ. В таком случае, если имеет место выражение (3.124), статистика $T(X)$ будет являться достаточной. Если же $T(X) \neq T(x)$, условная вероятность $P\big(T(X) = T(x)\big)$ будет равна нулю.

Из теоремы Неймана-Фишера вытекает следствие, устанавливающее связь между достаточной статистикой $T(X)$ и оценкой максимального правдоподобия θ^* (ОМП). Если $T(X)$ является достаточной статистикой для θ, то оценка ОМП, будет зависеть только от $T(X)$. В таком случае справедливо следующее утверждение. Если достаточная статистика $T(X)$ будет минимальной [133] и сохраняется условие достаточности, то последующее сокращение данных относительно $T(X)$ не даст никакого результата. Следовательно, все оставшиеся данные выборки можно считать случайными и никак не относящимися к θ. Из этого следствия получаем характерное соотношения для применения достаточных статистик в рамках алгоритма ВП для реализации вероятностного вывода в ДБС.

Наряду с критерием факторизации Неймана-Фишера, существует еще один критерий достаточности, который можно сформулировать следующим образом. Статистика T будет являться достаточной для θ только в том случае, если для всякого априорного распределения оценки параметра θ, апостериорное распределение будет зависеть от X только через статистику $T(X)$. Данный критерий может быть достаточно просто получен из формулы Байеса. Для этого определим статистику $T(X)$ с плотностью распределения $Q(\theta)$ относительно некоторой меры ξ. В таком случае апостериорная плотность $Q(\theta|X)$ относительно меры ξ может быть записана в следующем виде [49]:

$$Q(\theta|X) = \frac{f_\theta(X)Q(\theta)}{\int f_u(X)Q(u)\,\xi(du)} = \frac{\varphi(T(X),\theta)Q(\theta)}{\int \varphi(T(X),u)Q(u)\,\xi(du)}. \tag{3.128}$$

Если выбрать априорное распределение таким, чтобы плотность $Q(\theta) > 0$, тогда $f_\theta(X)$ будет иметь следующие значения для всех θ:

$$\begin{gathered} f_\theta(X) = \frac{Q(\theta|X)f(X)}{Q(\theta)}, \\ f(X) = \int f_u(X)Q(u)\xi(du). \end{gathered} \tag{3.129}$$

Если положить, что $Q(\theta|X) = g\big(\theta, T(X)\big)$ и $\varphi(T,\theta) = \frac{g(\theta,T)}{Q(\theta)}, h(X) = f(X)$, то выражения (3.128) и (3.129) могут быть преобразованы к формуле Неймана-Фишера (3.124).

Для получения ННО необходимо ввести понятие полной статистики. Статистика $T(X)$ будет являться полной (ограниченно полной), если для любой числовой статистики $f\big(T(X)\big)$ из $\mathbb{E}\left(f\big(T(X)\big)\right) = 0$, получаем $f\big(T(X)\big) = 0$. Тогда критерий полноты статистики можно записать в следующем виде:

$$P\big(f\big(T(X)\big) = 0\big) = 1, \forall\theta \in \Theta. \tag{3.130}$$

Для получения ННО и оптимизации алгоритма МЧФ, целесообразно использовать теорему ЛШ. Согласно утверждению Лемана [154,155] наилучшей будет являться оценка, которая зависит только от достаточной статистики $T = T(X)$. В свою очередь, в работе Колмогорова приведено доказательство, что оценка $\varphi = \varphi(X)$ будет являться несмещенной для функции $\tau(\theta)$. Из этого следует, что

любая произвольная оценка $\varphi^* = \varphi^*\big(T(X)\big) = \mathbb{E}\big(\varphi(X)\big|T(X)\big)$ рассматриваемой достаточной статистики $T(X)$ будет являться несмещенной в том случае, если справедливы соотношения для математических ожиданий и дисперсий оценок φ^* и φ. Для этого воспользуемся формулой полного математического ожидания

$$\mathbb{E}\big(\mathbb{E}(\varphi(X)|T(X))\big) = \mathbb{E}\varphi(X). \tag{3.131}$$

Тогда результирующие выражения для φ^* и φ имеют вид

$$\begin{gathered}\mathbb{E}\varphi^*\big(T(X)\big) = \mathbb{E}\big(\mathbb{E}(\varphi(X)|T(X))\big) = \mathbb{E}\varphi(X) = \tau(\theta),\\ \mathbb{D}\varphi^*\big(T(X)\big) \le \mathbb{D}\varphi(X).\end{gathered} \tag{3.132}$$

В таком случае оценка φ^* будет являться единственной эффективной оценкой для φ.

Неравенство для дисперсий φ^* и φ может быть получено путем расчета дисперсии для оценки φ

$$\begin{aligned}\mathbb{D}\varphi(X) &= \mathbb{E}\left(\varphi(X) - \varphi^*\big(T(X)\big) + \varphi^*\big(T(X)\big) - \tau(\theta)\right)^2\\ &= \mathbb{E}\left(\varphi(X) - \varphi^*\big(T(X)\big)\right)^2\\ &+ 2\mathbb{E}\left(\varphi(X) - \varphi^*\big(T(X)\big)\right)\mathbb{E}\left(\varphi^*\big(T(X)\big) - \tau(\theta)\right)\\ &+ \mathbb{E}\left(\varphi^*\big(T(X)\big) - \tau(\theta)\right)^2\end{aligned} \tag{3.133}$$

С учетом того, что разность $\varphi^*\big(T(X)\big) - \tau(\theta)$ зависит лишь от статистики T, то согласно формулам (3.131), (3.132) и (3.133) имеем

$$\begin{aligned}&\mathbb{E}\left(\varphi(X) - \varphi^*\big(T(X)\big)\right)\mathbb{E}\big(\varphi^*\big(T(X)\big) - \tau(\theta)\big) =\\ &= \mathbb{E}\left(\varphi^*\big(T(X)\big)\left(\mathbb{E}\big(\varphi(X)|T(X)\big) - \varphi^*\big(T(X)\big)\right)\right) = 0,\end{aligned} \tag{3.134}$$

$$\mathbb{D}\varphi(X) = \mathbb{E}\left(\varphi(X) - \varphi^*\big(T(X)\big)\right)^2, \mathbb{D}\big(\varphi(X)\big) \le \mathbb{D}\left(\varphi^*\big(T(X)\big)\right).$$

Следовательно, критерий полноты оценки $\varphi^*\big(T(X)\big)$ будет иметь вид

$$\begin{gathered}P\left(\varphi^*\big(T(X)\big) = \varphi(X)\right) = 1,\\ \mathbb{E}\varphi^*\big(T(X)\big) = 0.\end{gathered} \tag{3.135}$$

В таком случае теперь можно сформулировать теорему ЛШ. Сущность теоремы ЛШ заключается в том, что если статистика $T(X)$ является полной достаточной статистикой для параметра $\theta \in \Theta$ и $\varphi(X)$ несмещенная оценка для некоторой функции $\tau(\theta)$, определенной на множестве Θ, то существует оценочная функция с минимальной дисперсией оценки $\varphi^*\big(T(X)\big)$. Функция $\tau(\theta)$ будем являться допускающей несмещенную оценку. В таком случае оценка $\varphi^*\big(T(X)\big)$ будет являться несмещенной оценкой ожидаемого значения с наименьшей дисперсией. Обобщая теорему ЛШ на основе формул (3.132) и (3.135) сформулируем условие получения ННО в следующем виде:

$$\begin{gathered}\mathbb{E}\varphi^*\big(T(X)\big) = \mathbb{E}\varphi(X) = \tau(\theta), \forall\theta \in \Theta,\\ \mathbb{D}\varphi^*\big(T(X)\big) \le \mathbb{D}\varphi(X), \forall\theta \in \Theta,\\ P\left(\varphi^*\big(T(X)\big) = \varphi(X)\right) = 1, \forall\theta \in \Theta,\\ P\left(\varphi^*\big(T(X)\big) = \varphi'^*\big(T(X)\big)\right) = 1, \forall\theta \in \Theta.\end{gathered} \tag{3.136}$$

где $\varphi'^*\big(T(X)\big)$ – произвольная несмещенная оценка для $\tau(\theta)$, удовлетворяющая условию (3.132).

Исходя из определения теоремы ЛШ, можно сформулировать следующие критерии получения ННО.

Первый критерий связан с несмещенностью оценок и заключается в том, что смещения оценок $\varphi^*\big(T(X)\big)$ и $\varphi(X)$ будет равны, если обе оценки будут несмещенными.

Второй критерий связан с неравенством дисперсий, а именно, что дисперсия для $\varphi^*\big(T(X)\big)$ не будет превышать значение дисперсии для $\varphi(X)$.

Последний критерий связан с уравнением Колмогорова [42], а именно, что оценка $\varphi^*\big(T(X)\big)$ будет являться ННО, если она будет зависеть исключительно от достаточной статистики $T(X)$.

Из приведенных критериев получения ННО особое внимание стоит уделить критерию, получаемому из уравнения Колмогорова. Из этого следует, что если достаточная статистика $T(X)$ будет минимальной, то и соответствующая оценка

будет являться ННО. Это говорит о том, что в множестве исходных оценок θ' могут присутствовать несогласованные оценки. Такие оценки будут являться несостоятельными и не участвовать в формировании искомого распределения вероятностей. В связи с этим рассмотрим особенности нахождения минимальных достаточных статистик. В общем случае достаточная статистика $T(X)$ будет являться минимальной достаточной статистикой (МДС), если она позволяет получить максимально возможную редукцию рассматриваемых данных по отношению к другим достаточным статистикам $T' = (T_1, T_2, \dots, T_n)$. Для любой достаточной статистики T' существует некоторая функция f такая, что $T' = f(T)$. Из свойств полной достаточной статистики следует, что любая полная достаточная статистика (ПДС) будет являться МДС [166]. Данное утверждение было впервые описано в работах Лемана, Шеффе и Бахадура [5]. Однако в общем случае достаточно сложно получить ПДС, так как в любой статистике может присутствовать большое число побочной информации. В таком случае для любой статистики может рассматриваться понятие подчиненной статистики. Всякая статистика $\varphi(X)$ будет являться подчиненной в том случае, если она имеет распределение вероятностей независящее от θ. Всякая статистика $\varphi(X)$ будет являться подчиненной первого рода, если она имеет постоянное математическое ожидание $\mathbb{E}\big(\varphi(X)\big) = const$ и не зависит от θ. Для получения МДС необходимо, чтобы функция $f(T)$ была либо подчиненной статистикой, либо подчиненной статистикой первого рода. Рассматривая случай, когда имеет место ПДС, а МДС $\varphi(X)$ являться полной достаточной статистикой, можно определить факт независимости любой ПДС от статистики $T(X)$. Данное условие можно получить из теоремы Басу. Сущность данной теоремы заключается в том, что если мы имеем подчиненную достаточную статистику $\varphi(X)$ для некоторого семейства распределения $P = \{P_\theta, \theta \in \Theta\}$, то любая подчиненная статистика $\varphi(X)$ не будет зависеть от $T(X)$.

Альтернативный подход по нахождению МДС был предложен в работах Лемана и Шеффе [47]. Данный подход выделяется наглядностью и позволяет оптимизировать нахождения МДС. Введем следующее утверждение. Пусть $T'(X)$

произвольная достаточная статистика, $T(X)$ является достаточной статистикой, принимающей значения из Y, а функция $f\left(T'(X)\right)$ является взаимно-однозначной с значениями в Y, то используя критерий Неймана-Фишера (3.124), можно установить факт, что статистика $T(X) = f\left(T'(X)\right)$ будет являться минимальной достаточной статистикой. Для этого рассмотрим разбиение пространства на классы эквивалентности. Пусть $X \in D(X)$ – класс эквивалентности. Тогда разбиение на D достаточно, когда справедлива теорема факторизации (3.124)

$$f(X) = \varphi(T(\mathrm{X})|\theta)h(X), \tag{3.137}$$

где $\varphi(T(X)|\theta) = \varphi(T(X_0)|\theta)$ постоянно при $X \in D(X_0)$,

Возьмем разбиение, в котором $X_0 \in D(X)$ и отношение $f(X)/f(X_0)$ не зависит от параметра θ

$$\frac{f(X)}{f(X_0)} = h(X, X_0). \tag{3.138}$$

Из выражения (3.138) следует, что $D(X_0) = D(X_1) = D(X_2)$, если $X_1 \in D(X_0)$ и $X_2 \in D(X_0)$. Докажем, что $T(\mathrm{X})$ минимальная достаточная статистика. Выберем произвольную точку из разбиения $X_0 \in D(X)$, тогда отношение $f(X)/f(X_0)$ с учетом факторизации (3.137) для произвольной достаточной статистики $T'(\mathrm{X})$ будет иметь следующий вид:

$$\frac{f(X)}{f(X_0)} = \frac{\varphi(T'(\mathrm{X})|\theta)h(X)}{\varphi(T'(\mathrm{X}_0)|\theta)h(X_0)} = \frac{h(X)}{h(X_0)} \tag{3.139}$$

С учетом того, что полученное отношение не зависит от параметра θ, то $T'(\mathrm{X}) = T'(\mathrm{X}_0)$. Так как $T(\mathrm{X})$ является функцией от статистики $T'(\mathrm{X})$, то $T(\mathrm{X}) = T(\mathrm{X}_0)$. Статистика $T(\mathrm{X})$ минимальная достаточная статистика. При сравнении двух достаточных статистик $T'(\mathrm{X})$ и $T''(\mathrm{X})$ наилучшей будет считаться та, которая не вносит дополнительной информации о каком-либо параметре при наличии значений другой достаточной статистики.

В процессе применения теоремы ЛШ возникает вопрос, связанный с рассмотрением меры эффективности анализируемых оценок. Эффективная оценка подразумевает наименьший разброс значений относительно истинного значения самого параметра θ. В связи с этим необходимо определить нижнюю границу

разброса для оценок θ. Такую границу, в частности, можно получить на основе неравенства информации Рао-Крамера-Фреше (РКФ) [152]. С учетом того, что на основе теоремы ЛШ мы получаем ННО, то степень случайного разброса оценки θ^* относительно параметра θ будет совпадать с дисперсией $\mathbb{D}(\theta^*)$. В таком случае выражение для определения эффективности рассматриваемой оценки имеет следующий вид [119]:

$$\mathbb{D}(\theta^*) = \mathbb{E}(\theta - \theta^*)^2 = \mathbb{E}(\theta^* - \theta)^2. \quad (3.140)$$

Запишем формулу математического ожидания для дискретного случая оценки θ^* параметра θ

$$\mathbb{E}(\theta^*) = \sum_{X_n} \theta^*(X_1, X_2, \dots, X_n) L(X_1, X_2, \dots, X_n) = \theta + b(\theta), \quad (3.141)$$

где $b(\theta)$ – смещение оценки.

С учетом того, что теорема ЛШ дает нам несмещенные оценки, то второе слагаемое обращается в ноль $b(\theta) = 0$. Следовательно, его нет необходимости учитывать в процессе определения математического ожидания оценки θ^*. Для формирования условия эффективности оценки необходимо оценить плотность распределения выборки. Для этого зададим уравнение информации Фишера [215] для множества наблюдений $X = (X_1, X_2, \dots, X_n)$ через соответствующую функцию правдоподобия $L(\theta|X)$

$$I(\theta|X) = \mathbb{E}\left[\left(\frac{dL(\theta|X)}{d\theta}\right)^2\right] = \sum_X \left(\frac{dL(\theta|X)}{d\theta}\right)^2 L(\theta|X). \quad (3.142)$$

Учитывая, что наблюдения $X_1, X_2, \dots, X_n$ являются независимыми и имеют одинаковое распределение, то (3.142) можно записать, используя плотность распределения $p_i(X|\theta) = P\{\xi = X_i|\theta\}$

$$I(\theta|X) = n \sum_i \left(\frac{d\ln(p_i(X|\theta))}{d\theta}\right)^2 p_i(X|\theta) = n\, I(\theta|X). \quad (3.143)$$

Неравенство информации для дискретного случая можно записать с учетом выражения для информации Фишера

$$\mathbb{D}(\theta^*) = \mathbb{E}(\theta^* - \theta)^2 \geq \frac{\left(1 + \frac{db(\theta)}{d\theta}\right)^2}{n\sum_i \left(\frac{d\ln(p_i(X|\theta))}{d\theta}\right)^2 p_i(X|\theta)}. \quad (3.144)$$

С учетом того, что теорема ЛШ дает нам ННО неравенство (3.144) можно упростить. Получим

$$\mathbb{D}(\theta^*) \geq \frac{1}{nI(\theta|X)}. \quad (3.145)$$

При этом неравенство РКФ для n мерного случая параметра $\Theta = (\theta_1, \theta_2, \dots, \theta_n)$ с несмещенной оценкой Θ^* можно записать с использованием матрицы ковариаций $cov(\Theta^*)$

$$cov(\Theta^*) \geq \frac{1}{n} I^{-1}(\theta|X), \quad (3.146)$$

где $I^{-1}(\theta|X)$ – обратная информационная матрица.

На основе неравенства РКФ, как для одномерного, так и для многомерного случая необходимо определить некоторую меру эффективности оценок θ^* и Θ^*. Данная мера представляет собой минимально возможную дисперсию $\mathbb{D}' = \min_{\theta^*} \mathbb{D}(\theta^*)$. Значение дисперсии $\mathbb{D}'$ для получения эффективной оценки вычисляется за счет превращения неравенства информации (3.145) в равенство. В таком случае дисперсию наиболее эффективной оценки можно выразить через матрицу обратную информационной матрицы:

$$\mathbb{D}'(\theta^*) = \frac{1}{n} I^{-1}(\theta|X). \quad (3.147)$$

Определение дисперсий эффективных оценок может быть использовано в теореме РБК. Рассматривая применение теоремы ЛШ к алгоритму МЧФ можно сформулировать следующее утверждение. Исходя из теоремы ЛШ, критерий получения ННО, применительно к МЧФ, может быть сформулирован на основе определения минимальной достаточной статистики $T(X)$ и выражения, соответствующего теореме РБК.

В процессе вычисления ННО, при каждом формировании новой выборки на основе алгоритма МЧФ, производится проверка несмещенной оценки,

соответствующей текущей выборке. Для реализации данной проверки используются веса формируемых выборок. В процессе определения весов выборок производится определение их правдоподобия $L(\theta|X)$ и формирование матрицы Фишера $I(\theta|X)$. Применяя неравенство информации, определяются оценки весов выборок $W(V_{0:t+1}, U_{0:t+1})$ с наименьшей дисперсией. Учитывая выражение (3.113) для дисперсий и условного математического ожидания для весов выборок $W(V_{0:t+1}, U_{0:t+1})$ и $W\left(V_{0:t+1}^{i}\right)$, получим следующие неравенства, удовлетворяющие теореме РБК [170]:

$$\begin{gathered}
\mathbb{D}_{P(V_{0:t+1}|E_{1:t+1})}\hat{A}_{t+1}^{1} \leq \mathbb{D}_{P(V_{0:t+1},U_{0:t+1}|E_{1:t+1})}\hat{A}_{t+1}^{2} \\
\mathbb{D}_{P(V_{0:t+1}|E_{1:t+1})}\hat{B}_{t+1}^{1} \leq \mathbb{D}_{P(V_{0:t+1},U_{0:t+1}|E_{1:t+1})}\hat{B}_{t+1}^{2} \\
\mathbb{E}_{P(V_{0:t+1}|E_{1:t+1})}\left(\left(B_1 W(V_{0:t+1},U_{0:t+1})\right)^2\right) \\
\leq \mathbb{E}_{P(W(V_{0:t+1},U_{0:t+1})|E_{1:t+1})}\left(\left(B_2 W(V_{0:t+1})\right)^2\right), \\
B_1 = f_{t+1}(V_{0:t+1},U_{0:t+1}) - I(f_{t+1}), \\
B_2 = E_{P(U_{0:t+1}|E_{1:t+1},V_{0:t+1})}\left(f_{t+1}(V_{0:t+1},U_{0:t+1})\right) - I(f_{t+1}),
\end{gathered} \tag{3.148}$$

где $I(f_{t+1})$ – ожидаемое значение функции f_{t+1}, соответствующей текущему срезу $t+1$, оценки $\hat{A}_{t+1}^{1}, \hat{A}_{t+1}^{2}$ и $\hat{B}_{t+1}^{1}, \hat{B}_{t+1}^{2}$ вычисляются в соответствии с (3.112).

Из выражений (3.148), характеризующих применение теоремы РБК и ЛШ к алгоритму МЧФ с ВП следует, что оценка $\hat{B}_{t+1}^{1}$ будет являться наименьшей несмещенной оценкой. Так как оценка $\hat{B}_{t+1}^{1}$ зависит от весов $W(V_{0:t+1}, U_{0:t+1})$ и $W\left(V_{0:t+1}^{i}\right)$, искомое распределение по всем скрытым переменных вероятностей $P(X_{t+1}|E_{1:t+1})$ будет включать лишь выборки для которых существует ННО [165].

Важно отметить, что теорема ЛШ наиболее применима к МЧФ с ВП, так как для получения неравенства требуется определить функцию правдоподобия для каждой из выборок, генерируемых в процессе выполнения алгоритма МЧФ. Нет необходимости повторной генерации выборок методом Монте-Карло, так как в результате выполнения алгоритма МЧФ с применением теоремы ЛШ получаем оптимальные выборок с точки зрения их правдоподобия, а искомое распределение вероятностей $P(V_{0:t+1}|E_{1:t+1})$ является наиболее согласованным со

свидетельствами $E_{1:t+1}$. Это позволяет сократить число выборок, необходимых для реализации процедуры вероятностного вывода, без потери качества, так как рассматриваемые выборки не вносят вклад в распределение $P(V_{0:t+1}|E_{1:t+1})$ и обладают минимальными весами. При использовании ЛШ в рамках алгоритма МЧФ ВЗ не требуется повторная генерация выборок для повышения доли согласованных со свидетельствами выборок. В свою очередь, алгоритм МЧФ ВП с использованием теоремы ЛШ применяется исключительно к уже имеющейся ДБС и не может упростить топологию данной сети. Существует ряд основных подходов, направленных на снижение сложности структуры ДБС. По аналогии с статическими БС, наиболее эффективными алгоритмами для решения такого рода задач являются алгоритмы кластеризации, в частности дерево сочленений (ДС). Ключевой особенность ДС, применительно к семантике ДБС, является возможность кластеризации смежных узлов в соседних временных срезах, что позволяет упростить построение модели перехода $P(X_{t+1}|X_t)$. Процедура применения ДС для алгоритма МЧФ с применением теоремы ЛШ возможна за счет последовательного развертывания ДБС на k временных срезов. Основные этапы выполнения алгоритма построения ДС представлены нами в параграфе 2.2. В исследовании рассмотрим процедуру адаптации алгоритма ДС для упрощения процедуры вероятностного вывода в ДБС на основе МЧФ. С учетом того, что ДБС имеет транзитивные связи, то множество клик C_{t+k}^{i} и сепараторов S_i может включать узлы ДБС, располагаемые в разных временных срезах и иметь условные связи, задаваемые в рамках модели перехода $P(X_{t+1}|X_t)$. Основная идея предлагаемого подхода заключается в построении сильно связных деревьев отдельно для каждого из временных срезов ДБС, а использование кластеризации позволяет снизить число кликов C_{t+k}^{i} узлов, имеющих временные связи. Кластеры представляют собой объединения вершин, имеющих связи на разных временных срезах, задаваемых с помощью модели перехода. Можно изначально построить ДС для каждого из временных срезов, не учитывая транзитивные связи. После чего произвести построение ДС с учетом узлов, описываемых моделью перехода и

включенных в кластер. Это дает возможность использовать лишь те клики C_{t+k}^{i}, которые непосредственно содержат узлы, имеющие временные связи.

Процедура построения дерева сочленений для ДБС состоит из следующих основных этапов [64].

На первом этапе производим построение ДС для каждого из временных срезов ДБС J_t и J_{t+1} согласно алгоритму построения ДС, представленному в параграфе 2.2, но без непосредственного опроса сети каждого из ДС. В результате определяем все возможные клики, потенциалы и сепараторы для ДС каждого временного среза ДБС, вплоть до рассматриваемого в текущий момент временного среза $t+k$.

На следующем шаге необходимо определить априорное распределение вероятностей для кластеров первого временного среза $P(\Omega_t|\mathrm{E}_{1:t})$. Кластер Ω_t включает все переменные для среза t, имеющие связи в срезе $t+k$. С учетом того, что мы предварительно построили деревья сочленений для каждого временного среза J_t и J_{t+1}требуемое распределение достаточно легко получить из потенциалов для каждой клики C_t^i кластера ДБС в процессе построения ДС для среза $t+1$.

На третьем шаге вычисляем распределение вероятностей для каждого узла ДБС для среза $t+1$ путем перемножения значений из таблиц условных вероятностей на соответствующие потенциалы ψ_{t+1} в дереве сочленений J_{t+1}. Определяем значения свидетельств для каждой клики $\mathrm{C}_{1:t+1}^{i}$ до текущего момента времени $t+1$.

На четвертом шаге выполняем алгоритм Шефера-Шеноя путем распространения свидетельств E_{t+1}, начиная от корневой клики C_{t+1}^{i}. В результате получаем обновление потенциалов ψ_{t+1} на основе выражений (2.37) и (2.39).

На следующем шаге для кластера Ω_{t+1} получаем потенциал $\psi_{t+1}(\Omega_{t+1})$ при наличии свидетельств, полученных до текущего среза $t+1$. Значение потенциала рассчитывается за счет маргинализации потенциалов для каждой клики C_{t+k}^{i}, соответствующей кластеру Ω_{t+1}.

На шестом шаге производим обновление потенциалов $\psi_{t+1}(S_{t+1})$, соответствующих клике C_t^i за счет поглощения потенциалов клики C_{t+1}^i.

На завершающем этапе повторяем алгоритм Шефера-Шеноя и тиражирование свидетельств E_t между всеми кликами, начиная с клики C_t^i. В результате получаем искомое распределение вероятностей по всем скрытым переменным X_{t+1} [77,78]

$$P(X_{t+1}|\mathrm{E}_{1:t+1}) = \frac{\prod_{\Omega_{t+1}} \psi_{t+1}(\Omega_{t+1})}{\prod_{S_{t+1}} \psi_{t+1}(S_{t+1})} = \frac{\prod_{\Omega_{t+1}} P(\Omega_{t+1}|\mathrm{E}_{1:t+1})}{\prod_{S_{t+1}} P(\mathrm{S}_{t+1}|\mathrm{E}_{1:t+1})},$$

где S_{t+1} – множество сепараторов дерева сочленения J_{t+1} с кластером Ω_{t+1}.

Отличительной особенностью алгоритма построения ДС на основе кластеризации (ДСК) является то, что снижается общее число клик каждого среза ДБС, а процедура распространения свидетельств становится более простой и понятной. Сравнивая алгоритм ДСК с алгоритмом Боена-Коллера [16], описанным в параграфе 3.2, также использующим дерево сочленений в качестве основного инструмента построения логики алгоритма, в ДСК не используются интерфейсы. Это обусловлено тем, что в процессе формирования J_{t+1} в интерфейс могут попадать узлы, не только описанные в модели перехода, но и в модели восприятия. Данный факт значительно усложняет обработку и построение результирующего дерева сочленений для ДБС с большим числом узлов, так как возможен избыточный пересчет условных вероятностей для каждого из узлов $t+k$ временных срезов. В таком случае общая сложность алгоритма распространения будет пропорциональна общему числу интерфейсных узлов, входящих в состав ДБС, а таблицы сепараторов необходимо будет храниться для каждого среза ДБС. Процедура упрощения алгоритма построения ДС для ДБС на основе кластеризации, позволяет адаптировать данный алгоритм к любому из представленных алгоритмов вероятностного вывода и, в частности, алгоритмам на основе МЧФ с применением МГ и ЛШ. В таком случае можно реализовать хороший гибридный подход, позволяющий сначала упростить топологию ДБС, а затем использовать МЧФ для реализации процедуры вероятностного вывода. Использование такого подхода не накладывает никаких ограничений на

применяемые алгоритмы вероятностного вывода, так как в общем случае алгоритм ДСК используется для упрощения структуры ДБС. Единственное что стоит отметить в представленном гибридном методе, является его применимость к дискретным ДБС, так как ДСК недостаточно приспособлен для работы с непрерывными переменными. Поэтому рамки использования разработанного алгоритма будут ограничиваться лишь ДБС с дискретными переменными. Если существует необходимость использования непрерывных ДБС, тогда целесообразно использовать отдельно только алгоритма МЧФ с применением МГ или ЛШ, так как в последнем не требуется повторная генерация выборок. Использование такого подхода позволяет адаптировать разработанные алгоритмы к решению различных задач вероятностного вывода, как в дискретных, так и для непрерывных ДБС.

3.4. Выводы

В третьей главе работы:

1. Представлены задачи вероятностного вывода для динамических байесовских сетей организации процесса тестирования веб-приложений методом фаззинга и классические методы их решения.

2. Описаны гибридные стохастические методы вероятностного вывода на основе многочастичного фильтра.

3. Описан разработанный в рамках исследования метод вероятностного вывода для динамических байесовских сетей, базирующийся на многочастичном фильтре, отличающийся применением алгоритма Метрополиса-Гастингса на этапе повторной генерации выборок и теории достаточных статистик для сокращения количества выборок и оптимизации расчета весов генерируемых выборок.

4. Описан разработанный в рамках исследования метод повышения эффективности процедуры вероятностного вывода на основе многочастичного фильтра с применением теоремы Лемана и Шеффе, отличающийся использованием технологии снижения сложности структуры динамических байесовских сетей на основе построения деревьев сочленений.

Глава 4. Оптимизация процессов синхронизации для распределенных вычислительных систем решения задач обучения и вероятностного вывода в динамических байесовских сетях

4.1. Параллельная обработка данных на основе распределенных вычислений

Создание и применение параллельных вычислений для решения задач машинного обучения и вероятностного вывода является значимыми и актуальными направлениями исследования. Рост производительности современных вычислительных систем позволяет в значительной степени снизить временные затраты, необходимые для проведения того или иного вычисления. На сегодняшний день существует достаточно много технологических подходов, позволяющих использовать концепцию параллельных вычислений в области машинного обучения. Наибольший интерес представляют кластерные (облачные), Grid-технологии и сети добровольных вычислений. Представителями данных технологий являются программно-аппаратные платформы NVIDIA Cuda, Hadoop, Spark, Boinc и т.д. Платформа Cuda использует в качестве программной части графические видеопроцессоры и позволяет распараллеливать операции между несколькими видеокартами, объединяющими несколько видеопроцессоров. Hadoop и Spark являются кластерным решением и строятся по топологии «звезда», где есть явно заданный управляющий узел (мастер-узел) и ряд вспомогательных узлов (рабочий-узел). Boinc представляет собой платформу для распределенных вычислений. Архитектура Boinc, так же как Hadoop и Spark строиться по топологии «звезда». Boinc используется для координации, фрагментирования и распределения задач клиентам для последующего выполнения. В процессе формирования заданий платформа оптимизирована как для использования ресурсов центрального процессора, так и видеопроцессоров. Boinc является кроссплатформенной средой, однако не предназначена для обработки больших объемов данных. Она также имеет ряд ограничений, связанных с объемом дискового пространства компьютера, на котором производятся вычисления.

В процессе моделирования и разработки параллельных программ необходимо учитывать показатели производительности параллельных систем. Основными показателями производительности параллельных систем и алгоритмов являются эффективность и ускорение. Ускорение представляет собой метрику, определяющую во сколько раз снизиться общее время выполнения алгоритма при использовании распараллеливания вычислений

$$S = \frac{T_s}{T_n}, \tag{4.1}$$

где T_s и T_n – время выполнения на одном процессоре и на системе из n параллельных процессоров.

Эффективность отражает максимально полное использование ресурсов параллельной системы во время проведения вычислений

$$E = \frac{S}{n}. \tag{4.2}$$

Согласно закону Амдала, значение ускорения S может быть приведено к следующему виду

$$S = \frac{Pt}{\left(P_s + \frac{P_n}{n}\right)t} = \frac{1}{\alpha + \frac{1-\alpha'}{n}}, \tag{4.3}$$

где $\alpha = \frac{P_n}{n} + P_s$ – доля операций, которые должны быть выполнены с использованием n параллельных процессоров, $P = P_s + P_n$ – общее число операций, которое необходимо выполнить, P_s – число скалярных операций, P_n – число параллельных операций, t -время выполнения одной операции.

Из формулы (4.3) следует, что при увеличении $n \to \infty$ значение ускорения параллельной системы S будет стремиться к значению $1/a$. Для построения многопроцессорных систем с индивидуальной памятью могут быть использованы связующие сети, в частности Ethernet. При использовании сетевого взаимодействия между многопроцессорными системами необходимо учитывать задержки на передачу сообщений. В связи с этим можно переписать закон для расчета ускорения параллельной системы (4.3) в сетевой форме [101]

$$S = \frac{Pt}{\left(P_s + \frac{P_n}{n}\right)t + P_e \times t_e} = \frac{1}{\alpha + \frac{1-\alpha}{n} + \frac{P_e t_e}{Pt}} = \frac{1}{\alpha + \frac{1-\alpha}{n} + d},$$
$$d = \frac{P_e t_e}{Pt}, \tag{4.4}$$

где P_e – число сообщений, переданных по сети между узлами многопроцессорной системы, t_e – время, затраченное на передачу одного сообщения, d – коэффициент временной задержки, связанный с передачей всех сообщений по сети.

Для оценки максимального ускорения параллельного алгоритма можно использовать закон Густавсона-Барсиса. Для этого необходимо определить соотношение временных затрат для последовательных и параллельных фрагментов алгоритма [217]

$$\sigma = \frac{t_s}{t_s + \frac{t_n}{n}}, \tag{4.5}$$

где t_s и t_n – время, затраченное для обработки последовательной и параллельной части алгоритма, n – число процессоров, используемых для вычислений. Время выполнения алгоритма в целом на одном и n – процессорах описывается на основе следующего выражения:

$$T_s = t_s + t_n,$$
$$T_n = t_s + \frac{t_n}{n}. \tag{4.6}$$

Выражение (4.5) для t_s и t_n можно преобразовать к следующему виду [72]:

$$t_s = \sigma\left(t_s + \frac{t_n}{n}\right),$$
$$t_n = (1-\sigma)n\left(t_s + \frac{t_n}{n}\right). \tag{4.7}$$

Используя формулы (4.6) и (4.7), выражение (4.1), описывающее расчет значения ускорения параллельного алгоритма [120], преобразуется к следующей форме:

$$S = \frac{T_s}{T_n} = \frac{t_s + t_n}{t_s + \frac{t_n}{n} =} = \sigma + n(1-\sigma) = n + \sigma(1-n). \tag{4.8}$$

Оценка ускорения алгоритма (4.8) описывает закон Густавсона-Барсиса. Применение данных вычислений необходимо для оценки степени параллелизма алгоритма, выявления его функциональных блоков, которые могут выполняться независимо друг от друга, а также для определения общего прироста производительности за счет использования параллелизма.

На сегодняшний день существуют два основных способа распараллеливания: параллелизм задач и данных. Распараллеливание задач связано с разделение алгоритма на несколько самостоятельных блоков, каждый из которых использует ресурсы процессора для выполнения вычислительной операции. Особенностью данного параллелизма является использования общих данных для обмена сообщениями между задачами. Второй подход, связанный с параллелизмом данных, заключается в том, что одна операция может использовать целый набор данных. Исходя из этого, можно обрабатывать отдельные фрагменты данных независимо. При этом, если существуют общие результирующие данные, формируемые в процессе параллельной обработки, то такие процессы нуждаются во взаимной блокировке с целью предотвращения возникновения коллизий внутри потоков, обрабатывающих данных.

Среди программно-аппаратных платформ общего назначения выделяют системы с распределенной общей памятью (РОП). В таких системах объединение узлов осуществляется посредством среды передачи данных, а все адресное пространство определяется путем суммирования всей доступной памяти на каждом из узлов. В качестве среды передачи данных могут выступать сети Ethernet, волоконно-оптические (ВОЛС) и беспроводные сети. РОП-системы [161] обладают высоким уровнем масштабируемости и позволяют объединять в единое адресное пространство неограниченное число физических вычислительных процессоров. Однако масштабируемость накладывает ряд ограничений, связанных с необходимостью создания механизмов координации обмена данными между процессорами для каждого узла, а также сокращения времени, связанного с передачей информации через среду передачи данных. Создание эффективных масштабируемых РОП является необходимым условием для реализации сложных

параллельных систем, предназначенных для решения широкого спектра задач, в том числе задач машинного обучения. Важную роль в процессе физического размещения и развертывания таких систем играет создание облачных платформ обработки и хранения данных. Отличительной особенностью таких платформ является возможность объединения в единую вычислительную сеть мощности нескольких разнородных аппаратных платформ. Сущность построения облачных систем заключается в объединение распределенных вычислительных ресурсов в единую систему с возможностью обеспечения постоянного доступа к ней. Облачные вычисления являются динамически развивающимся направлением в области моделирования сложных стохастических процессов, а также в решении задач машинного обучения и классификации разнородных данных. В настоящее время существует несколько моделей обслуживания, характерных для облачных систем распределенной обработки данных [32,149]:

1. Программное обеспечение как услуга (SaaS). Предоставляется доступ к программному обеспечению (ПО), располагаемому на стороне провайдера и позволяющему производить базовые операции с инфраструктурой (передача, запись данных, запуск заданий).

2. Платформа как услуга (PaaS). Модель, позволяющая устанавливать и использовать необходимое стороннее кластерное ПО и последующее размещение на нем необходимых приложений.

3. Инфраструктура как услуга (IaaS). Сочетает особенности двух вышеперечисленных моделей и позволяет получить полный контроль над элементами облачной инфраструктуры, включая аппаратную и программную часть, используемую для построения облака.

Определение и создание вышеперечисленных моделей, в первую очередь, связано с необходимостью разделения уровней доступа к облачной инфраструктуре, а также выделения компонентов, необходимых для быстрого развертывания приложений и сервисов, предназначенных для решения вычислительных задач. Характерной особенностью данных моделей является высокая отказоустойчивость и масштабируемость, что позволяет расширять спектр

решаемых задач за счет использования большого разнообразия аппаратных платформ. Сравнение моделей обслуживания, характерных для облачных систем приведено на рисунке 4.1.

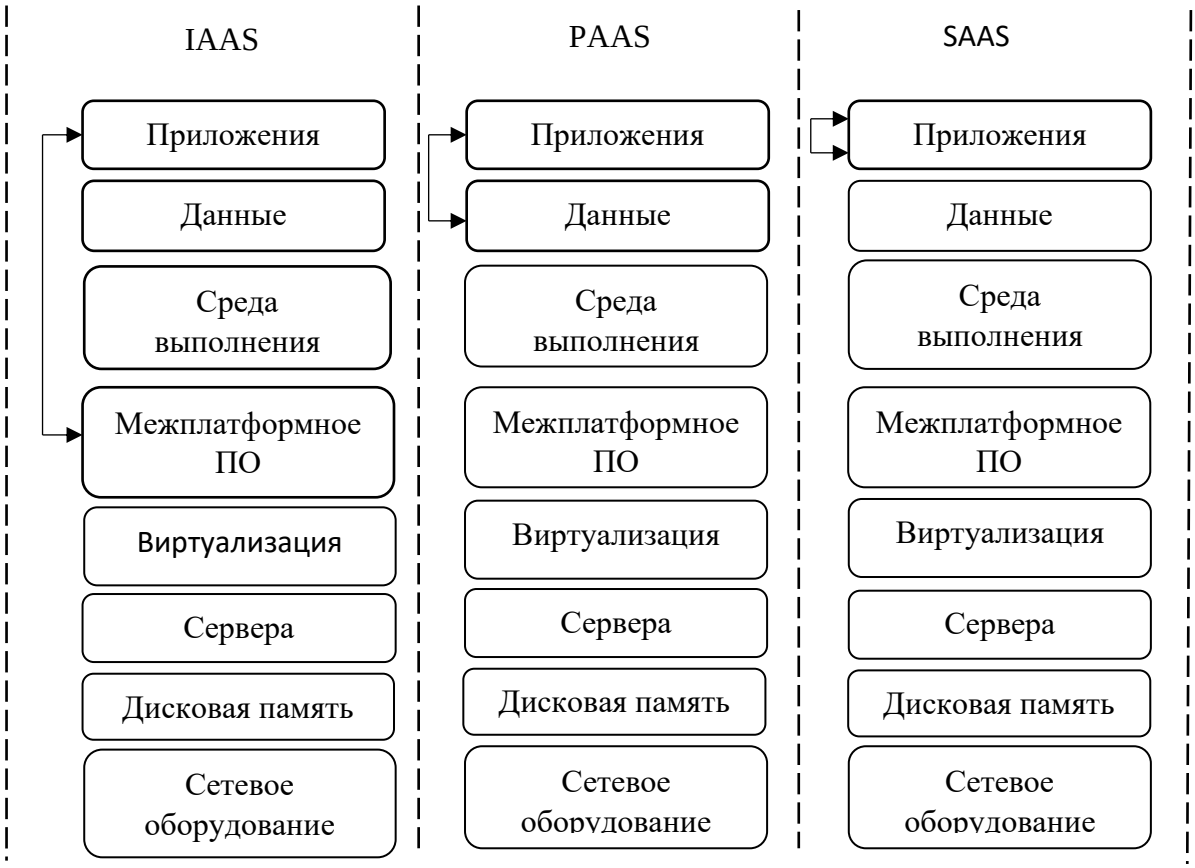

Рисунок 4.1. Сравнение моделей обслуживания облачных платформ

Наиболее гибкими с точки зрения оптимального развертывания являются платформы PaaS и IaaS. Они позволяют использовать готовые образы различных вычислительных систем, что значительно упрощает процедуру развертывания программного обеспечения. Среди облачных PaaS и IaaS платформ наиболее распространенными являются платформы Amazon EC2, Google Cloud, IBM Blue Cloud, Microsoft Azure, VK Cloud и Yandex Cloud. Преимущество данных платформ заключаются в том, что они позволяют исключить необходимость физического размещения и обслуживания серверной аппаратной части, а также предлагают обширный спектр программных утилит и образов быстрого развертывания Apache Hadoop и Spark. В связи с увеличение таких PaaS и IaaS платформ, особую роль играет создание параллельных алгоритмов для решения различного рода математических и технических задач с использованием облачной инфраструктуры.

В частности, одним из таких направлений является необходимость решения задач машинного обучения, классификации и распознавания образов. Такой подход позволяет использовать физические ресурсы облака для реализации вычислений, а также осуществлять оптимизацию проведения данных вычислений.

С точки зрения доступа и размещения выделяют три основных разновидности облачных сервисов: публичное, частное и гибридное облако. Частное облако подразумевает размещения аппаратной инфраструктуры в изолированной среде организации, где доступ к ресурсам предоставляется исключительно в рамках корпоративной внутренней сети. Публичное облако используется для организации распределенного доступа различных пользователи и компаний по всему миру. В таком случае пользователи не имеют возможности управлять облачной инфраструктурой, а их доступ ограничивается набором программно-аппаратных компонентов, необходимых для развертывания индивидуальных вычислительных систем и сервисов. Гибридное облако является сочетанием двух ранее описанных платформ и позволяет объединять ресурсы как частных, так и публичных облаков для решения сложных вычислительных задач. Гибридные облака наиболее применимы в тех случаях, когда необходимо временно расширить возможность частного облака за счет использования вычислительных мощностей публичного.

Использование концепции кластерных и облачных вычислений применительно к процедуре обучения и решения задач вероятностного вывода ДБС является достаточно значимым направление на пути повышения алгоритмической эффективности, а также адаптацию ДБС к предметным областям, требующим построения вероятностных моделей со сложной неоднородной структурой и большим числом параметров. Такое решение обусловлено тем, что множество обучающих выборок для ДБС в основном представляется в виде файлов или набора таких файлов. Как следствие, блоки обучающих выборок могут быть обработаны в параллельном режиме. В процессе выполнения сложных задач машинного обучения ДБС рационально использовать фрагментацию данных на отдельные блоки (операция свертки). В процессе же получения результата выполняется

объединение полученных фрагментов в единую структуру (операция отображения). Данная концепция получила название MapReduce. Использование такого подхода лежит в основе многих облачных и кластерных систем.

В основе создания параллельных алгоритмов и программ особую роль играет разделение уровней параллелизма. Наиболее крупными уровнями такого разделения являются мелкозернистый и крупнозернистый параллелизм [203]. Мелкозернистый параллелизм реализуется за счет создания небольших блоков программы (базовых блоков), в которых полностью отсутствую операции перехода [105]. Использование таких блоков особенно характерно для реализации процедур межпрограммного и межпроцессного взаимодействия. Для оптимизации параллелизма за счет базовых блоков используются различные методы и алгоритмы. Инструменты и библиотеки программных компонентов, разрабатываемые на основе MapReduce, в основном используют стратегию мелкозернистого параллелизма с разделением исходных данных на равные блоки размером от 16 до 64 Мбайт. На сегодняшний день наиболее высокопроизводительные системы, адаптированные для параллельной обработки данных, используют программные платформы Hadoop и Spark. Данные платформы строятся на основе концепции MapReduce [24]. MapReduce структурно делится на две основные операции: свертка (map) и отображение (reduce). На этапе свертки происходит распределения задач между узлами кластера, а также выполнения всех необходимых операций обработки данных. На этапе отображения происходит агрегация всех обработанных данных и их представление в виде единого результата. Платформы Hadoop и Spark объединяют весь спектр возможностей, необходимых для построения параллельных высоконагруженных облачных платформ, а также используются для построения мультикомпьютерных (кластерных) систем по топологии «звезда». Данная топология подразумевает наличие одного мастер-узла и множества вычислительных узлов. Другим аспектом таких систем является распределенная файловая система (РФС), позволяющая объединять дисковое пространство узлов кластера. Это дает возможность оперировать единым дисковым пространством в рамках проведения вычислений, а

также реализовать организацию и систематизацию хранения большего объема файлов. Общая схема РФС представлена на рисунке 4.2

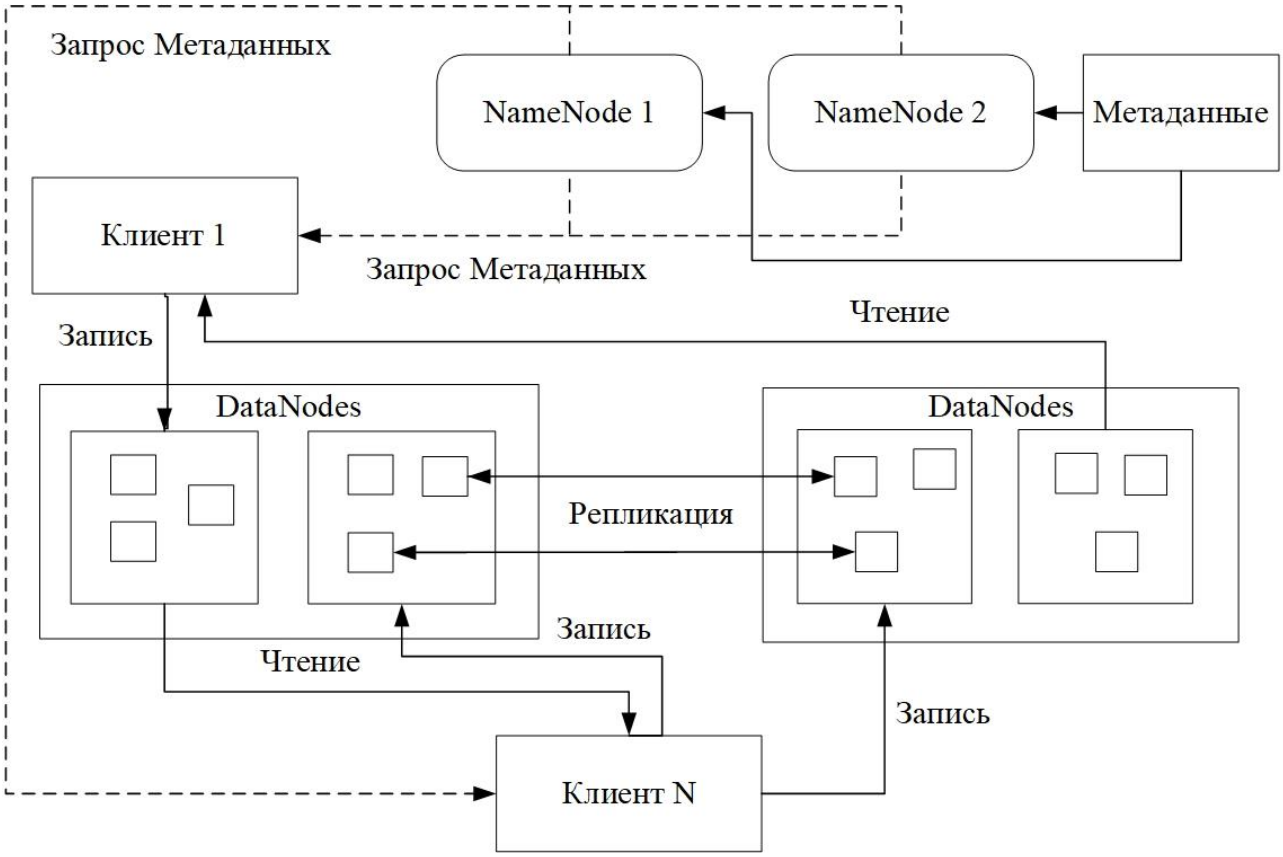

Рисунок 4.2. Структура распределенной файловой системы с репликацией данных

Из рисунка 4.2 видно, что узлы NameNode используются для хранения метаданных относительно реплицированных файлов, расположенных на РФС, а также их имен и расширений. Как правило, в РФС присутствует первичный NameNode и вторичный (резервный) NameNode. Узлы DataNode используются для непосредственного хранения реплицированных данных, при этом NameNode и DataNode могут располагаться на одном физическом сервере – это не нарушает устойчивость РФС. Наибольшее распространение среди таких систем получили HDFS (работает совместно с Hadoop), GlusterFS, Ceph, OpenIO. Использование данных систем обусловлено необходимостью хранения большого набора данных, используемых в процессе обучения различных интеллектуальных систем. Большинство MapReduce систем оптимизировано для работы с РФС и позволяют формировать параллельную обработку файлов непосредственно с РФС. Если

рассматривать РФС применительно к Hadoop и Spark, наиболее оптимальным решением будет HDFS, так как позволяет достичь хороших показателей операций чтения и записи по сравнению с другими РФС. Hadoop и Spark построены по архитектуре РОП, что исключает возникновения различного рода переполнений и коллизий. Преимуществом Spark относительно базового набора компонентов параллельной обработки данных Hadoop MapReduce является кеширование обрабатываемых данных в памяти, что исключает необходимость их постоянной загрузки с файловой системы. Однако в том случае, если имеются ограничения на использование памяти, Hadoop будет наиболее оптимальным решением для выполнения параллельных операций.

Основной абстракцией для Spark является понятие RDD (распределенное множество данных). В общем смысле RDD [93] представляет собой некоторую совокупность (коллекцию) данных, обрабатываемых поверх модели MapReduce и реализующую разновидность формализованного представления РОП систем, оперирующих единой памятью, расположенной на каждом вычислительном узле кластера. Другим компонентом архитектуры Spark является понятие распределенной переменной. Использование распределенных переменных обусловлено тем, что Spark запускает все задания в параллельном режиме. Следовательно, необходимо использовать общее адресное пространство для хранения таких переменных, чтобы не использовать их копии отдельно для каждого параллельного процесса. В свою очередь, Spark предполагает хранение копий переменной для каждого параллельного потока без ее распространения. В RDD отсутствуют реплики в явном виде, что позволяет оптимизировать хранение графа потока выполнения параллельной программы. В таком случае каждый набор команд будет ассоциирован с множеством блоков данных RDD, что исключает необходимость реплицирования данных по сети.

Применительно к решению задач машинного обучения, использование Spark в процессе построения параллельных алгоритмов необходимо для обработки большого объема обучающей выборки и решения задач вероятностного вывода в ДБС с большим числом временных срезов. В таком случае процедура накопления

данных может иметь как дискретный, так и непрерывный характер, а общее число срезов выбирается из условия получения наиболее точных и правдоподобных результатов. В системы Spark интегрирован интерфейс подключения и работы с распределенной файловой системой Hadoop HDFS, что обеспечивает обработку больших файлов, которые располагаться непосредственно на HDFS. Дополнительным преимуществом Spark относительно классического Hadoop MapReduce является более упрощенная процедура создания очередей операцией Map и Reduce, предназначенных для обработки данных, а также оптимизация загрузки данных различных файловых форматов. Рассмотрим общую структуру этапов запуска и выполнения задания с использованием концепции параллельной системы на основе MapReduce. На начальном этапе MapReduce программа выполняет разбиение входных данных на равные блоки, после чего производится многократный запуск приложения на кластере. На следующем этапе мастер-узел, осуществляет распределение заданий Map и Reduce между рабочими узлами и их последующее выполнение. В процессе выполнения Map-заданий рабочий-узел производит обработку выделенному ему блоку данных и производит кеширование результатов выполнения в оперативную память или на диск. Далее мастер-узел производит передачу управляющих команд каждому рабочему узлу о необходимости чтения и применения Reduce-заданий к кэшированным данным. Перед началом выполнения операций Reduce производится сортировка полученных промежуточных данных, которые необходимо обработать в рамках одного и того же Reduce-задания. Это позволяет осуществить конвейеризацию заданий. Обработанные данные Reduce-заданий для каждого рабочего узла аккумулируются в единый блок, который формирует общий результат обработки. Если в процессе выполнения MapReduce-заданий происходит сбой, связанный с недоступностью узла по каким-либо причинам, мастер-узел, инициализирует повторный запуск необработанных данных на доступных узлах. Для хранения очереди заданий используется направленный ациклический граф, что позволяет обеспечить отказоустойчивость выполнения заданий и реализовать процедуру повторного запуска тех узлов-заданий, которые были завершены с ошибкой. В

таком случае сохранение структуры параллельной операция для отдельного задания возможно только с использованием файла реплик узлов, хранящего все реплицированные данные для каждого из узлов. Важной ролью в обеспечении выполнения MapReduce и RDD-заданий является балансирование нагрузки между отдельными узлами вычислительной системы, а также реализация механизмов управления ее распределения. С учетом того, что задания могут выполнятся на достаточно разнородной аппаратной платформе, следовательно, необходимо детально подходить к процессу моделирования распределения заданий, загрузке вычислительных единиц (процессоров), а также временных задержек, связанных с передачей блоков данных через сетевой канал взаимодействия. Общая структура параллельных систем на основе MapReduce представлена на рисунке 4.3

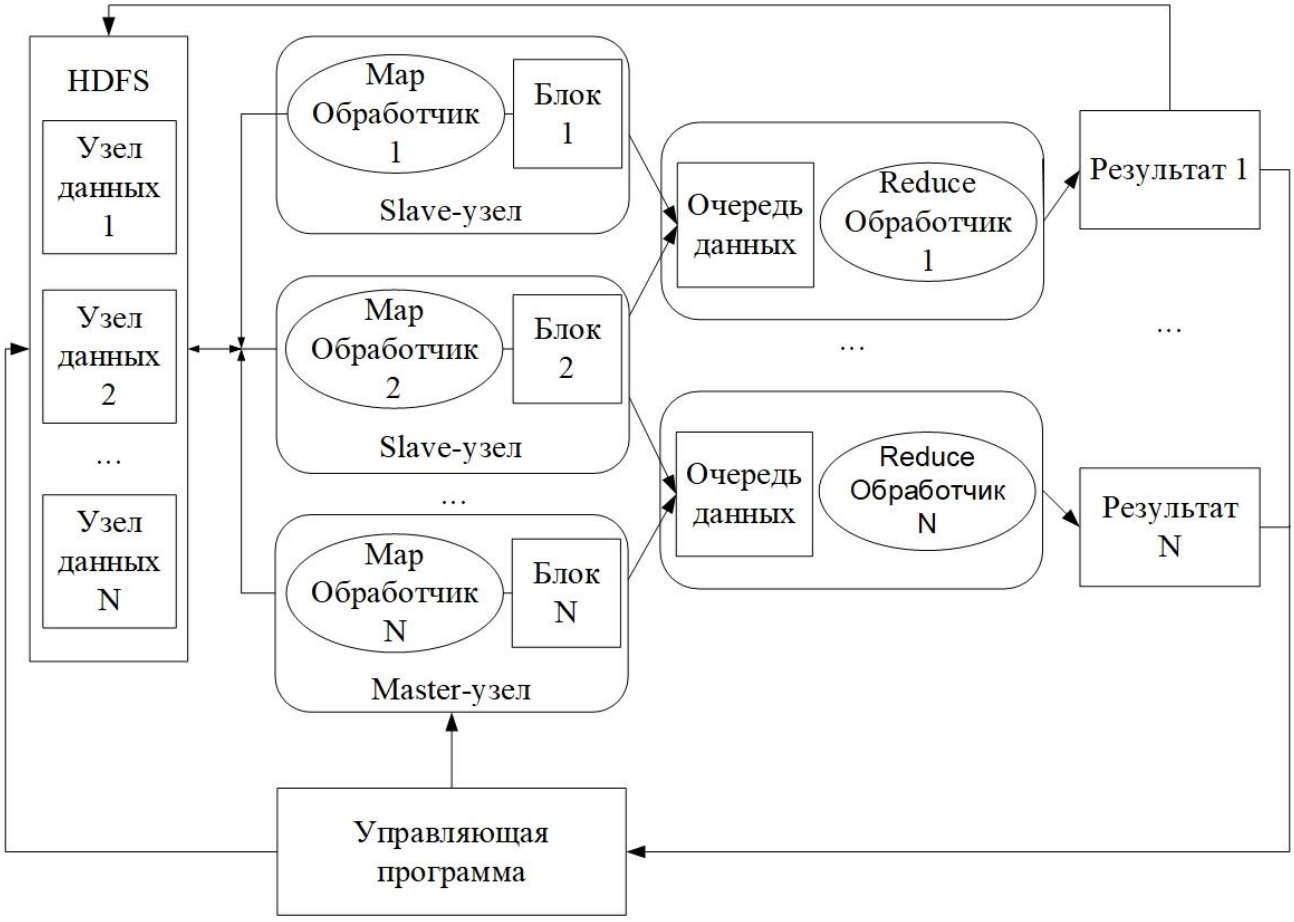

Рисунок 4.3. Структура запросов для параллельной системы с РОП на основе концепции MapReduce

Из данного рисунка видно, что модель RDD может быть преобразована к классическому представлению MapReduce за счет встроенных операторов обработки и свертывания данных с учетом заданной схемы в виде программной

абстракции с соответствующими полями, необходимыми для извлечения результатов выполнения параллельных операций. Такой подход позволяет адаптировать разработанные ранее алгоритмы для MapReduce к выполнению с использованием Spark RDD.

В рамках разработки параллельных алгоритмов обучения и вероятностного вывода в ДБС предполагается использования платформы Spark, работающей поверх развернутого кластера Hadoop и файловой системы HDFS. В работе предполагается оптимизация методологии распараллеливания MapReduce за счет введения абстракций РОП с блокировкой, позволяющей реализовать модель консистентности с синхронизацией по входу или по выходу. Другим направлением исследования является разработка параллельных алгоритмов и их апробация с использованием облачных вычислительных платформ. Следствием этого является повышение скорости выполнения математических операций, совершенствование механизмов распределения вычислительных ресурсов, а также оптимизация хранения моделей, используемых в процессе решения стохастических задач вероятностного вывода. В классической интерпретации структура параллельного процесса может быть представлена в виде диаграммы с отложенными дискретными квантами времени и связанными с ними действиями.

Особый интерес с точки зрения решения задач синхронизации представляет внутреннее устройство RDD, а также порядок выполнения операций с памятью, в процессе передачи переменных между параллельными процессами и вычислительными узлами. Рассмотрим возможные источники данных, используемые в рамках формирования RDD-коллекций:

1. Файлы, находящиеся на распределенной файловой системе HDFS.
2. Сериализованные массивы, формируемые средствами языков программировании Scala/Java, а также коллекции, передаваемые с помощью потокового интерфейса взаимодействия (Spark Streaming).
3. Наборы данных, полученные в результате трансформации уже существующих RDD. В частности, набор элементов, соответствующих типу A может быть преобразован к типу B с использованием оператора *flatMap*,

позволяющего преобразовывать каждый элемент $a \in A$ в элемент $b \in B$ в соответствии с пользовательской функцией $A \to LIST(B)$, задающей правило трансформации элементов.

4. За счет модификации ранее декларированного RDD. Модификация достигается за счет того, что RDD представляют собой модель хранения с отложенной инициализацией. Такой подход позволяет формировать конвейер обработчиков, с инициализацией по мере необходимости в процессе выполнения параллельных операций.

Порядок выполнения заданий Spark представлен на следующем рисунке

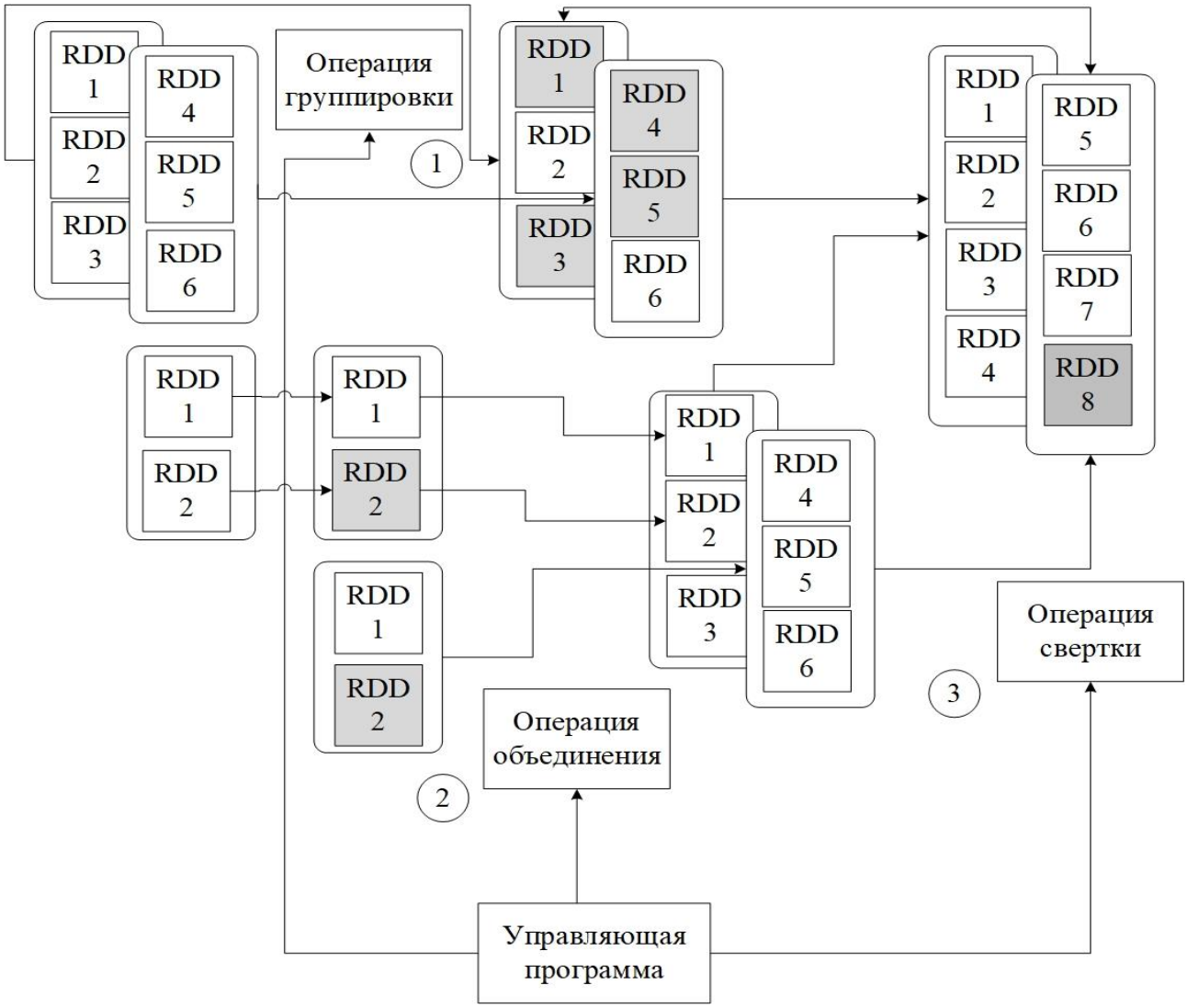

Рисунок 4.4. Порядок обработки заданий на платформе Spark с использованием RDD

На рисунке 4.4 представлен порядок обработки типичных операций с использованием RDD. Закрашенными блоками обозначены RDD в которые уже

загружены данные, используемые в процессе работы параллельных участков кода Spark. В свою очередь, для создания и изменения RDD могут использоваться операции: *reduce* (объединение данных в соответствии с заданной функцией отображения), *collect* (перенаправление всех данных, находящихся в RDD в управляющую программу), *foreach* (обработка каждого элемента RDD в соответствии с пользовательской функцией, определяемой внутри управляющей программы). Рассмотрим порядок обработки параллельных заданий и распределение данных в RDD-памяти. Такой порядок определяется приоритетами операторов, отвечающими за трансформацию данных. Далее рассмотрим RDD с точки зрения контроля персистентности разделяемых ресурсов в процессе выполнения вычислительных операций. Для этого определим возможные типы персистентности и рассмотрим возможность их адаптации к особенностям построения RDD и MapReduce.

Синхронизация распределенных систем (РС) представляет собой достаточно сложный стохастический процесс, требующий постоянной оценки состояния системы и распределяемых параметров, для которых необходимо выполнять синхронизацию. Особенность РС, выполняющих задачи с использованием модели MapReduce, накладывает ряд ограничений, связанных с необходимостью распределения процедуры синхронизации между этапами – Map и Reduce по отдельности. В первую очередь это связано с тем, что невозможно точно предсказать продолжительность выполнения отдельного задания процессом, в связи с чем, переход между состояниями процесса до и после синхронизации будет носить стохастический характер. С другой стороны, в процессе выполнения синхронизации параметров могут возникать различного рода сбои и ошибки, приводящие к остановке выполнения отдельных фрагментов задания, что создает риск возникновения бесконечной блокировки разделяемых параметров такими фрагментами. Среди основных подходов синхронизация выделяют синхронизация на основе событий и передачи маркера. Синхронизация на основе событий (волновые алгоритмы), характеризуется рассылкой широковещательных сообщений, содержащих информацию относительно изменяемого параметра и его

новое значение, которое должно быть задано для всех процессов, выполняющих обработку задания согласно модели MapReduce. При этом схема передачи сообщений, определяется топологией узлов вычислительной системы, а расстановка приоритетов в передачи таких сообщений может быть установлена произвольным образом. Синхронизация на основе передачи маркера подразумевает выполнение синхронизации параметров при получении токена каким-либо процессом, в частности завершения этапа Map. Большинство современных систем, представляют собой системы с общей распределенной памятью, в которой хранение данных производится на основе распределенных таблиц (блоков данных). В связи с этим выделяются несколько основных подходов для обеспечения консистентности данных на основе синхронизации данных систем. Консистентность подразумевает нахождение РОП в неконфликтном состоянии, в котором существует некоторая согласованность относительно использования памяти несколькими процессами. Это дает возможность обеспечить корректность чтения (запись) переменных из памяти в процессе выполнения вычислительных операцией [205]. Рассматривая процедуру синхронизации на основе волновых алгоритмов, стоит отметить, что для реализации подхода на основе распространения сообщений один из процессов должен выступать в качестве инициализатора. Инициализатор используется для определения входной точки процесса синхронизации и определяет условие входа в критическую секцию. После выполнения синхронизации процесс-инициатор передает сообщения другим процессам. Если передать сообщение невозможно в силу возникновения ошибки, то событие ошибки передается узлу координатору. Для параллельного процесса p событие a_p, а для других процессов событие b_p должно быть выполнено не раньше, чем событие a_p. Сформулируем условие непротиворечивости обработки событий для нескольких параллельных процессов [207]. Если процессу необходимо выполнить a_p, он инициализирует отправку сообщения другим процессам о необходимости выполнения события a_p, гарантируется единоличный доступ к a_p, остальные процессы будут ожидать завершения обработки события a_p.

Рассмотрим наиболее известные модели синхронизации на основе консистентности.

Среди РОП без синхронизации выделяют следующие модели: строгой, последовательной, причинной, PRAM (Pipelined RAM) и слабой консистентности. Модель *строгой консистентности* связана с понятием времени доступа к области памяти, в которой хранится переменная. Разделение доступа процессов разграничивается временем доступа к ресурсу памяти до тех пор, пока процесс использует переменную, другие процессы не имеют к ней доступа. Для модели *последовательной консистентности* характерно четкое понимание последовательности операцией записи в память, при этом сама последовательность записи строится исходя из графа выполнения программы. В *причинной модели* впервые описано то, что каждое изменение значений зависимых переменных должно соблюдаться всеми процессами. Если два процесса изменяют различные переменные в один и тот же момент времени, то это будут свидетельствовать об отсутствии между ними причинно-следственной связи. Из этого следует, что операции записи переменных несколькими параллельными процессами должны быть согласованными. Модель PRAM представляет собой набор условий консистентности, заключающихся в том, что все операции записи характерные для текущего процесса, будут видны всем других процессам в той же последовательности, в которой они обрабатывались для текущего процесса, в то время как операции, выполняемые произвольными процессами будут доступны в любом порядке. В этом случае операции записи могут быть объединены в конвейер, однако следует учесть, что порядок постановки и выполнения задач в конвейере должен быть доступен для любых других процессов с целью исключения коллизий. Для модели *слабой консистентности* [117,118] будут характерны следующие правила: работа с переменными, подлежащими синхронизации будет выполняется в соответствии с моделью последовательной консистентности, доступ к переменной, для которой выполняется синхронизация, блокируется для всех процессов до тех пор, пока не будут завершены все операция записи. Обработка

данных на чтение и запись будет приостановлена, до того времени, пока не будут обработаны все обращения к синхронизированным переменным.

Модели РОП с синхронизацией можно разделить исходя из вида консистентности: слабая, по выходу, по входу. В моделях консистентности на основе синхронизации вводится понятие критических секций, определяющее и устанавливающее порядок блокирования переменной синхронизации (операция Lock) и ее освобождение (операция Release). Ключевое отличие между моделью синхронизацией по входу и выходу заключается в том, что синхронизация по входу предназначена для установления связи между изменяемой переменной и ее переменной синхронизации (семафором). Такой подход позволяет реализовать селективную блокировку между переменными и исключить возникновение коллизий в процессе выполнения операций чтения или записи. *Слабая модель* устанавливает консистентность системы только после выполнения синхронизации разделяемой переменной. Это достигается путем реализация монопольного доступа к разделяемым ресурсам системы. С точки зрения использования представленных моделей консистентности существует ряд алгоритмов, позволяющих реализовать как базовые, так и основные аспекты синхронизации распределенных систем. Базовым алгоритмом синхронизации является централизованный алгоритм синхронизации. Данный алгоритм является интуитивно понятным и предполагает, что один из процессов может выступать в качестве координатора, отвечающего за распределение разделяемых ресурсов между различными процессами распределенной системы. Такому процессу задается наивысший приоритет. Запросы на доступ к разделяемым ресурсам формируются в виде очереди и выполняются в порядке поступления.

Далее рассмотрим модель распределения ресурсов между параллельными процессами, представленную в Spark. Одним из способов обеспечения синхронизации является механизм полного копирования содержимого разделяемого ресурса между процессами, в таком случае каждый процесс будет иметь свою локальную копию переменной в процессе выполнения операций с RDD. Все декларируемые переменные в рамках пользовательских функции имеют

ограниченную видимость и располагаются непосредственно в RDD. Такой подход имеет ряд ограничений, связанных с необходимостью постоянного копирования значений переменных в случае выполнения параллельных блоков, что ведет к перегрузке системы. Альтернативным механизмом синхронизации данных между параллельными процессами является механизм широковещательной рассылки переменных, выполняемый один раз перед реализацией какой-либо операции с RDD. В таком случае значение переменной может передаваться в виде указателя на соответствующий блок RDD, отвечающий за хранение данной переменной. Для изменения значений переменных с числовыми значениями предусмотрен так называемый шаблон «Аккумулятор», позволяющий выполнять атомарный доступ и инкрементирование значений переменных внутри параллельного блока. В основе реализации данных подходов синхронизации ресурсов в Spark предложена разновидность протокола BitTorrent. Механизм BitTorrent для осуществления доступа к разделяемым ресурсам системы, размещенным в распределенной памяти представлен на рисунке 4.5

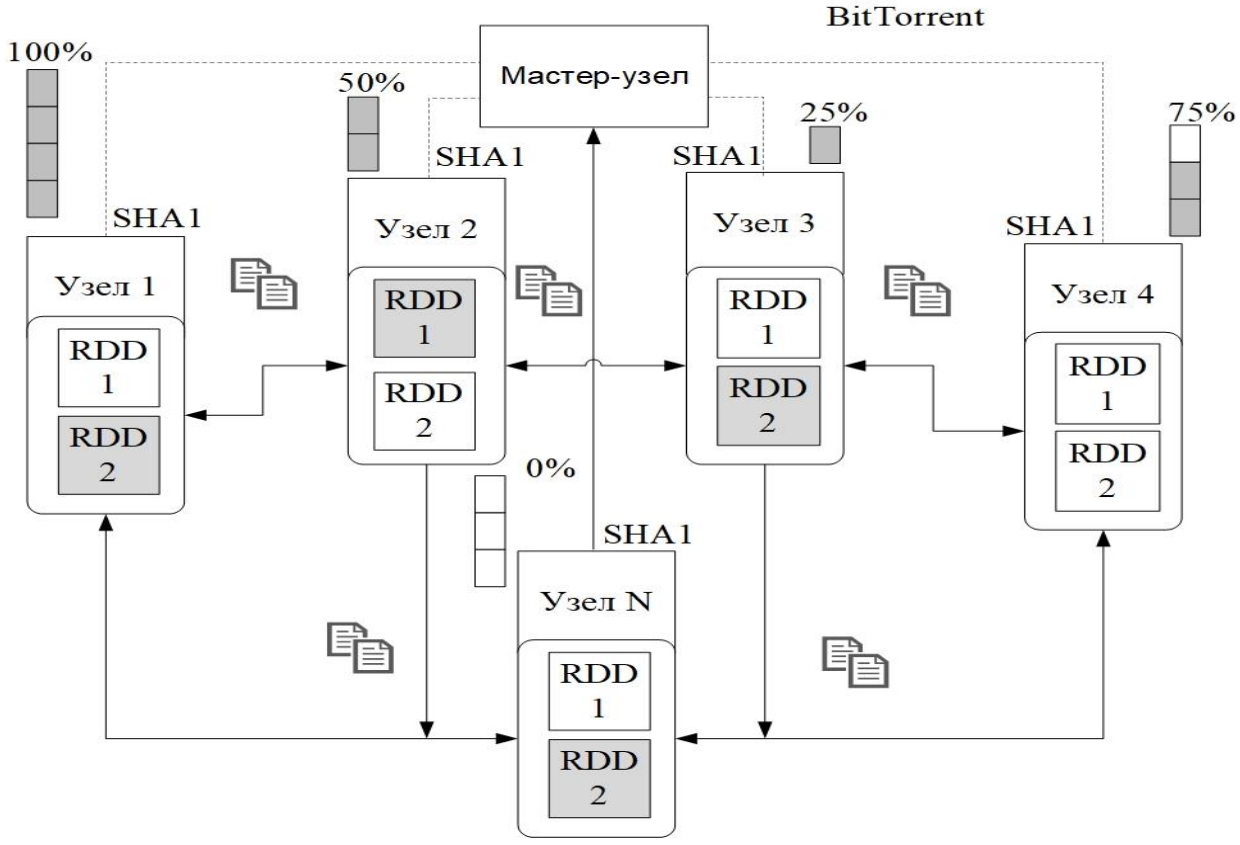

Рисунок 4.5. Распределение разделяемых ресурсов внутри RDD-узлов Spark

Из рисунка 4.5 видно, что мастер-узел выступает в виде «трекера», аккумулирующего информацию, относительно узлов, которые доступны для распределения информации. После чего узел может как запрашивать недостающие фрагменты данных, так и осуществлять широковещательную рассылку между соседними узлами информации, которая уже ему доступна в данный момент. В отличии от классического алгоритма BitTorrent, Spark используется валидацию не отдельных фрагментов данных, передаваемых между узлами, а общего блока данных, который получает каждый из узлов на окончательном этапе выполнения алгоритма. Такой подход обладает преимуществом, заключающимся в том, что нет необходимости вычислять хэш-суммы для каждого блока. Наряду с этим есть существенный недостаток, связанный с тем, что при сбое выполнения, часть данных, которая пересылалась ранее, может быть потеряна. Соответственно требуется повторная их пересылка для выполнения операции контроля целостности, передаваемой между узлами информации. Для этого в Spark уменьшен интервал запросов отдельными узлами на мастер-узел, однако это ведет к повышению нагрузки на канал взаимодействия между элементами вычислительной системы. В качестве процедуры оптимизации в работах Захарии и Чевдхуру [19] предложен алгоритм оптимизации распределения блоков данных между узлами вычислительной сети с использование следующего алгоритма. На первом этапе происходит формирование матрицы расстояний D, элементами которой являются средние значения времени передачи данных между соседними узлами. После чего элементы матрицы D проецируются на двумерное пространство за счет использования метода многомерного шкалирования. На завершающем этапе выполняется кластеризация узлов с использованием смеси гауссовых распределений с дисперсией $\mathbb{D}$ и осуществляется автоматический выбор числа классов на основе БИК-критерия. Применение такого подхода в сочетании с алгоритмом BitTorrent позволяет оптимизировать процедуру распределения разделяемых данных между отдельными RDD. На практике при использовании алгоритмов распределения ресурсов возникает ряд сложностей, связанных с возникновением избыточности в случае сбоев выполнения параллельных модулей

вследствие того, что целостность передаваемых данных выполняется лишь при полной загрузке данных в RDD отдельных узлов. Второй существенной проблемой является использование матрицы расстояний, так как необходимо предварительно определять среднее время выполнение передачи данных. Для решения этого могут быть использованы различные метрики, в частности XOR-метрики, позволяющие выбрать наиболее оптимальные для организации взаимодействия узлы. Алгоритм BitTorrent наиболее применим к распределенным разнородным сетям, без возможности организации централизованного управления.

В Hadoop MapReduce в явном виде отсутствует механизм синхронизации переменных, неявно же Hadoop осуществляет копирование всех переменных между узлами системы, но только в пределах операций свертки и отображения. С ростом числа реплик наблюдается существенное повышение времени, необходимого на распределение ресурсов между параллельными процессами. В свою очередь, для Spark и Hadoop могут быть использованы альтернативные подходы к решению задач распределения ресурсов и синхронизации данных. Наибольший интерес представляют модели на основе сетей Петри и систем массового обслуживания, предоставляющие различные механизмы моделирования процессов синхронизации и решения задач оптимального распределения ресурсов вычислительной системы.

4.2. Использование инструментов теории массового обслуживания для синхронизации вычислений в распределенных вычислительных системах

Первые работы на основе логики событий, реализующие процесс синхронизации распределенных систем, основываются на понятии синхронизации времени и приводятся в работах Лэмпорта [46]. Им была предложена модель на основе логических часов, используемая для определения порядка следования событий в распределенной системе, формируемых параллельными процессами при выполнении задач. На основе логических часов происходит формирование меток времени для каждого процесса, который хочет получить доступ к разделяемому

ресурсу. Согласно утверждению Лэмпорта для работы алгоритма необходимо, чтобы сеть была полносвязной, а каждый процесс получивший доступ к критической секции, мог накапливать очередь поступающих заявок на обращение к разделяемому ресурсу другими процессами. В отличии от классического централизованного алгоритма, где последовательность заявок может носить стохастический характер, в алгоритме Лэмпорта очередь будет формироваться в порядке обращения процессов к разделяемому ресурсу. Разработанная модель синхронизации Лэмпорта представлена на рисунке 4.6

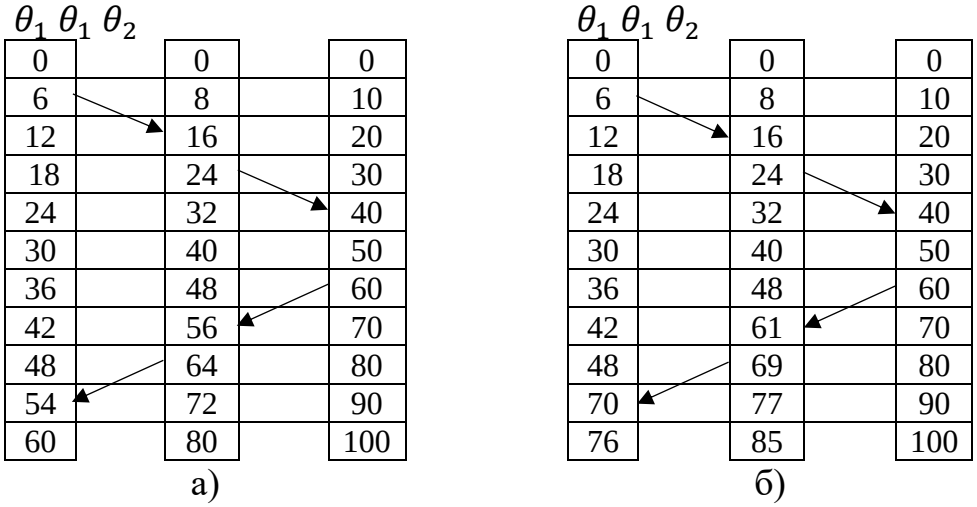

Рисунок 4.6. Синхронизация с помощью логических часов для трех параллельных процессов

Для модели Лэмпорта в качестве основы для синхронизации событий, протекающих внутри распределенной системы, используется функция времени $\tau(E_i)$. Можно определить текущее время процесса T_p в виде совокупности локального и глобального времени, характеризующего текущий процесс. В своих работах Лэмпорт сформулировал несколько условий, характеризующих непротиворечивость логических часов. Первое заключается в том, что каждый процесс в ходе выполнения задания характеризуется приращением времени $\Delta t = T + \delta$. При этом параметр δ в большинстве случаев полагаем равным константе. Второе условие заключается в добавлении в каждое передаваемое сообщения M_s между процессами θ_i и θ_j отметок времени T_i, ассоциируемых с процессом θ_i. Обозначим данную сумму как ΔT_E. В таком случае отметка времени для процесса θ_j определяется как максимальное из значений $\max(T_j, \Delta T_E)$. Сформулируем

условие непротиворечивости логических часов Лэмпорта для событий E_i и E_j, характеризующих переход между процессами θ_i и θ_j

$$E_i \to E_j \Rightarrow T_i(E_i) < T_j\left(E_j\right). \quad (4.9)$$

Пусть $T_{\text{i}}(E_i) = \Delta t_i$ и $T_j\left(E_j\right) = \Delta t_j$. Общая интенсивность $I(t)$ обмена сообщениями M_s будет зависеть от параметра δ для обоих процессов θ_i и θ_j. Для соблюдения условий Лэмпорта подразумевается, что каждый из процессов завершает выполнение очереди заданий без ошибок, иначе временные метки $T(E)$ не смогут быть определены. С практической точки зрения Лэмпортом было доказано [45], что δ должно быть максимально допустимым для выполнения передачи всех генерируемых сообщений за δ секунд, в противном случае, это приведет к переполнению канала связи, обеспечивающего взаимодействие между рассматриваемыми процессами. Стоит отметить, что если процессы θ_i и θ_j имеют связь через промежуточный процесс θ_k, то $T_{\text{j}}\left(E_{\text{j}}\right)$ будет включать в себя приращение времени Δt_k, соответствующее процессу θ_k. Если промежуточный процесс θ_k завершился с исключением, то синхронизация с помощью логических часов для θ_j не сможет быть выполнена. На основании вышеизложенного, определим структуру алгоритма Лэмпорта для выполнения синхронизации распределенной системы. На первом этапе, если процессу θ_i требуется осуществить доступ к разделяемому ресурсу R, он производит рассылку сообщения $M_R(T_i, i)$, содержащего локальную метку времени для текущего процесса, всем другим процессам $\theta = (\theta_1, \theta_2, \dots, \theta_n)$. В свою очередь, процесс θ_j получает сообщение M_L и отправляет процессу θ_i подтверждающее сообщение $M_A\left(T_j, j\right)$, содержащее локальное время T_j и гарантирующее, что процесс θ_j не будет использовать R до тех пор не получит соответствующее сообщение от θ_i. Процесс θ_i сохраняет все метки времени из сообщений $M_A(T_k, k)$, полученных от каждого из процессов, включая свою метку времени в очереди Q_i. На следующем этапе процесс θ_i осуществляет доступ к разделяемому ресурсу, соблюдая при этом условие непротиворечивости, гарантирующее, что наименьшей меткой времени в Q_i будет T_i. На завершающем этапе процесс θ_i производит рассылку сообщений

$M_U(T_i, i)$, характеризующих его выход из критической секции и освобождение доступа к ресурсу R. После чего алгоритм повторяется для процесса θ_k и взаимодействующих с ним процессов θ. Суммарный объем сообщений, необходимых для обработки одного события синхронизации будет $S_M = 3 \times (n - 1)$, n – число процессов, требующих получение доступа к разделяемому ресурсу.

Алгоритм Лэмпорта обладает достаточно большой избыточностью, так как для обеспечения требуемого уровня синхронизации необходимо сгенерировать большой объем сообщений $M_R(T_i, i)$ и $M_U(T_i, i)$. Анализируя, получаем, что для сообщений $M_U(T_i, i)$ сигнализирующих удаление блокировки ресурса, отметки времени T_i можно отбросить, так как они не участвуют в процессе формирования блокировок для последующего процесса θ_k. В таком случае целесообразно объединить сообщения $M_A(T_k, k)$ и $M_U(T_i, i)$, а отметка времени T_i будет совместно определять состояние логических часов в момент подтверждения блокировки ресурса и его освобождение процессом θ_i. Применение такого подхода было предложено в работах Рикарта-Агравала. Объединение двух вышеописанных сообщений синхронизации достигается за счет использования отложенных ответов. Любой процесс θ_k для которого необходимо выполнить блокировку ресурса, в случае входа в критическую секцию θ_i, будет осуществлять сравнение своей отметки времени T_k и отметки времени T_j, характеризующей процесс θ_i. Если $T_k > T_j$, то отправка сообщения $M_U(T_i, i)$ процессу θ_j будет отложена, в противном случае сообщение будет отправлено незамедлительно. В этом случае получаем, что процесс с наименьшей меткой времени T_k будет входит в критическую секцию значительно раньше остальных процессов, которые будут ожидать получения сообщения $M_U(T_k, k)$.

Альтернативой двум описанным алгоритмам является подход, предложенный Сузуки и Касами [82]. В общем случае это широковещательный алгоритм, где в качестве параметра синхронизации используется маркер, в отличии от временных меток в рассмотренных алгоритмах. Такой подход имеет ряд преимуществ, главный из которых заключается в том, что исключается необходимость определения локальных меток времени каждого их процессов.

Условие синхронизации заключается в том, что если процесс владеет маркером, то порядок доступа к нему при освобождении текущим процессом, определяется согласно очереди, формируемой в результате отправки запросов для обработки ресурса. Процедура передачи маркера, характеризующая вход в критическую секцию процессом θ_i, выполняется за счет трансляции привилегированного сообщения Pr_i. Первоначально только один процесс θ_i имеет привилегию на владение маркером. Если какой-либо из процессов θ_k хочет получить маркер, то он производит широковещательную рассылку сообщений R_j всем остальным процессам θ_j. Отметим, что процесс θ_j, получивший маркер, может осуществлять к нему доступ периодически, до тех пор, пока не будет отправлено сообщение Pr_i другому процессу θ_i. Структура сообщения на запрос маркера определяется как $R_j = R(j,p), p = 1,2, \dots m$. Здесь j определяет порядковый номер процесса, а p – кортеж чисел, определяющих число обращений процесса θ_j для входа в $(p+1)$ критическую секцию. В таком случае каждый процесс θ_j содержит список обращений RN, используемый для хранения максимального значения из кортежа p для каждого из остальных процессов. Процедура обновления очереди RN при поступлении сообщения $R(i,n)$ для процесса θ_i имеет следующий вид:

$$RN[j] = max(RN[j], n). \qquad (4.10)$$

Далее рассмотрим механизм отправки сообщений Pr_i. Формат сообщений имеет вид $Pr_i(Q, LN)$, Q – очередь узлов размерностью N_Q, для которых выполняется рассылка, LN – список обращений, хранящий кортежи значений, такие, что $LN[j]$ порядковый номер сообщения запроса процесса θ_j. После того, как процесс θ_j заканчивает выполнение операций внутри критической секции, производится процедура обновления списка $LN[i] = RN[i]$ в соответствии с последним полученным сообщением Pr_i для процесса θ_i. Далее выполняем обновления списка $RN[j] = LN[j] + 1$ для процесса θ_i, после чего добавляем процесс θ_i в очередь Q, если $\theta_i \notin Q$. После того, как предыдущая операция будет завершена и $Q \neq \emptyset$, отправляем сообщение $Pr_i\big(Q\big[N_Q - 1\big], LN\big)$ первому процессу в очереди Q. Пока $Q = \emptyset$, процесс θ_i будет удерживать маркер до тех пор, пока не

появится процесс $\theta_j \in Q$ за счет рассылки сообщений R_j процессом θ_j. Преимуществом алгоритма Сузуки-Касами является его устойчивость к появлению циклических блокировок, связанных с тем, что лишь единственный процесс имеет возможность входа в критическую секцию и эта операция выполняется пошагово. На практике при использовании алгоритма Сузуки-Касами рационально ограничить длительность пребывания в критической секции некоторым максимально допустимым значением t_{max}, тогда время блокировки ресурса одним процессом не будет превышать данного порогового значения $t < t_{max}$. Такой подход обеспечивает исключение взаимных блокировок в случае возникновения исключений внутри процесса, владеющего в текущий момент времени разделяемым ресурсом. После достижения порогового значения t_{max}, алгоритм автоматически передаст управления другим процессам в соответствии со схемой работы алгоритма.

Функциональная схема алгоритма Сузуки-Касами [66] имеет вид

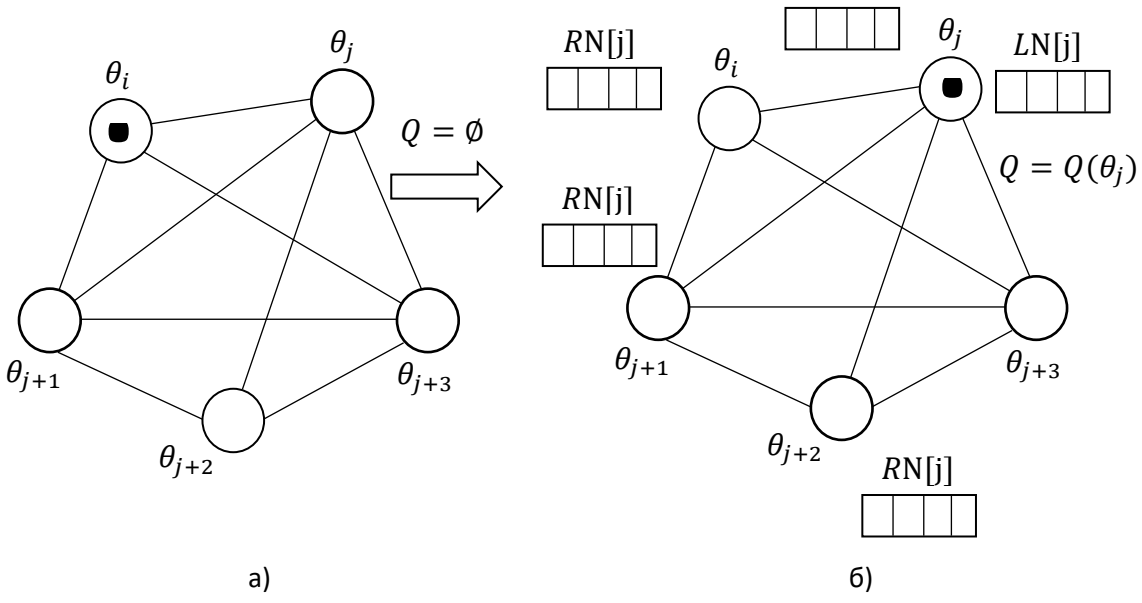

Рисунок 4.7. Этапы синхронизации параллельных процессов на примере процессов θ_i и θ_j с использованием алгоритма Сузуки-Касами

Отметим, что на рисунке 4.7 жирной линией обозначены узлы (процессы), имеющие непосредственную связь с узлами θ_i и θ_j, владеющими маркерами для шагов а) и б) соответственно.

Другим подходом для обеспечения синхронизации параллельных процессов

является моделирование синхронизации в виде графа. В таком случае любой параллельный набор действий можно представить в виде совокупности событий, представленных в виде дискретного процесса, а процедура синхронизации может выполняться на основе событий. Ребра графа задают правила обеспечения синхронизации. Понятие процесса, введенное в работах Петри, тесно связано с такими понятиями как действие и изменение условий. Первое предполагает, что процесс порождается за счет возникновения некоторого события в системе. Изменение условий предполагает, что выполнение процесса связано с изменением состояния системы. В общем случае процесс структурно может быть представлен в виде ориентированного графа $G = (V, E)$. Множество вершин графа V может быть представлено как совокупность множества процессов $P = \{p_1, p_2, \dots, p_n\}$ и ресурсов $R = \{r_1, r_2, \dots, r_m\}$, которые обрабатываются данными процессами. Ребра $e = (p_j, r_i,)$ графа G будут определять доступ подпроцесса p_i к ресурсу r_i. В процессе выполнения алгоритма предполагается, что число ребер $n_{e_k} \geq 0$ и что каждый подпроцесс имеет доступ хотя бы к одному ресурсу в рамках выполнения математических расчетов.

В результате доступа нескольких подпроцессов к разделяемому ресурсу, а именно наличию одновременно нескольких ребер $e = (p_j, r_i,)$, приоритет доступа по умолчанию выставляется в соответствии с очередью операций на получение доступа к ресурсу r_i. Общее число обращений к ресурсу r_i определяется как сумма всех возможных обращений к r_i

$$\sum_j |(p_j, r_i,)| \leq n^i_{e_k}. \tag{4.11}$$

Для оптимизации выполнения параллельных блоков, требующих доступ к разделяемым ресурсам r_i, целесообразно представить процесс выполнения операции в виде стохастической модели. Данная модель будет описывать вероятностные задержки параллельной системы, связанные с осуществлением операций чтения и записи в область памяти распределенных таблиц данных, содержащих разделяемые между процессами переменные. Среди стохастических моделей процессов синхронизации рассмотрим модели сетей Петри и массового

обслуживания. Сети Петри и их возможные структурные разновидности представляют собой хорошо апробированный инструмент построения сложных динамических процессов, позволяющий выстраивать логико-временные временные связи между моделируемыми процессами. Обширное применение сети Петри находят для моделирования дискретных динамических процессов, в частности, параллельных и асинхронных процессов. Основное преимущество данных сетей в процессе моделирования параллельных систем заключается в возможности декомпозиции сложных процессов на отдельные параллельные логические блоки.

Математическую модель сети Петри можно представить в виде совокупности следующих параметров. $N = (P, Tr, I, O)$, $P = \{p_1, p_2, \dots, p_n\}, n > 0$ – множество позиций сети, $Tr = \{t_{r_1}, t_{r_2}, \dots, t_{r_m}\}$ – возможные переходы внутри сети, $I: P \times Tr \to N_0$ и $O: Tr \times P \to N_0$ – входные и выходные функции соответственно. Важной особенностью данных сетей является то, что множества P и Tr не имеют общих элементов и являются непересекающимися множествами $P \cap Tr = \emptyset$. В свою очередь, для наглядного представления сети Петри используется ее графовое представление в виде двудольного ориентированного мультиграфа. Определить такую сеть также удобно в следующей матричной форме [106]:

$$R = R^+ - R^- \text{ матрица инцидентности,}$$
$$R^+ = r_{ij}^+ = \begin{cases} k, p_i \in \Gamma(t_j) \\ 0, p_i \notin \Gamma(t_j) \end{cases}, k \text{ –кратность,} \tag{4.12}$$
$$R^- = r_{ij}^- = \begin{cases} h, p_i \in \Gamma^{-1}(t_j) \\ 0, p_i \notin \Gamma^{-1}(t_j) \end{cases}, h \text{ –кратность.}$$

Из матричного представления сети Петри (4.12) следует, что элементы r_{ij}^+ и r_{ij}^-, входящие в состав матриц R^+ и R^-, равны кратностям для входных и выходных состояний для каждого перехода, входящего в состав сети.

Для определения состояния параллельного процесса возникает необходимость использования маркированной сети Петри. Основная сущность маркировки заключается в определении фишек для заданных позиций сети, при

этом число фишек может изменяться. Маркированная сеть Петри $M_N = (N, M), M: P \to N, N = \{0,1,2, \dots, n\}$ является объединением сети N и маркировки M. Маркировка $M = (m_1^0, m_2^0, \dots, m_n^0)^T$ представляет собой начальный вектор маркированной сети. Выражение, определяющее маркированную сеть Петри, имеет следующий вид:

$$N_m = (P, Tr, I, O, M). \quad (4.13)$$

Процесс перехода из одного состояния в другое для сети Петри происходит за счет удаления входных меток маркированной сети с последующим установлением выходных состояниях. Если из одного состояния допустим переход в два и более состояний, то переход может быть выполнен случайным образом в любое состояние, связанное с текущим. Если ни одно из условий перехода между состояниями не выполняется, то данная сеть будет заблокированной, а все последующие переходы будут недостижимы. После срабатывания перехода маркировка сети M_{m+1} имеет следующий вид [135,136,204]:

$${M_{m+1}}^i = {M_m}^i + O(t_j, p_i) - I(p_i, t_j), p_i \in P, t_j \in Tr. \quad (4.14)$$

Маркированная сеть (два маркированных ресурса), описывающая модель параллельного доступа к ресурсам, представлена на следующем рисунке в виде сети Петри

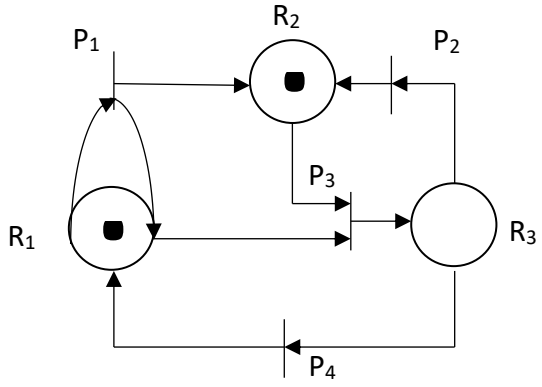

Рисунок 4.8. Сеть Петри для распределения ресурсов между подпроцессами

При описании моделей параллельных процессов с помощью сети Петри, представленной на рисунке 4.8, переходы ассоциируются с процессами, в то время как узлы описывают ресурсы, а именно необходимые условия выделения (освобождения) ресурса. Между тем концепция параллельного процесса

предусматривает одновременный доступ нескольких подпроцессами к одному и тому же ресурсу с возможностью изменения состояния данного ресурса. Такая модель предусматривает введения времени ожидания процесса до освобождения блокировки. Моделирование данного случая возможно с применением временной сети Петри, получаемой в процессе преобразования сети и представленной на рисунке 4.9

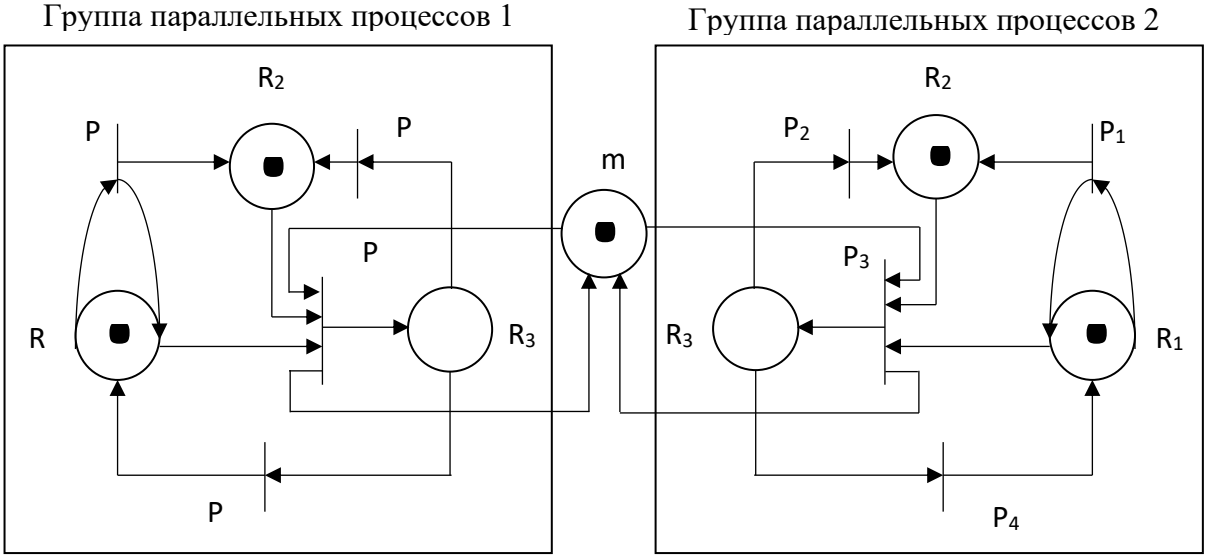

Рисунок 4.9. Временная сеть Петри с блокировкой ресурсов между параллельными процессами

Из рисунка 4.9 видно, что ресурс, имеющий условное обозначение R_3, расположенный в обоих параллельных процессах, представляет собой критическую секцию. Попадание процесса в критическую секцию означает то, что он будет ожидать освобождения m до тех пор, пока ресурс будет занят другим параллельным процессом. Такие процессы будут называться конкурентными. В классическом представлении сети Петри достаточно сложно использовать в качестве модели распределенной вычислительной системы. Это обусловлено тем, что в реальной параллельной вычислительной системе (ПВС), число параллельных процессов, одновременно имеющих доступ к разделяемому ресурсу, будет достаточно велико. В связи с этим возникает необходимость оптимизации процедуры блокировки в ПВС. В частности, допустима ситуация, когда необходимо выполнить блокировку доступа к распределяемому ресурсу только на запись, в то время как на чтение он будет доступен всем параллельным процессам

без исключения. Такой подход может дать существенный прирост в системах, где запись в ресурс происходит достаточно редко. В общем случае сети Петри, представленной на рисунке 4.9, несложно задать начальную маркировку для упрощения моделирования синхронизации процессов. Однако, в сложных параллельных системах начальная маркировка сети не всегда возможна в связи с тем, что при запуске не известно, какой именно ресурс должен быть ассоциирован с текущим процессом. Возникновение таких ситуаций приводит к появлению временных задержек. Для установления стохастических характеристик данных задержек используется аппарат теория вероятностей. В основе стохастических сетей Петри лежит вероятностная оценка переходов внутри сети, а именно с каждым переходом ассоциируются условные вероятности срабатывания. В таком случае распределение маркировок сети может носить дискретный или непрерывный характер. Для моделирования параллельных процессов наиболее целесообразно использование дискретных временных стохастических сетей Петри (ДВССП). В основе ДВССП предполагается использование Марковской цепи с дискретным временем. В основе формирования переходных вероятностей лежит критерий однородности цепи Маркова. Данное условие показывает, что последовательность случайных величин, определяющих множество состояний S, будет являться однородной марковской цепью в том случае, если текущее состояние S_{t+1} зависит лишь от предшествующего состояния S_t

$$P(X_{t+1} = k | X_0 = k_0, \dots, X_t = k_t) = \\ = P(X_{t+1} = k | X_t = k_t), k \in S, k_t \in S \quad (4.15)$$

где $P(X_{t+1} = k | X_t = k_t)$ – переходная вероятность из состояния S_{t+1} в S за единицу времени.

Совокупность переходных вероятностей можно представить в виде матрицы переходов $p_{ik} = P(X_{t+1} = k | X_t = i)$ для которой выполняется следующее условие:

$$\sum_{k \in S} p_{ik} = 1, \forall i = 1,2,..,n. \quad (4.16)$$

Математическая модель ДВССП может быть получена из классической сети Петри за счет добавления функции вероятностей условных переходов $\Phi_N: T_n \to (0;1)$ и имеет вид $N^*(N, \Phi_N)$.

На каждом шаге, соответствующем изменению маркировки сети на временном срезе t, происходит срабатывание множества допустимых переходов, связанных с данной маркировкой, но только в том случае, если они не приводят к конфликтному состоянию. В противном случае срабатывает только один из возможных переходов. Основная сущность ДВССП заключается в определении условных вероятностей для каждого из таких переходов, связанных с текущей маркировкой M' для временного среза t. Условная вероятность срабатывания переходов $Q \subseteq T$ в маркировке M определяется на основе следующего выражения:

$$P(Q, M) = \prod_{\tau \in Q} \Phi_N(\tau) \prod_{\tau \in E(M') \setminus Q} \left(1 - \Phi_N(\tau)\right) \tag{4.17}$$

где E – множество всех допустимых переходов для M'.

Анализируя число элементов, принадлежащих множеству достижимых маркировок ДВССП можно прийти к выводу, что оно пропорционально аналогичному числу элементов множества маркировок в оригинальной сети Петри. На основе этого утверждения следует, что свойства ДВССП можно анализировать через граф достижимости сети Петри. Использование стохастических сетей Петри с блокировками в алгоритмах обучения структуры и параметров динамических байесовских сетей, позволяет повысить эффективность распределения ресурсов между параллельными процессами, а также снизить временные задержки и число взаимных блокировок между процессами в рамках организации доступа к разделяемым ресурсам.

Рассматривая сети Петри с точки зрения модели синхронизации распределенных вычислительных систем (РВС), возникает ряд проблем, связанных с тем, что требуется определить какой именно процесс в текущий момент времени владеет маркером. Необходимо сформировать достаточно большой набор управляющих сообщений для блокировки и периодического опроса сети на предмет освобождения маркера одним их процессов. Использовать механизм

блокировки на запись в то время, как чтение будет выполняться всеми процессами, достаточно проблематично, так как исходная сеть Петри тогда становится избыточной. Необходимо моделировать процесс перехода маркера, в том числе и для операций чтения разделяемого ресурса или блока программного кода. Наиболее применимыми для РВС являются модели на основе передачи маркера и распределения управляющих сообщений между всеми узлами системы.

В качестве альтернативы ДВССП для решения задач стохастической синхронизации процессов рассмотрим инструменты теории массового обслуживания в сочетании с моделями Рикарта-Агравала и Сузуки-Касами. Процесс синхронизации параллельных процессов может быть представлен в виде системы массового обслуживания (СМО). Основными характеристиками СМО будут являться: свойства потока требований, поступающего на вход системы, процедуры формирования и управления очередью, а также дисциплины обслуживания. Важную роль при использовании СМО в процессе синхронизации играет определение случайной величины τ, характеризующей время ожидания входа процессами в критическую секцию. Поток требований для систем синхронизации может быть неограниченным, он определяется общим числом процессов, которые используются внутри параллельной системы. Рассмотрим СМО с ожиданием (СМОО). В рамках СМОО для синхронизации целесообразно задание некоторого T_{max}^{s} для процессов, определяющего максимальный период ожидания синхронизации. Это необходимо для отслеживания возможности возникновения исключительных ситуаций, не допуская нарушение функционирования параллельной системы. Параметр T_{max}^{s} может принимать фиксированное значение или задаваться в виде случайной величины с некоторым законом распределения. Наряду с T_{max}^{s} для СМОО важное значение имеет параметр $T_{\text{обсл}}$, определяющий время обслуживания одной заявки. В большинстве случаев $T_{\text{обсл}}$ представляет собой случайную величину с показательным законом распределения $f_{T_{\text{обсл}}}(t) = \mu e^{-\mu t}$. Параметр μ обратно пропорционален среднему времени обслуживания заявки [135]

$$\mu = \frac{1}{\bar{T}_{\text{обсл}}}, \bar{T}_{\text{обсл}} = \mathbb{M}(T_{\text{обсл}}). \quad (4.18)$$

Для построения СМО в рамках решения задач синхронизации будем использовать пуассоновское распределение потока заявок $A(t) = 1 - \mathrm{e}^{\lambda \mathrm{t}}, \mathrm{t} \geq 0$. Система с данными характеристиками потока заявок и показательным законом времени обслуживания обладает марковским свойством M. Для задания СМО принято использовать трехзнаковое обозначение, в частности, для задания системы с множеством каналом обслуживания используется обозначение – $M|M|N$. Введем в рассмотрение случайную величину $N(t)$ для пуассоновского потока требований, характеризующую количество заявок, поступивших вплоть до момента времени t. Функция $N(t)$ будет изменять свое значение на единицу при поступлении каждого нового требования. Для $N(t)$ введем моменты скачков $\tau = (\tau_1, \tau_2, \dots, \tau_n)$, возникающих в процессе появления требований $h = (h_1, h_2, \dots, h_n)$. Поток требований можно задать в виде вектора $y = (y_0, y_1, \dots, y_{n-1})$, $y_k = \tau_{k+1} - \tau_k, k \geq 0$. Определим распределение для временных промежутков y в виде $P(y < t) = 1 - \mathrm{e}^{\lambda \mathrm{t}}$ и получим для моментов скачков τ следующее распределение [10]:

$$P(\tau_n < y) = 1 - e^{-\lambda y}\left(1 + \lambda y + \cdots + \frac{(\lambda y)^{n-1}}{(n-1)!}\right), \quad (4.19)$$

где λ – параметр (интенсивность) потока.

Из выражения (4.19), учитывая, что временные промежутки между требованиями y_n независимы [99,135], получаем искомое распределение вероятностей для $N(t)$

$$P_n(t) = P(N(t) = n) = \frac{(\lambda t)^n}{n!}, n \geq 0. \quad (4.20)$$

Величина $N(t)$ имеет пуассоновское распределение с параметром λt. Определим среднее число требований, соответствующее входному потоку $N(t)$, согласно формуле Литтла [122]

$$\mathbb{E}\big(N(t)\big) = \sum_{n=1}^{\infty} nP_n(t) = e^{-\lambda t}\sum_{n=1}^{\infty} n\frac{(\lambda t)^n}{n!} = \lambda t \quad (4.21)$$

Обозначим $\gamma(t)$ – фактическое число требований, поступивших на вход СМО

за время t. Интенсивность входного потока требований γt будет являться математическим ожиданием $\mathbb{E}\big(\gamma(\mathrm{t})\big)$. Для простейшего потока будет справедливо следующее равенство:

$$\gamma t = \mathbb{E}\big(\gamma(t)\big) = \lambda t. \tag{4.22}$$

Для любого произвольного стационарного потока будет справедливо неравенство:

$$\gamma \geq \lambda. \tag{4.23}$$

Для функции $P_n(t)$ стационарного потока без последействий на интервале длиной $(t + \Delta t)$ справедливо выражение:

$$P_n(t + \Delta t) = \sum_{i=1}^{n} P_i(t) P_{n-i}(\Delta t), \Delta t > 0, t > 0. \tag{4.24}$$

Для потока без последействия $P_0(t) = e^{-\lambda t}$, $P_0(\Delta t) = 1 - \lambda \Delta t + o(\Delta t), \Delta t \to 0$, формула (4.24) может быть приведена к следующему виду [3]:

$$P_n(t + \Delta t) = (1 - \lambda \Delta t) P_n(t) + \sum_{i=0}^{n-1} P_i(t) P_{n-1}(\Delta t) + o(\Delta t). \tag{4.25}$$

При $n > 0$ имеем

$$\frac{P_n(t + \Delta t) - P_n(t)}{\Delta t} = -\lambda P_n(t) + \sum_{i=0}^{n} \frac{P_i(\Delta t)}{\Delta t} P_{n-i}(t) + o(\Delta t). \tag{4.26}$$

Пологая $\Delta t \to 0$, ограничим значение параметра $i > 0$ и рассмотрим следующее выражение [216]:

$$\frac{P_i(\Delta t)}{\Delta t} = \frac{P_i(\Delta t)}{\varphi(\Delta t)} \frac{\varphi(\Delta t)}{\Delta t} \to \lambda a_n, \tag{4.27}$$

где a_n – некоторая постоянная, $\varphi(\Delta t)$ – вероятность того, что за промежуток времени Δt выполнится обработка хотя бы одного требования.

При условии существования производной $P_n'(t)$, выражение для ее вычисления будет иметь следующий вид [113]:

$$\frac{dP_n(t)}{dt} = -\lambda P_n(t) + \lambda \sum_{i=1}^{n} a_i P_{n-i}(t), i \geq 1. \tag{4.28}$$

С учетом того $P_0'(t) = -\lambda \mathrm{P}_0(t)$ можно определить систему уравнений Эрланга:

$$\begin{cases} \dfrac{df_1(t)}{dt} = \lambda a_1 f_0(t), \\ \dfrac{df_2(t)}{dt} = \lambda\big(a_1 f_1(t) + a_1 f_0(t)\big), \\ \qquad\qquad \cdots \\ \dfrac{df_k(t)}{dt} = \lambda\big(a_1 f_{k-1}(t) + a_2 f_{k-2}(t) + \cdots + a_k f_0(t)\big), \end{cases} \tag{4.29}$$

где $f_k(t) = \frac{P_k(\mathrm{t})}{e^{\lambda \mathrm{t}}}$, $P_k(0) = f_k(0) = 0$.

Для решения системы уравнений Эрланга рассмотрим метод, основанный на определении производящей функции [13]. Введем общее представление производящей функции

$$F(t,x) = \sum_{n=0}^{\infty} P_n(t) x^n. \tag{4.30}$$

Из выражения (4.30) следует, что для нахождения $P_n(t)$ необходимо построить производящую функция $F(t,x)$. Можно преобразовать выражение (4.28) за счет его умножения на параметр x^n и суммирования по всем значениям n, взятым из полуинтервала $n \in [0;\infty)$, получим

$$\begin{aligned} \frac{dF}{dt} &= -\lambda F + \lambda \sum_{n=1}^{\infty} x^n \sum_{i=1}^{n} a_i P_{n-i}(t) = \\ &= -\lambda F + \lambda \sum_{i=1}^{\infty} a_i \sum_{j=0}^{\infty} P_j(t) x^{i+j} = -\lambda F \\ &+ \lambda \sum_{i=0}^{\infty} a_i x^i \sum_{j=1}^{\infty} P_j(t) x^j. \end{aligned} \tag{4.31}$$

Для определения производящей функции $F(t,x)$ введем следующее обозначение:

$$F(x) = \sum_{i=1}^{\infty} a_i x^i. \tag{4.32}$$

Тогда выражение (4.31) для производной $F'(x)$ с учетом обозначения (4.32) будет иметь следующий вид [144]:

$$\frac{dF}{dt} = F\lambda(F(x) - 1). \tag{4.33}$$

Учитывая, что при любом значении x, $F(x,0) = P_0(0) = 1$, интегрируя относительно t выражение (4.33) имеем

$$F(t,x) = e^{\lambda t(F(x)-1)}. \tag{4.34}$$

Для производящей функции при любом t будет справедливо равенство

$$F(t,1) = \sum_{n=0}^{\infty} P_n(t)x^n = 1. \tag{4.35}$$

Тогда с учетом (4.34) имеем

$$S(1) = \sum_{i=0}^{\infty} a_i = 1. \tag{4.36}$$

Пусть дан стационарный поток без последействия. Убедимся, что производящая функция $F(t,x)$ будет задаваться на основе выражения (4.34). Обозначим $p_k(t)$ как вероятность поступления k требований за время t

$$p_k(t) = \sum_{n=0}^{\infty} \left(e^{\lambda t}\right)^n \frac{(\lambda t)^n}{n!} P_n(k), \tag{4.37}$$

где $P_n(k)$ – вероятностью поступления k требований за n моментов.

С учетом того, что при $\mathrm{k} < r$ вероятность $P_n(k) = 0$, выражение для производящей функции можно записать в следующем виде $F(t,x)$:

$$\begin{gathered} F(t,x) = \sum_{k=0}^{\infty} p_k(t)x^k = \sum_{n=0}^{\infty} e^{\lambda t} \frac{(\lambda t)^n}{n!} \rho(x,k), \\ \rho(x,k) = \sum_{k=0}^{\infty} P_n(k)x^k. \end{gathered} \tag{4.38}$$

Величина $\rho(x,k)$ представляет собой сумму случайных величин для каждой из которых ставится в соответствие производящая функция $F(x)$. Тогда на основе свойства производящей функции получим следующее выражение для случайной величины $\rho(x,k)$:

$$\rho(x,k) = \sum_{k=0}^{\infty} P_n(k)x^k = \big(F(x)\big)^n = \left(\sum_{i=1}^{\infty} a_i x^i\right)^n. \quad (4.39)$$

В результате получим

$$F(t,x) = e^{\lambda t(x-1)} = \sum_{n=0}^{\infty} \frac{\big(\lambda t F(x)\big)^n}{n!} e^{\lambda t} = e^{\lambda t(F(x)-1)}. \quad (4.40)$$

Следовательно, при $p_k = 0$ и $p_1 = 1$ производящая функция для простейшего потока требований будет иметь следующий вид:

$$F(t,x) = e^{\lambda t(x-1)} = \sum_{n=0}^{\infty} \left(e^{-\lambda t}\frac{(\lambda t)^n}{n!}\right) x^n. \quad (4.41)$$

Рассмотрим процесс обслуживания применительно к процедуре синхронизации ресурсов между отдельными процессами. Для этого важно определить: задержки синхронизации, связанные с ожиданием получения доступа к ресурсу, вероятности перехода в состояние синхронизации и выхода из него и дисциплину обслуживания заявок синхронизации в такой системе. В исследовании будем рассматривать систему ожидания смешанного типа, где максимальный период ожидания синхронизации T_{max}^s имеет фиксированное значение для всех процессов. Длительности обслуживания заявок, поступающих от разных процессов, будут независимыми случайными величинами, имеющими одинаковый закон распределения вероятностей. После превышения значения $\mathrm{T}_{\text{обсл.}} > T_{max}^s$ заявка на синхронизацию разделяемых ресурсов будет считаться необработанной, а СМО будет генерировать исключения для данного процесса. Для выбора наиболее оптимального значения T_{max}^s необходимо учесть ряд факторов, связанных с производительностью и загруженностью вычислительной системы. Интенсивность выходящего потока требований будет напрямую зависеть от параметра T_{max}^s и определяться как $\mu = 1/T_{\text{обсл.}}$. Распределение времени обслуживания будет соответствовать показательному закону

$$\varphi(\sigma) = \begin{cases} 1 - e^{-\mu\sigma}, \sigma \geq 0 \\ 0, \sigma < 0 \end{cases}, \quad (4.42)$$

где $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ интенсивности обслуживания для n заявок, поступающих на вход СМО системы синхронизации от разных процессов.

Для представления процесса обслуживания СМО можно использовать математический аппарат цепей Маркова. Однако стоит отметить, что классические цепи Маркова могут быть использованы для процессов, где входящий поток требований является Пуассоновским, в других случаях необходимо использовать метод вложенных цепей Маркова [146,206]. Рассмотрим представление СМО марковским процессом с пуассоновским входящим потоком. Понятие Марковского процесса было введено нами в главе 4. Запишем выражение для дискретного марковского процесса $\xi(t)$, характеризующего переход между временными моментами t_n и t_{n+1}

$$P(\xi(\,t_{n+1}) = P(\xi(\,t_{n+1}) = X_{n+1}|\xi(t_1) = X_1, \dots, \xi(t_n) = X_n) \\ = P(\xi(\,t_{n+1}) = X_{n+1}|\xi(t_n) = X_n). \tag{4.43}$$

Далее рассматривается СМО с показательным распределением времени обслуживания заявок. В таком случае будем иметь марковские процесс с дискретным множеством состояний и непрерывным временем.

Для задания марковского процесса необходимо определить два типа вероятностей – начальное распределение вероятностей $P(\xi(\,t_0) = j) = P_j(0)$, характеризующее момент времени t_0 и вероятность перехода между смежными временными моментами $P\big(\xi(\,t_{n+1})|\xi(\,t_n)\big)$. При моделировании процессов синхронизации будем рассматривать однородные марковские процессы. Запишем выражение для переходной вероятности, соответствующее однородному марковскому процессу [108,112]

$$P(\xi(\,t_{n+1}) = j|\xi(\,t_n) = i) = P_{ij}(\Delta t), \sum_j P_{ij}(\Delta t) = 1 \;\forall \Delta t, \tag{4.44}$$

где $\Delta t = t_{n+1} - \,t_n$ – разность между t_{n+1} и t_n, i, j – число требований в СМО для моментов t_n и $\mathrm{t_{n+1}}$.

Простейший граф переходов, соответствующий СМО с интенсивностями входящего и выходящего потоков заявок λ_m и μ_m, будет иметь вид, представленный на рисунке 4.10

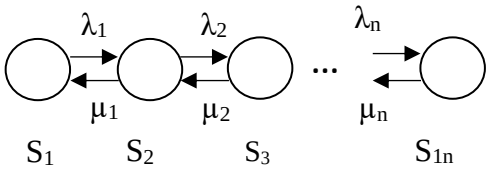

Рисунок 4.10. Граф интенсивностей переходов СМО, описываемой в виде марковского процесса

Первоначально параметр S_i будет определять начальное состояние графа переходов (канал свободен), S_j характеризует блокировку канала (вход в критическую секцию) и выполнение операций с разделяемым ресурсом [176]. Тогда вероятность перехода из исходного состояния S_i в состояние S_j в течение заданного интервала времени Δt запишем в следующем виде:

$$P_{ij}(t+\Delta t) = P\big(S(t+\Delta t) = S_j \big| S(t) = S_i\big). \tag{4.45}$$

Использую марковское свойство для $S(t)$, вычисление вероятности $P_{ij}(t+\Delta t)$ можно произвести на основе уравнения Чепмена-Колмогорова

$$P_{ij}(t+\Delta t) = \sum_{k=0}^{\infty} P_{ik}(t) P_{kj}(\Delta t). \tag{4.46}$$

Тогда для $P_{ij}(t)$ и $P_{ij}(\Delta t)$ будут справедливы следующие условия:

$$P_{ij}(t) = \begin{cases} 0, i \neq j, \\ 1, i = j, \end{cases} \tag{4.47}$$

$$P_{ij}(\Delta t) = \begin{cases} \lambda_{ij}\Delta t, i \neq j, \\ 1 - \lambda_i \Delta t, i = j, \end{cases} \tag{4.48}$$

где λ_{ij} – интенсивность перехода из состояния S_i в состояние S_j, λ_i – выходная плотность для S_j.

Матрица интенсивностей будет иметь следующий вид:

$$\lambda_{ij} = \begin{pmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1m} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{n1} & \lambda_{n2} & \dots & m \end{pmatrix}. \tag{4.49}$$

Интенсивность перехода λ_{ij}, характеризующая переход между состояниями S_i и S_j, будет иметь следующий вид:

$$\lambda_{ij} = \lim_{\Delta t \to 0} \frac{P_{ij}(t + \Delta t) - P_{ij}(t)}{\Delta t} = \frac{dP_{ij}}{dt} \tag{4.50}$$

Тогда с учетом условия (4.47) получим

$$\lambda_{ij} = \begin{cases} \lim\limits_{\Delta t \to 0} \dfrac{\mathrm{P_{ij}(t + \Delta t)}}{\Delta t} \geq 0, i \neq j, \\ \lim\limits_{\Delta t \to 0} \dfrac{\mathrm{P_{ij}(t + \Delta t)}}{\Delta t} \leq 0, i = j. \end{cases}$$

Из свойства переходных вероятностей $P_{ij}(\Delta t)$ с учетом (4.48) получим

$$\sum_{j \neq i} P_{ij}(\Delta t) = 1 - P_{ii}(\Delta t) = \sum_{j \neq i} \lambda_{ij} \Delta t + 1 - \lambda_i \Delta t = 1. \tag{4.51}$$

Следовательно

$$\begin{gathered} \lambda_i = \sum_{j \neq i} \lambda_{ij}, \\ \lambda_{ii} = -\lambda_i = -\sum_{j \neq i} \lambda_{ij}. \end{gathered} \tag{4.52}$$

Для функции $P_{ij}(t)$ с учетом условия (4.48) можно определить прямое и обратное дифференциальные уравнения Колмогорова

$$\frac{dP_{ij}(t)}{dt} = \sum_{k=0}^{\infty} \lambda_{ik} P_{ki}(t), \tag{4.53}$$

$$\frac{dP_{ij}(t)}{dt} = \sum_{k=0}^{\infty} P_{ik}(t) \lambda_{ki}. \tag{4.54}$$

Применяя нормировочное условие для вероятности состояний $P_{ik}(t)$ для любого момента времени t, можно исключить обратное дифференциальное уравнение. В таком случае прямое уравнение Колмогорова [199] можно записать с учетом выражения (4.52)

$$\frac{dP_{ij}(t)}{dt} = \sum_{k \neq j} P_{ik}(t) \lambda_{kj} - P_{ij}(t) \sum_{k \neq j} \lambda_{jk} \tag{4.55}$$

Отметим, что в системе (4.55) первая сумма связанна со значениями j для которых имеет место переход из состояний S_j в S_i,а вторая справедлива для тех j для которых возможен переход из S_i в S_j.

4.3. Разработка направлений повышения эффективности процессов синхронизации вычислений в распределенных вычислительных системах

Рассматривая СМО с точки зрения моделирования систем синхронизации, возникает ряд вопросов, связанных с производительностью данной системы. В первую очередь возникновение задержек, связанных с блокировкой разделяемых ресурсов, а также ограниченностью очереди заявок ресурсными характеристиками вычислительной системы (число вычислительных ядер, объем оперативной памяти, пропускная способность). Сложность моделирования систем синхронизации заключается в том, что в некоторых случая нельзя точно определить закон распределения времени обработки заявок и, как следствие, классическая теория представления СМО в виде марковского процесса становится неприменима. Отметим, что решение задач синхронизации в терминах СМО, моделируемой с помощью марковского процесса, является обоснованным только в том случае, если представленная распределенная система является полностью однородной – все элементы такой системы имеют эквивалентную производительность. На практике обеспечение условия однородности не всегда выполнимо, следовательно, время обслуживания заявок синхронизации зачастую будет носить произвольный, а не показательный закон распределения. Впервые задача моделирования СМО с общим законом распределения, а также расчет ее вероятностных характеристик представлены в работах Кендалла [142]. Наибольший интерес для представления СМО с произвольных законом распределения времени представляют методы «псевдосостояний» и вложенных цепей Маркова. Основная идея данных методов заключается к сведению случайного процесса к марковскому. В таком случае из исходного процесса выбираются только те элементы, которые могут образовывать цепь Маркова.

Рассмотрим метод вложенных цепей Маркова применительно к СМО с произвольным распределением временем обслуживания заявок [141]. Пусть мы имеем систему обслуживания заявок синхронизации, поступающих от отдельных процессов с пуассоновским потоком требований и произвольным распределением

времени обслуживания. Введем следующие обозначения $T = (t_1, t_2, \dots, t_k)$ для моментов обслуживания требований, имеющих произвольный закон распределения времени обслуживания $\varphi(t)$. Рассмотрим представление данного процесса в виде вложенной марковской цепи. Обозначим $\omega(\mathrm{t}_k)$ – количество заявок в моменты времени $\mathrm{t}_k \geq 0$. Тогда $\{\omega(\mathrm{t}_k)\}$ будет образовывать случайный процесс с дискретным временем

$$X = (X_0 = 0, X_1 = \omega(t_1), \dots, X_k = \omega(t_k)), k \geq 1. \quad (4.56)$$

Последовательность (4.56) будет являться цепью Маркова, исходя из того, что на входе СМО имеем пуассоновский поток с интенсивностью λ. Рассмотрим справедливость данного утверждения. Положим, что состояния данного процесса характеризуются требованиями, оставшимися после обслуживания текущей заявки, а переходы описываются моментами времени, когда завершено обслуживание заявки. Если в момент времени t_k состоянии процесса было i то переход в состояние j выполняться с вероятностью $P_{ij}(\mathrm{t_k})$ и процесс, описанный выражением (4.56), будет являться марковским процессом, так как для X_k будет справедливо следующее выражение [139]:

$$X_{k+1} = \begin{cases} X_k - 1 + K, X_k \geq 1 \\ K,\ X_k = 0 \end{cases}, \quad (4.57)$$

где K – общее количество требований, поступающих на вход СМО за время T, в течение которого будет обслужено $k + 1$ требований.

В силу стационарности пуассоновского потока, случайный параметр K будет зависеть только от длительности обслуживания T и являться независимым от начального времени обслуживания t_0 и длины очереди заявок. В таком случае получаем справедливость утверждения, что случайный процесс $\{X_k\}$ будет являться марковским. Определим вероятностное распределение $P(K)$, используя следующую формулу:

$$P(K) = \int_0^\infty P(K = k|T = t)\, B(t)dt, \quad (4.58)$$

где $B(t)$ – плотность распределения времени обслуживания.

С учетом того, что общее число требований, поступающих на вход СМО за время t_k представляет собой простейший пуассоновский поток, будет справедливо следующее выражение [211]:

$$P(K = k|\, T = t) = e^{-\lambda t}\frac{1}{k!}(\lambda t)^k. \tag{4.59}$$

Тогда выражение для расчета вероятностей $P_{ij}(\mathrm{t_k})$ с учетом формул (4.58) и (4.59), можно записать в следующем виде:

$$\begin{gathered} k_j = P_{ij}(t_k) = P(X_{k+1} = i|X_k = j) = P(K = m) = \\ = \int_0^\infty P(K = m|T = t)B(t)dt = \int_0^\infty e^{\lambda t}\frac{(\lambda t)^m}{m!}B(t)dt, \\ j \geq i-1, i \geq 1, \\ P_{ij}(t_k) = 0, j < i-1, i \geq 1, \end{gathered} \tag{4.60}$$

где $m = j - i + 1$.

Производящая функция $K(z)$, соответствующая распределению $\mathrm{k_j}$, будет иметь следующий вид:

$$K(z) = \sum_{j=0}^{\infty} k_j z^j \tag{4.61}$$

Матрицу переходов можно записать через вероятности k_j, описывающие вероятность того, что за время обработки одного требования поступит j новых требований

$$P(t_k) = \begin{bmatrix} k_0 & k_1 & k_2 & \dots \\ k_0 & k_1 & k_2 & \dots \\ 0 & k_0 & k_1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{4.62}$$

Получим выражение для расчета параметра нагрузки СМО

$$\rho = \sum_{n=1}^{\infty} n\, k_n = \lambda \int_0^\infty tB(t)dt = \lambda\, \mathbb{E}\big(B(t)\big). \tag{4.63}$$

Из выражения (4.62) следует, что данная цепь является неприводимой и непериодической. При $\rho < 1$ она являться эргодической, при $\rho = 1$ возвратно-

нулевой, $\rho > 1$. В таком случае при $\rho < 1$ для вложенной цепи Маркова существует эргодическое распределение $\bar{P}_j$

$$\bar{P}_j = \sum_{i=1}^{\infty} P_{ij}\,\bar{P}_i. \tag{4.64}$$

Следовательно

$$\begin{gathered} \bar{P}_0 = k_0(\bar{P}_0 + \bar{P}_1), \\ \bar{P}_j = \bar{P}_0 k_j + \sum_{i=1}^{j+1} \bar{P}_i\, k_{j-i+1}, j = 2,3, \dots, n \end{gathered} \tag{4.65}$$

Найдем производящую функцию $\bar{P}(z)$

$$\bar{P}(z) = \sum_{i=1}^{\infty} \bar{P}_i z^i. \tag{4.66}$$

Умножая записанное выше выражение на z^i и суммируя по i, с учетом выражения (4.61) получим

$$\begin{aligned} \bar{P}(z) = \bar{P}_0 \sum_{j=0}^{\infty} k_j z^j + \sum_{j=0}^{\infty} z^j \sum_{i=1}^{j+1} \bar{P}_i k_{j-i+1} = \bar{P}_0 \bar{P}(z) + \frac{K(z)}{z}(\bar{P}(z) - \bar{P}_0) \\ = \frac{\bar{P}_0(1-z)K(z)}{K(z)-z}. \end{aligned} \tag{4.67}$$

Формула (4.67) определяет производящую функцию стационарного распределения с точностью до параметра $\bar{P}_0$. Следовательно

$$K(1) = \sum_{i=1}^{\infty} i k_i = 1. \tag{4.68}$$

Тогда дробь, стоящая в выражения (4.67), стремится при z → 1 к величине $1 - K'(1)$, $K'(1)$ – среднее число требований, поступающих в процессе обслуживания одного заявки на синхронизацию. Следовательно

$$\begin{gathered} \lim_{z\to 1} \bar{P}(z) = \lim_{z\to 1} K(z) = 1, \\ \lim_{z\to 1} \frac{K(z)-z}{1-z} = \lim_{z\to 1} \left(\frac{K(z)-1}{1-z} + \frac{1-z}{1-z} \right) = 1 - K'(1) = 1 - p. \end{gathered} \tag{4.69}$$

С учетом выражения (4.67) при z → 1 получим

$$\bar{P}_0 = 1 - p \tag{4.70}$$

Производящая функция стационарного распределения имеет вид [68]

$$\bar{P}(z) = \frac{(1-p)K(z)(z-1)}{K(z)-z} = \frac{(1-p)(1-z)}{1-\frac{z}{B^*(\lambda-\lambda z)}}, \tag{4.71}$$

где B^* – преобразование Лапласа-Стилтьеса для функции $B(t)$, $1-p$ – стационарная вероятностью, когда система обслуживания будет свободна.

Выражения (4.71) представляет собой формулу Поллачека-Хинчина, задающей стационарное распределения для вложенной цепи Маркова в виде функции от преобразования Лапласа-Стилтьеса B^*, соответствующего заданной функции распределения длительности обслуживания требований $B(t)$.

Рассмотрим возможные способы оптимизации СМО с учетом решения задач синхронизации. На сегодняшний момент наиболее изучены вопросы оптимизации входящего потока требований за счет решения экстремальных задач, а также группировки требований. Рассмотрим особенности СМО, реализующие принцип группового обслуживания, также возможные способы применения таких СМО в рамках решения задач синхронизации. Принцип группового обслуживания заключается в формировании групп по r – требований. В таком случае на вход обслуживающего блока будет поступать n групп требований размерностью r. Если на входе СМО число требований меньше размерности группы r, то обслуживающий элемент ожидает до тех пор, пока наберется полная группа требований. Лишь после этого группа направляется на обслуживание. Отметим, что обслуживание каждой группы будет осуществляться за случайное время, исходя из закона распределения. В соответствии с законом распределения входного потока и времени обслуживания, система с групповым обслуживанием будет эквивалентна системе с эрланговым потоком входящих требований. Как показывает практика, ожидание формирования полной группы требований размером r не является оптимальным. Наиболее рационально использовать обработку любого числа требований, накопившихся до текущего момента времени даже, если оно не превысило значение размерности группы r. Обозначим $E = (E_0, E_2, E_3, \dots, E_r)$ – множество состояний, характеризующих процесс обработки групповых требований. Тогда для всех

состояний E_k, за исключением состояния E_0, согласно диаграмме интенсивностей, приведенной на рисунке 4.10, переход будет возможен из соседнего состояния E_{k-1} и из всех состояний $E_{1:r}$, но только в случае завершения процесса группового обслуживаний. В тоже время переход из состояния E_k возможет только при поступлении новых заявок или когда обработка группы заявок в рамках состояния E_k завершена. Что касается начального состояния E_0 переход в него возможен из любого состояния $E_{1:r}$ с интенсивностью μ_{k-1}, а выход из E_0 возможен только в том случае, когда на вход СМО поступают новые заявки. Системы уравнений для расчета вероятностей $\mathrm{P}_k(t)$ можно записать в следующем виде [143]:

$$(\lambda+\mu)P_k(t) = \mu P_{k+1}(t) + \lambda P_{k-1}(t), k \geq 1,$$
$$\lambda P_0(t) = \mu \sum_{i=1}^{r} P_i(t). \tag{4.72}$$

Используя метод производящих функций $P(z)$, умножая на параметр z^k и суммируя по всем $k \in (1; r)$, выражение (4.72) перепишем в следующем виде:

$$(\lambda+\mu)\big(P(z) - P_0(t)\big) = \frac{\mu}{z^r}\left(K(z,t) - \sum_{k=0}^{r} P_k(t)z^k\right) + \lambda z P(z). \tag{4.73}$$

Решая уравнение (4.73) относительно параметра $P(z)$ получаем промежуточное значение для производящей функции

$$\bar{P}(z) = \frac{\mu\sum_{k=1}^{r} P_k(t)z^k - P_0(t)z^r(\lambda+\mu)}{\lambda z^{r+1} - z^r(\lambda+\mu) + \mu} \tag{4.74}$$

С учетом (4.72) и (4.74), можно получить

$$-P_0(t)z^r(\lambda+\mu) = -\mu z^r \sum_{i=0}^{r} P_i(t). \tag{4.75}$$

Выражение (4.74) может быть переписано в следующем виде [122]:

$$K(z,t) = \frac{\mu\sum_{k=1}^{r-1} P_k(t)(z^k - z^r)}{r\alpha z^{r+1} - (1+\alpha\lambda)z^r + 1} \tag{4.76}$$

где $\alpha = \mu r/\lambda$, $\bar{t} = 1/\mu$ – среднее время обслуживания r требований.

Далее рассмотрим процедуру упрощения выражения (4.76). Для этого положим, что среди корней в знаменателе будет присутствовать $r-1$ корней в

пределах области $|z| < 1$, один внутри единичной области $|z| = 1$ и один внутри области $|z_0| > 1$. В числителе производящая функция $P(z)$ будет ограничена областью $|z| < 1$. В таком случае $r - 1$ корней числителя будут совпадать с корнями знаменателя в пределах области $|z| < 1$. Тогда числитель и знаменатель будут пропорциональны

$$\frac{K\sum_{k=1}^{r-1} P_k(t)(z^k - z^r)}{1-z} = \frac{r\alpha z^{r+1} - (1+\alpha\lambda)z^r + 1}{(1-z)\left(1-\frac{z}{z_0}\right)}. \tag{4.77}$$

Сокращая числитель и знаменатель в выражении (4.76), и используя равенство (4.77), имеем

$$\bar{P}(z) = \frac{1}{K\left(1-\frac{z}{z_0}\right)}, \tag{4.78}$$

где K – константа.

С учетом $P(1) = 1$, вычисляем значение константы K и получаем

$$\bar{P}(z) = \frac{1-\frac{1}{z_0}}{1-\frac{z}{z_0}}. \tag{4.79}$$

Используя обратное преобразование, запишем искомое выражение для распределение числа заявок

$$P_k(t) = \left(1-\frac{1}{z_0}\right)\left(\frac{1}{z_0}\right)^k. \tag{4.80}$$

Из выражения (4.80) видно, что система с групповым обслуживанием заявок является частным случаем марковских СМО и, как следствие, для определения параметров таких систем может быть использован математический аппарат цепей Маркова, представленный нами в параграфе 4.2.

Далее рассмотрим способ получения распределения $P_k(t)$ для групповых требований с произвольным распределением времени обслуживания. Расчет вероятностей $P_k(t)$ возможен с использованием метода вложенных цепей Маркова. Рассмотрим процедуру применения данного метода для расчета вероятностных характеристик групповых требований. Пусть $\Delta t_k = \{\Delta t_1, \Delta t_2, \dots, \Delta t_k\}$ множество

случайных величин, определяющих временные промежутки между поступлением двух групп требований, имеющих следующее распределение вероятностей:

$$P(\Delta t_k \le t) = 1 - e^{\lambda x}, \lambda > 0. \tag{4.81}$$

Время, за которое будут поступать k групп требований можно представить в виде суммы всех моментов времени, предшествующих k, включая сам срез k

$$a_k = \sum_{i=1}^{k} \Delta_{i.} \tag{4.82}$$

Выразим общее число требований, поступающих в интервале $(0, t)$ через параметр C_k, определяющий общее число требований, содержащихся в k группах

$$\begin{gathered} A(0,t) = \sum_{a_i \in (0,t)} \mathrm{C}_i, 0 \le i < k, \\ P(\mathrm{C}_k = j) = c_j. \end{gathered} \tag{4.83}$$

Тогда

$$P(A(0,t) = k) = a_k(t) = \sum_{n=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} C_k^n, \tag{4.84}$$

где C_k^i – свертка для $\{C_k\}$.

Введем производящую функцию $K(z,t)$

$$\begin{gathered} K(z,t) = \sum_{k=0}^{\infty} a_k(t) z^k = e^{-\lambda t(1-c(z))}, \\ c(z) = \sum_{j=0}^{\infty} c_j z^j. \end{gathered} \tag{4.85}$$

Состояние системы с групповым обслуживанием будет определятся двумя параметрами – число требований системы в момент покидания ее требования n и непосредственно временем ухода требования n. Данная последовательность событий будет являться марковской цепью. Тогда для вероятности $P_{ij}^n(t)$, характеризующей переход системы из моментов времени t_0 в $t + t_0$, описывающих состояния до и после ухода требования n, будет справедливо выражение Чепмена-Колмогорова

$$P_{ij}^{n+1}(t) = \sum_{n=-1}^{j-1} \int_0^t P_{i,j-n}^{n}(t-x) a_{n+1}(x) dB(x), \tag{4.86}$$

Решения данного уравнения можно получить, используя производящую функцию

$$\bar{P}(z) = \sum_{j=1}^{\infty} p_j z^j = \frac{(1-p)(1-z)B^*\left(\lambda\left(1-c(z)\right)\right)}{B^*\left(\lambda\left(1-c(z)\right)\right) - z}, \tag{4.87}$$

где B^* – преобразование Лапласа-Стилтьеса для функции $B(t)$.

Математическое ожидание общего количества заявок, обрабатываемых системой за время t, имеет следующий вид:

$$\mathbb{E}_k = \rho + \frac{\rho}{2(1-\rho)}\left(\left(\frac{\beta_1}{\beta_2} - 1\right) + \rho\left(1 + \frac{\sigma^2}{\mathbb{E}_1^2}\right)\right), \tag{4.88}$$

где $\rho = \lambda\beta_1\mu_1$ – параметр загрузки СМО, $\mathbb{E}_1$ и $\mathbb{E}_2$ – математические ожидания времени обслуживания, соответствующие первому и второму моменту, $\sigma^2 = \mathbb{E}_2 - \mathbb{E}_1$, β_1 и β_2 – первый и второй моменты распределения $\{C_k\}$.

Тогда распределение времени ожидания обработки заявки будет иметь преобразование Лапласа-Стилтьеса следующего вида:

$$W(t) = (1-\rho)\left(1 + \lambda\frac{1-K(z,t)}{S - \lambda\left(1-K(z,t)\right)}\right). \tag{4.89}$$

Выражения (4.89) может быть упрощено и приведено к формуле Поллачека-Хинчина, если положить, что требования, входящие в состав группы k, будут покидать систему одновременно

$$W(t) = \frac{1-\rho}{1 - \lambda\left(\frac{1-K(z,t)}{K(z,t)}\right)}. \tag{4.90}$$

Тогда среднее время ожидания равно

$$\bar{t} = \frac{\rho}{2(1-\rho)}\left(\frac{\beta_1}{\beta_2} + \frac{\sigma^2}{\mathbb{E}_1^2}\right)\mathbb{E}_1. \tag{4.91}$$

А дисперсия определяется следующим выражением:

$$\mathbb{D}_W = \frac{\rho}{(1-\rho)^2}(\rho C_1^2 + (1-\rho)C_2) \tag{4.92}$$

где C_1 и C_2 первый и второй моменты распределения $H(t)$

$$H(t) = \frac{1}{\mathbb{E}_1 \beta_1} \times \frac{1 - K(z,t)}{K(z,t)} \tag{4.93}$$

Наряду с групповой обработкой требований, альтернативным подходом к оптимизации СМО является использование статистической оптимизации. Основной целью статистической оптимизации является определение наиболее оптимальных параметров моделей СМО на основе проведения ряда статистических испытаний. Для решения задач оптимизации выбираются критерии оптимальности. В качестве таких критериев выбираются функционалы, позволяющие определить эффективность процессов, моделируемых с помощью СМО. В качестве одного из показателей эффективности СМО будем использовать функцию $\Phi(X)$

$$\Phi(X) = \Phi(x_1, x_2, \dots, x_k, \dots, x_{k'}), \tag{4.94}$$

где x_k – параметр рассматриваемой системы, $k = 1,2, \dots, k, \dots, k'$ – количество параметров x_k.

Тогда для определения критерия эффективности системы необходимо определить вектор параметров

$$X' = \left(x_1', x_2', \dots, x_k', \dots, x_{k'}'\right), \tag{4.95}$$

позволяющий получить экстремальное значение показателя $\Phi(X)$

$$\Phi(X') = \underset{X}{extr}\left(\Phi(X)\right). \tag{4.96}$$

Область определения функции $\Phi(X)$ может быть задана неравенством $x_k^{min} \leq x_k \leq x_k^{max}$.

В работах Лифшица [156] рассматривается возможность применения для решения задачи оптимизации простейших алгоритмов случайного поиска, в частности, алгоритма наказания за случайность (НС). Алгоритм наказания за случайность можно применять как в случаях, когда критериальная функция $\Phi(X)$ задается аналитическим способом, так и вычисляется методом Монте-Карло. Рассмотрим более детально данный алгоритм. Полагаем, что мы имеем функцию $\Phi(X)$ от k' переменных, заданных в области $x_k^{min} \leq x_k \leq x_k^{max}, k = 1, \dots, k'$ с дополнительными ограничениями

$$R_\lambda(X) < b_\lambda, \lambda = 1,2,\dots,\lambda', \tag{4.97}$$

где b_λ – граничное значение, $R_\lambda(X)$ – налагаемое ограничение.

В процессе выполнения алгоритма происходит пошаговое перемещение в случайном направлении ξ. Если при переходе в новую позицию $i+1$, значение $\Phi(X_{i+1})$ будет больше (при поиске максимума функции) $\Phi(X_i)$, то происходит повторное смещение в направлении ξ на шаге $i+2$, в противном случае, система возвращается в искомое состояние, а следующий шаг делается в другом случайном направлении. Цикл алгоритма повторяется до того момента, пока не будет достигнут экстремум функции $\Phi(X)$. Этапы алгоритма НС можно представить в следующем виде

$$X_{i+1} = X_i + \Delta X_{i+1},$$
$$\Delta X_{i+1} = \begin{cases} \Delta X_i \; \Phi(X_i) < \Phi(X_{i+1}) \\ -\Delta X_i + a\xi, \Phi(X_i) \geq \Phi(X_{i+1})' \end{cases} \tag{4.98}$$

где a – величина шага смещения, ξ – случайный вектор значений, i – номер шага.

Одним из механизмов работы с функцией $\Phi(X)$ является изучение работающей СМО в течение некоторого промежутка времени T и выявление общего числа требований n, количества обработанных требований m, а также длительности периода занятости $t_з$. Под $t_з$ будем понимать сумму длительностей обслуживания всех требований за промежуток $t_з$. Пусть $v(t)$ – число требований, характеризующих момент времени t. С учетом того, что рассматриваются система с пуассоновским распределением времени обслуживания, $v(t)$ можно представить в виде марковского процесса с непрерывным временем и дискретным набором состояний $k = (1,2\dots)$. Для любого состояния $k > 0$ распределение времени пребывания в данном состоянии будет носить экспоненциальный характер с параметром $\lambda + \mu$ и параметром λ для начального состояния $k = 0$. Вероятность перехода из состояния k в состояние $k+1$ будет равна $P(X_{k+1}|X_k) = \frac{\lambda}{\lambda+\mu}$.

В работах Риордана [198] впервые затрагиваются вопросы представления критериальной функции $\Phi(X)$ в виде функции правдоподобия для СМО, что дает возможность применения различных алгоритмов поиска, в том числе алгоритма

наказания за случайность. В процессе определения $\Phi(X)$ в виде функции правдоподобия L, необходимо ввести оценки интенсивностей входящего потока $\lambda = n/\, t_з$ и потока обслуживания (выходящего потока) $\mu = m/\, t_з$. Для получения оценок методом максимального правдоподобия необходимо произвести обучение на основе случайной выборки за фиксированный интервал времени. На практике такие вычисления являются достаточно трудоемкими. В связи с чем для упрощения алгоритма получения функции правдоподобия, в частности, для стационарного случая, необходимо ввести следующее ограничение для длительности наблюдения τ СМО, а именно, чтобы она зависела от характера производимых наблюдений (наблюдение происходит для ранее определенного периода занятости СМО $t_з$). В процессе расчета оценок можно использовать параметр времени занятости СМО $t_з$, вместо общего времени τ, затраченного на обработку заявки. В процессе проведения наблюдений необходимо определить следующий набор параметров СМО: $v(t)$ – число требований, характеризующих начальное состояние системы, m – общее число требований, обработанных СМО за период $t_з$, t – временной срез, когда $i = 1,2, \dots, m$ требований покидают систему после обработки, общее число требований n, поступивших на вход СМО для последующей обработки, $n_з = n - n_0$ – число требований, полученных в момент обслуживания. Для рассматриваемого случая $v(t)$ является марковским процессом с непрерывным временем и дискретным числом возможных состояний $\{0,1,2, \dots\}$. Первоначально полагаем, что система будет находится в состоянии равновесия $p < 1$, тогда v будет иметь показательное распределение с параметром $p = \lambda/\mu$. В следствие того, что распределение для начального числа требований системы $v(0)$ является стационарным, получим

$$P_k = (1 - p)p^k, k = 1,2, \dots. \quad (4.99)$$

При условии того, что длительность периода занятости τ задана, выходной поток требований имеет распределение $\frac{e^{-\mu\tau}(\mu\,\tau)^m}{m!}$, то входной поток в интервале времени $(0; T)$ также имеет пуассоновское распределение $\frac{e^{-\lambda T}(\lambda\, T)^n}{n!}$.

Для определения функции правдоподобия $L(\lambda,\mu)$ необходимо задать ряд параметров: распределение вероятностей потока требований размерностью n_0, соответствующее начальному моменту времени, распределение для поступающих s и покидающих систему m требований. С учетом введенных входных параметров, условное правдоподобие L' можно определить в виде функции от интенсивностей потоков λ и μ

$$L'(\lambda,\mu) = (\lambda+\mu)^{n_з+m} e^{-(\lambda+\mu)n_з} \lambda^{n_0} e^{-\lambda t_0} \left(\frac{\lambda}{\lambda+\mu}\right)^{n_з} \left(\frac{\mu}{\lambda+\mu}\right)^{m}. \quad (4.100)$$

С учетом того, что поток требований, поступающий на вход СМО в начальный момент времени t_0, будет иметь нулевое значение, формулу (4.100) можно привести к следующему виду:

$$L'(\lambda,\mu) = \lambda^n \mu^m e^{-\lambda t-\mu t_з}. \quad (4.101)$$

Формула (4.101) описывает условную функцию правдоподобия, однако на практике для получения статистических оценок параметров СМО необходимо определить безусловное правдоподобие $L(\lambda,\mu)$. Одним из важных параметров, необходимых для получения $L(\lambda,\mu)$ является распределение вероятностей для случайного процесса $v(t)$, соответствующего начальному моменту времени $P_k(v(0)=k)$. Можно определить значение безусловной функции правдоподобия для начального состояния системы с учетом формулы (4.101), когда в ней находилось k требований $L(\lambda,\mu) = P_k(v(0)=k)L'(\lambda,\mu)$. Полагая, что система будет находится в стационарном состоянии, начальное распределение $v_0 = v(0)$ будет представлять собой геометрическое распределение (4.99). Запишем искомое выражения для функции правдоподобия

$$L = L(\lambda,\mu) = (1-p)\lambda^{n+v_0}\mu^{m-v_0} e^{-\mu T-\lambda T}. \quad (4.102)$$

Используя функцию правдоподобия $L(\lambda,\mu)$, можно получить оценки методом максимального правдоподобия для интенсивностей λ и μ. Полагаем, то интенсивности λ и μ ограничены областью $0<\lambda<\mu$. Тогда оценками максимального правдоподобия будут являться значения параметров, при которых $L(\lambda,\mu)$ будет принимать максимальное значение. Такие оценки могут быть получены путем решения двух дифференциальных уравнений относительно λ и μ

$$\frac{dL'(\lambda,\mu)}{d\lambda} = \frac{d\,ln\,L'(\lambda,\mu)}{d\lambda} = 0,$$
$$\frac{dL'(\lambda,\mu)}{d\mu} = \frac{d\,ln\,L'(\lambda,\mu)}{d\mu} = 0. \quad (4.103)$$

В развернутой форме система (4.103) имеет вид

$$\frac{n}{\lambda} - t = 0, \frac{m}{\mu} - T = 0. \quad (4.104)$$

Тогда оценки правдоподобия для интенсивностей λ и μ без учета начальной информации $v(t)$ имеют следующий вид:

$$\widehat{\lambda'} = \frac{n}{t}, \widehat{\mu'} = \frac{m}{T}. \quad (4.105)$$

Для повышения точности получения оценок для λ и μ рассмотрим случай, учитывающий начальную информацию о распределении $v(t)$. Из формулы (4.102), в силу условия $p < 1$ очевидно, что $0 < \lambda < \mu$, а значение $L = 0$. Следовательно, функция правдоподобия $L(\lambda,\mu)$ будет достигать максимального значения во внутренней точке $(\hat{\lambda},\hat{\mu})$

$$\hat{\lambda} = (\hat{\mu} - \hat{\lambda})(n + v_0 - \hat{\lambda}t),$$
$$\hat{\mu} = (\hat{\lambda} - \hat{\mu})(m - v_0 - \hat{\mu}t). \quad (4.106)$$

Разделим первое выражение в формуле (4.106) на второе и получим значения оценок интенсивностей λ′ и μ′ в следующем виде [20]:

$$\hat{\lambda} = \frac{n+m}{t_з + \hat{p}t},$$
$$\hat{\mu} = \frac{n+m}{t_з + \hat{p}t}\hat{p}, \hat{p} = \frac{\hat{\lambda}}{\hat{\mu}}. \quad (4.107)$$

Оценку максимального правдоподобия для p можно получить из выражения (4.106) в следующем виде:

$$f(p') = t(\hat{p})^2(m - v_0 - 1) - \big(t(m - v_0) + T(n + v_0 + 1)\big)p' + \\ + t_з(n + v). \quad (4.108)$$

Из выражения (4.108) следует, что $f(0) = (n + v_0)t_з$ и $f(1) = -(t_з + t)$. Тогда область значений переменных функции $f(p')$ будет определена в интервале $(0;1)$. Для получения приближенных значений p' заменим множители $(m - v_0 -$

1) и $(n+v_0+1)$ на соответствующие значения $m-v_0$ и $n+v_0$. Тогда, решая уравнение (4.108), получим два корня – первый из которых p_1 будет равен единице, а второй p_2 будет иметь следующее значение:

$$\begin{aligned} p_1 &= 1, \\ p_2 &= \frac{(n+v_0)t_з}{(m-v_0)t}. \end{aligned} \tag{4.109}$$

Если ограничить область значения корня p_2 в виде интервала $(0;1)$ и подставляя в формулу (4.108) значение коря p_2, получаем следующее выражение:

$$(p_2-p')(1-p)(m-v_0) = p'\left(p'+\frac{t_з}{t}\right). \tag{4.110}$$

Из выражения (4.110) получим следующие оценки для параметра p':

$$0 < p_1 - p' < \frac{1+p'}{1-p'}\frac{p'}{m-v_0}. \tag{4.111}$$

Точные оценки для $\hat{\lambda}$ и $\hat{\mu}$ с учетом корня уравнения p_2 можно получить в следующем виде:

$$\begin{aligned} \hat{\lambda} &= \frac{(n+m)p'}{p'(t+t_з)}, \\ \hat{\mu} &= \frac{n+m}{p'(t+t_з)}. \end{aligned} \tag{4.112}$$

Если значения оценки $\hat{p} < p_2$, разность $p_1 - p'$ значительно мала и p_1 будет являться соответствующей аппроксимаций для p', то производя замену p' на p' в выражении (4.107), можно получить оценки интенсивностей $\hat{\lambda}$ и $\hat{\mu}$ в следующей приближенной форме [20]:

$$\begin{aligned} \hat{\lambda} &\approx \frac{n+v_0}{t}, \\ \hat{\mu} &\approx \frac{m-v_0}{t_з}, \\ \hat{p} &= \frac{(n+v_0)t_з}{(m-v_0)t}. \end{aligned} \tag{4.113}$$

Из формулы (4.113) следует, что с увеличением параметра t вклад, вносимый начальными параметрами, будет постепенно уменьшаться. На практике можно повысить точность аппроксимации оценок $\hat{\lambda}, \hat{\mu}$ так как в формуле (4.113) параметр

$n + v_0/t$ и $m - v_0/t_з$ являются несмещенными оценками. Следовательно, математическое ожидание для них может быть записано в следующей форме:

$$\mathbb{E}\frac{n + v_0}{t} = \frac{\mathbb{E}n + \mathbb{E}v_0}{t} = \lambda + \frac{p}{(1-p)t}. \quad (4.114)$$

$$\mathbb{E}\frac{m - v_0}{t_з}, = \frac{\mathbb{E}m - \mathbb{E}v_0}{t_з} = \mu - \frac{p}{(1-p)t_з}. \quad (4.115)$$

Выражения для оценок $\hat{\lambda}, \hat{\mu}$ можно записать с учетом смещений $n + v_0/t$ и $m - v_0/t_з$ соответственно

$$\hat{\lambda} \approx \hat{\lambda}_c + \frac{1}{t}\left(v_0 - \frac{\hat{p}_c}{1-\hat{p}_c}\right), \hat{\lambda}_c = \frac{n}{t},$$

$$\hat{\mu} \approx \hat{\mu}_c + \frac{1}{t_з}\left(v_0 - \frac{\hat{p}_c}{1-\hat{p}_c}\right), \hat{\mu}_c = \frac{m}{t_з}, \quad (4.116)$$

$$\hat{p}_c = \frac{\hat{\lambda}_c}{\hat{\mu}_c},$$

где $\hat{\mu}_c, \hat{\lambda}_c$ – оценки методом максимального правдоподобия, полученные без учета начальных данных.

Наряду с показательным распределением времени обслуживания требований для СМО необходимо также сформировать методику определения оценок на основе максимального правдоподобия для СМО с произвольным распределением временем обслуживания. Для получения таких оценок воспользуемся описанным нами ранее методом вложенных цепей Маркова. Рассмотрим вложенную цепь Маркова и введем следующие обозначения: $\mu_i = \sum_{i \neq i} \lambda_{ij}$ – интенсивность выхода из состояния i, λ_{ij} – интенсивность перехода между состояниями i и j, $P_{ij} = \frac{\lambda_{ij}}{\mu_i}$ переходная вероятность. Предположим, что интенсивность входного потока требований λ_{ij} для рассматриваемого процесса $v(t)$ будет зависеть от параметра $\theta = (\theta_1, \theta_2, \dots, \theta_n)$, а именно $\lambda_{ij} = \lambda_{ij}(\theta)$. Пусть t_k представляет собой множество наблюдаемых значений случайной величины ξ_k, характеризующей время перехода между состояниями вложенной цепи. Тогда цепочку наблюдаемых событий можно представить в следующем виде:

$$S = \{(Xi\ _o, t_o), (X_1, t_1), (X_2, t_2), \dots, (X_{n-1}, t_{-1}), X_n\}. \quad (4.117)$$

где $X = X_o, X_1, X_2, \dots, X_{n-1}, X_n$ – состояния вложенной цепи Маркова.

Следовательно, функцию правдоподобия $l(\theta)$ для выборки S можно записать в следующем виде [138]:

$$\begin{aligned} L(\theta) = \mu_0(\theta) e^{-\mu_0(t) t_o} P_{0,1}(\theta) \mu_1(\theta) e^{\mu_1(t) t_1} \times \dots \\ \times P_{n-2,n-1}(\theta) \mu_{n-1}(\theta) e^{\mu_{n-1}(t) t_{n-1}} P_{n-1,n}(\theta) = \\ = \prod_{k=0}^{n-1} \lambda_{k,k+1}(\theta) e^{\mu_k(t) t_k}. \end{aligned} \tag{4.118}$$

Пусть n_{ij} – общее число переходов из состояния i в j, τ – время нахождения процесса в состоянии i. Тогда выражение для функции правдоподобия (4.118) приведем к следующему виду:

$$l(\theta) = \prod_{i \neq j}^{n-1} \lambda_{ij}^{n_{ij}}(\theta) \exp\left(- \sum_i \mu_i(\theta)\, \tau \right). \tag{4.119}$$

Поскольку $P_{ij} = \frac{\lambda_{ij}}{\mu_i}$, имеем

$$l(\theta) = \prod_{i \neq j}^{n-1} \lambda_{ij}^{n_{ij}}(\theta)\, e^{-\mu_{ij}(\theta)\tau}. \tag{4.120}$$

Логарифмируя обе части равенства (4.120) получим [175]

$$L(\theta) = \ln l(\theta) = \sum_{i \neq j} \left(n_{ij} \ln \lambda_{ij}(\theta) - \tau\, \lambda_{ij}(\theta) \right). \tag{4.121}$$

Из выражения (4.121) получим искомое выражение для определения оценки $\hat{\theta}$ методом максимального правдоподобия

$$\hat{\theta} = arg\, \max_{\theta} L(\theta). \tag{4.122}$$

По аналогии с оценками для СМО $M/M/N$ для оптимизации логарифма правдоподобия в СМО с произвольным распределением временем обслуживания $M/G/N$ могут использоваться различные методы и алгоритмы для решения задач локального поиска и определения максимального значения для функции правдоподобия параметров λ_{ij} и μ_{ij}.

Наряду с получением оценок максимального правдоподобия, можно использовать алгоритм локального поиска на основе Ньютона-Рафсона [4,132],

впервые предложенного Лифшицом для решения задач статистического моделирования СМО. Для оптимизации процедур поиска экстремума интерес представляют квазиньютоновские алгоритмы, в частности, алгоритм Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ). В качестве оценочной функции будем использовать функцию правдоподобия $L'(\lambda, \mu)$. Для упрощения процедур формирования матриц Якоби и Гессе можно воспользоваться уравнением (4.103) для получения оценок интенсивностей $\hat{\lambda}$ и $\hat{\mu}$.

Решение задачи поиска для итерации, соответствующей состоянию системы $i + n$, запишется в виде выражения для метода Ньютона-Рафсона

$$\begin{pmatrix} \lambda_{i+n} \\ \mu_{i+n} \end{pmatrix} = \begin{pmatrix} \lambda_{i+n-1} \\ \mu_{i+n-1} \end{pmatrix} + H^{-1} \times J,$$

$$H = \begin{pmatrix} \dfrac{d^2 \ln L'(\lambda,\mu)}{d\lambda^2} & \dfrac{d^2 \ln L'(\lambda,\mu)}{d\lambda d\mu} \\ \dfrac{d^2 \ln L'(\lambda,\mu)}{d\mu d\lambda} & \dfrac{d^2 \ln L'(\lambda,\mu)}{d\mu^2} \end{pmatrix}, \qquad (4.123)$$

$$J = \begin{pmatrix} \dfrac{d \ln L'(\lambda,\mu)}{d\lambda} \\ \dfrac{d \ln L'(\lambda,\mu)}{d\mu} \end{pmatrix}.$$

Для решения задачи оптимизации поиска экстремума функции правдоподобия, определим метод БФГШ. Данный метод является итеративным, где каждый шаг основывается на разложении искомой функции в полином следующего вида для состояния x_k:

$$m_k(x_k + \gamma) = f(x_k) + \nabla f^T(x_k)\gamma + \frac{1}{2}\gamma^T H(x_k)\gamma\,, \qquad (4.124)$$

где $H_k(x)$ – матрица Гессе, соответствующая итерации x_k, $\nabla f(x_k)$ – градиент функции $f(x_k)$.

Из формулы (4.124) следует, что $m_k(x_k + \gamma)$ и ее градиент при $\gamma = 0$ будут принимать значения функции $f(x_k)$ и $\nabla f(x_k)$ соответственно. Также видно, что каждая итерация требует перерасчет гессиана, что накладывает значительные ресурсные и временные ограничения. В методе БФГШ предполагается вместо прямого вычисления матрицы $H(x_k)$ использовать приближенное значение $B(x_k)$.

Это позволяет осуществлять расчет гессиана исключительно для первой итерации. В дальнейшем значение гессиана будет вычислено приближенно. В таком случае можно определить значение γ для каждой k итерации

$$\gamma_k = -B^{-1}(x_k)\nabla f(x_k). \tag{4.125}$$

Для определения выражения для шага при переходе к x_{k+1} требуется выполнение условия Вольфе [90]

$$\begin{gathered} f(x_k + \alpha_k\lambda_k) \le f(x_k) + c_1\alpha_k\nabla f_k^T(x_k)\lambda_k \\ \nabla f(x_k + \alpha_k\lambda_k)^T \ge c_2\nabla f_k^T(x)\lambda_k, \end{gathered} \tag{4.126}$$

где c_1 и c_2 – константы, выбираемые в соответствии с условием $0 < c_1 < c_2$.

Другой вариант условия Вольфе [91] предполагает, что приближение функции $f(x_k + \alpha_k\gamma_k)$ на каждой итерации находится в пределах локального минимума (максимума)

$$\begin{gathered} f(x_k + \alpha_k\gamma_k) \le f(x_k) + c_1\alpha_k\nabla f_k^T(x)\lambda_k \\ |\nabla f(x_k + \alpha_k\lambda_k)^T| \le c_2\left|\nabla f_k^T(x)\lambda_k\right|. \end{gathered} \tag{4.127}$$

Как правило, выбор констант c_1 и c_2 для формул (4.126) и (4.127) осуществляется из небольшой окрестности нуля.

Каждая следующая итерация x_{k+n} будет формироваться с учетом условия Вольфе

$$x_{k+n} = x_k + \alpha_k\gamma_k, \tag{4.128}$$

где α_k – длина шага итерации, удовлетворяющая условию Вольфе (4.126).

Запишем выражение для следующей $k+n$ итерации $m_{k+n}(\gamma)$ по аналогии с (4.124) за исключением того, что заменим матрицу Гессе на соответствующую приближенную матрицу $B_{x_{k+n}}$

$$F_{k+n}(x_{k+n} + \gamma) = f(x_{k+n}) + \nabla f^T(x_{k+n})\gamma + \frac{1}{2}\gamma^T B_{x_{k+n}}\gamma \tag{4.129}$$

В соответствии с формулой (4.129), на каждом шаге необходим пересчет матрицы $B_{x_{k+n}}$ с учетом матрицы $B_{x_{k+n-1}}$, полученной на предыдущей итерации. При этом полагаем, что исходная матрица, будет равна единичной $B_{x_0} = I$. Для получения $B_{x_{k+n}}$ на основе имеющихся данных, полученных на предыдущих

итерациях, необходимо, чтобы значение градиента $m_{k+n}(\gamma)$ было равно градиенту функции $f(x_{k+n})$ как минимум на двух смежных итерация $\nabla f(x_{k+n})$ и $\nabla f(x_{k+n-1})$. Подставляя в выражение (4.129) $\gamma = 0$, получаем значение $\nabla F_{k+n-1}(x_{k+n-1}) = \nabla f(x_{k+n-1})$ для первой итерации. Для шага $k+n$ значение ∇F_{k+n} можно определить из следующего выражения:

$$\nabla F_{k+n}(-\alpha_{k+n-1}\gamma_{k+n-1}) = \nabla f(x_{k+n}) - \alpha_{k+n-1}B(x_{k+n})\gamma_{k+n-1} = \nabla f(x_{k+n}). \quad (4.130)$$

Перепишем выражение (4.130) через разность градиентов

$$B_{x_{k+n}}\alpha_{k+n-1}\gamma_{k+n-1} = \nabla f(x_{k+n}) - \nabla f(x_{k+n-1}). \quad (4.131)$$

Далее введем следующие обозначения: $s_{k+n-1} = x_{k+n} - x_{k+n-1}$ и $y_{k+n-1} = \nabla f(x_{k+n}) - \nabla f(x_{k+n-1})$. С учетом введенных обозначений выражение (4.131) может быть преобразовано к уравнению секущих

$$B_{x_{k+n}}s_{k+n-1} = y_{k+n-1}. \quad (4.132)$$

В таком случае при сдвиге s_{k+n-1} и изменении значения градиентов y_{k+n-1} необходимо выполнение условия выпуклости функции $f(x_{k+n})$

$$y_{k+n-1}^T s_{k+n-1} > 0 \quad (4.133)$$

Из неравенства (4.133) следует, что если функция $f(x)$ является выпуклой, то оно будет справедливо для любых смежных точек x_{k+n} и x_{k+n-1}. В реальных условиях для обеспечения выполнения условия выпуклости функции, необходимо соблюдение условий Вольфе (4.126) или (4.127). Докажем справедливость данного утверждения. Из равенств $s_{k+n-1} = x_{k+n} - x_{k+n-1}$ и $y_{k+n-1} = \nabla f(x_{k+n}) - \nabla f(x_{k+n-1})$ с учетом коэффициента c_2 и строгого неравенства Вольфе получаем следующие неравенства:

$$\nabla f(x_{k+n})^T s_{k+n-1} > \mathrm{c}_2 \nabla f(x_{k+n-1})^T s_{k+n-1},$$
$$y_{k+n-1}^T s_{k+n-1} \geq (\mathrm{c}_2 - 1)\alpha_{k+n-1}\lambda_{k+n-1}\nabla f(x_{k+n})^T \quad (4.134)$$

С учетом того, что $\mathrm{c}_2 < 1$ и значения γ_{k+n-1} будет находиться в направлении спуска функции, то неравенство (4.133) имеет место.

Необходимо определить приближенную матрицу $B_{x_{k+n}}$. Для этого мы предполагаем, что для каждой матрицы $B_{x_{k+n}}$, формируемой для любой итерации,

справедливо уравнение секущих (4.132). Сформулируем задачу следующим образом:

$$\min_{B}\left|B_{x_{k+n}} - B_{x_{k+n-1}}\right|. \tag{4.135}$$

Первоначально идея расчета $B_{x_{k+n}}$ была сформулирована в алгоритме Девидона-Флетчера-Пауэлла (ДФП) и в дальнейшем расширена в алгоритме БФГШ. В таком случае определим значения приближенной матрицы $B_{x_{k+n}}$ для каждой итерации согласно алгоритму ДФП [210]

$$\begin{aligned} B_{x_{k+n}} = (I - \delta_{k+n-1}s_{k+n-1}y_{k+n-1}^T)B_{x_{k+n-1}}(I - \delta_{k+n-1}y_{k+n-1}s_{k+n-1}^T) \\ + \delta_{k+n-1}s_{k+n-1}s_{k+n-1}^T, \end{aligned} \tag{4.136}$$

где $\delta_{k+n-1} = 1/y_{k+n-1}s_{k+n-1}^T$.

Для оптимизации формулы (4.136) введем обозначение обратной матрицы $M_{x_{k+n}} = B_{x_{k+n}}^{-1}$. Используя тождество Шермана-Моррисона-Вудбери [79,80], можно получить формулу для расчета обратной матрицы для шага x_{k+n}

$$M_{k+n} = M_{k+n-1} - \frac{M_{k+n-1}s_{k+n-1}s_{k+n-1}^T M_{k+n-1}}{s_{k+n-1}^T M_{k+n-1}s_{k+n-1}} + \frac{y_{k+n-1}y_{k+n-1}^T}{y_{k+n-1}^T s_{k+n-1}}. \tag{4.137}$$

Выражения (4.136) и (4.137) описывают основной подход, представленный в квазиньютоновских методах, заключающийся в получении аппроксимации матрицы Гессе на основе данных о целевой функции взамен вычисления матрицы на каждой итерации. В свою очередь, значение обратной приближенной матрицы Гессе M_{k+n-1} можно выразить через соответствующую матрицу Якоби:

$$M_{k+n} = J_{k+n-1}^T J_{k+n-1} + M_{k+n-1} \tag{4.138}$$

Выражение (4.138) избавляет нас от необходимости вычисления вторых производных. В таком случае, если матрица M_{k+n-1} задана в соответствии с (4.138) и является положительно определенной, то и искомая матрица M_{k+n} для метода БФГШ, определяемая согласно формуле (4.137), будет также положительно определенной. Данное свойство может быть использовано для формирования асимптотических оценок функций, что является особенно важным с точки зрения определения направленности поиска в процессе выполнения метода БФГШ. В таком случае на конечных итерациях алгоритма БФГШ, когда значения матриц

Якоби J_{k+n-1} и J_{k+n-1}^T будут практически равными, сохраняется положительность определения матрицы M_{k+n-1} для каждого следующего шага, что доказывает корректность алгоритма. В свою очередь, на ранних итерация, когда J_{k+n-1} и J_{k+n-1}^T имеют совершенно разные значения, целесообразно определить дополнительные методы коррекции, так как знак матрицы $M_{k+n} = J_{k+n-1}^T J_{k+n-1} + M_{k+n-1}$ может быть неопределенным.

Метод ДФП является достаточно эффективным квазиньютоновским методом, его можно включить в качестве составляющей метода БФГШ. Метод БФГШ может быть преобразован к методу ДФП за счет трансформации выражения (4.136). Для получения искомого выражения для аппроксимации обратной матрицы H_{k+n} методом БФГШ необходимо ввести ограничения: матрица M_{k+n} должна быть симметричной и положительно определенной, M_{k+n} должна удовлетворять уравнению секущих $M_{k+n} y_{k+n-1} = s_{k+n-1}$. Исходя из этого, определим искомое выражение для расчета $M_{x_{k+n}}$ на основании метода ДФП [57]

$$M_{x_{k+n}} = (I - \delta_{k+n-1} s_{k+n-1} y_{k+n-1}^T) M_{x_{k+n-1}} (I - \delta_{k+n-1} y_{k+n-1} s_{k+n-1}^T) + \delta_{k+n-1} s_{k+n-1} s_{k+n-1}^T, \quad (4.139)$$

где $\delta_{k+n-1} = 1 / y_{k+n-1} s_{k+n-1}^T$.

Рассматривая описанный алгоритм БФГШ с практической стороны, видно, что для БФГШ необходимо хранение значения как минимум матриц для двух смежных итераций – текущей и предшествующей итерации. Это накладывает значительные ограничения на размер памяти ЭВМ, используемой для обработки данного метода. Для решения данной проблемы можно использовать версию БФГШ с ограниченной памятью. Основная идея данного подхода заключается в том, что в отличии от классического алгоритма БФГШ, алгоритм с ограниченной памятью хранит лишь фиксированный набор строк, используемых в аппроксимации $M_{x_{k+n}}$, для каждой из двух матриц. Наряду с ресурсными возможностями, использование алгоритма БФГШ с ограниченной памятью позволяет получить приемлемый уровень сходимости. Для описания алгоритма

определим каждый шаг формирования значений x_{k+n} в виде следующего выражения согласно критерию Вольфе

$$x_{k+n} = x_{k+n-1} + \alpha_{k+n-1} M_{k+n-1}. \quad (4.140)$$

Выражение для расчета обратной матрицы M_{k+n} будет получено из выражения (4.139)

$$M_{x_{k+n}} = (I - \delta_{k+n-1} s_{k+n-1} y_{k+n-1}^T)^T \times M_{x_{k+n-1}} \times (I - \delta_{k+n-1} y_{k+n-1} s_{k+n-1}^T) + \delta_{k+n-1} s_{k+n-1} s_{k+n-1}^T. \quad (4.141)$$

Из выражения (4.141) следует, что обновление обратной матрицы напрямую зависит от параметров y_{k+n-1} и s_{k+n-1}, агрегирующих приближенную информации относительно предыдущего состояния $M_{x_{k+n-1}}$. В таком случае отпадает необходимость хранения матрицы $M_{x_{k+n}}$, так как пары значений $\{y_{k+n-1}, s_{k+n-1}\}$ будет достаточно для восстановления любого предшествующего состояния матрицы $M_{x_{k+n-1}}$. В таком случае результирующее произведение $M_{x_{k+n-1}} \nabla f(x_{k+n-1})$ будет определяться значением градиента $\nabla f(x_{k+n-1})$, а также парой значений $\{y_{k+n-1}, s_{k+n-1}\}$, полученных на каждой итерации алгоритма. Больше нет необходимости хранить весь набор пар значений $\{y_{k+n-1}, s_{k+n-1}\}$. Каждая следующая пара будет перезаписывать предыдущую, что позволяет сократить затраты на хранения всего множества векторов, формируемых из $\{y_{k+n-1}, s_{k+n-1}\}$. Исходя из этого, результирующее значение $M_{x_{k+n}}$ может быть получено как сумма всех аппроксимаций относительно всех векторов, формируемых из значений пар $\{y_{k+n-1}, s_{k+n-1}\}$. Такой подход позволяет произвести аппроксимацию матрицы $M_{x_{k+n-1}}$ за $k + n - m$ шагов, используя для хранения лишь ограниченный набор векторов, сформированных из $\{y_{k+n-1}, s_{k+n-1}\}$ для текущей итерации.

Обобщенная формула аппроксимации матрицы Гессе на основе метода БФГШ с ограниченной памятью может быть записана с учетом выражения (4.141) в следующей форме:

$$M_{x_{k+n}} = (L_{k+n-1}^T, \dots, L_{k+n-m}^T) \times M_{x_0} \times (L_{k+n-1}, \dots, L_{k+n-m}) + \delta_{k+n-m}(L_{k+n-1}^T, \dots, L_{k+n-m+1}^T) \times s_{k+n-m}^T s_{k+n-m} \times (L_{k+n-1}, \dots, L_{k+n-m+1}) + (L_{k+n-1}^T, \dots, L_{k+n-m+2}^T) \times s_{k+n-m+1}^T s_{k+n-m+1} \times (L_{k+n-1}, \dots, L_{k+n-m+2}) + \dots + \delta_{k+n-1} s_{k+n-1} s_{k+n-1}^T, \quad (4.142)$$

где $L_{k+n-1} = I - \delta_{k+n-1} s_{k+n-1} y_{k+n-1}^T$.

В процессе вычисления обратной матрицы $M_{x_{k+n}}$ особый интерес представляет процедура расчета параметра ρ_{k+n}, определяющего направление поиска в процессе вычисления $s_{k+n} = \alpha_{k+n-1}\rho_{k+n}$. В отличие от классического метода, в БФГШ с ограниченной памятью, параметр ρ_{k+n} может быть выражен через градиент функции. Данное выражение может быть получено с использованием метода сопряженных градиентов на основе подхода Хестенеса-Штифеля (ХШ). Метод сопряженных градиентов ХШ для расчета ρ_{k+n} можно записать в следующем виде:

$$\rho_{k+n} = \frac{\nabla f(x_{k+n})^T (\nabla f(x_{k+n}) - \nabla f(x_{k+n-1}))}{\rho_{k+n-1} (\nabla f(x_{k+n}) - \nabla f(x_{k+n-1}))^T}. \quad (4.143)$$

Перепишем выражение (4.143) с учетом выражения (4.142) и получим искомую формулу расчета параметр ρ_{k+n} через метод сопряженных градиентов для алгоритма БФГШ с ограниченной памятью

$$\rho_{k+n} = -\nabla f(x_{k+n}) + \frac{\nabla f(x_{k+n})^T y_{k+n-1}}{\rho_{k+n-1} y_{k+n-1}^T} = -\left(I - \frac{s_{k+n-1} y_{k+n-1}^T}{y_{k+n-1}^T s_{k+n-1}}\right) \nabla f(x_{k+n}) = -M_{x_{k+n}} \nabla f(x_{k+n}) \quad (4.144)$$

Значения матрицы $M_{x_{k+n}}$, удовлетворяющей уравнению секущих, можно получить из уравнения (4.142)

$$M_{x_{k+n}} = L_{k+n-1}^T \times L_{k+n-1} + \delta_{k+n-1} s_{k+n-1} s_{k+n-1}^T, \quad (4.145)$$

где матрица $M_{x_{k+n}}$ – обратная матрица Гессе.

С учетом (4.144), а также полагая $s_{k+n-1} = M_{x_{k+n-1}} y_{k+n-1}$, можно определить условие ортогональности y_{k+n-1} и ρ_{k+n} [121]

$$y_{k+n-1}^T \rho_{k+n} = -y_{k+n-1}^T M_{x_{k+n}} \nabla f(x_{k+n}) = -y_{k+n-1}^T \nabla f(x_{k+n}). \quad (4.146)$$

Из формулы (4.146) следует, чтобы для ρ_{k+n} была характерна направленность спуска, необходимо выполнение следующего условия для скалярного произведения $y_{k+n+m}^{T} s_{k+n+m} > 0$. Также отметим, что формула ХШ для расчета параметра ρ_{k+n}, характеризующего направленность спуска может быть сведена к формуле Полака-Райбера, если положить значение произведения $\nabla f(x_{k+n})\rho_{k+n-1} = 0$. Из этого следует, что БФГШ с ограниченной памятью позволяет генерировать сопряженные по направлению градиенты, тем самым решаемая задача сводится к классической задаче одномерного поиска.

Если сравнивать БФГШ и БФГШ с ограниченной памятью, можно прийти к выводу, что каждый подход обладает своими преимуществами и недостатками. БФГШ с ограниченной памятью позволяет сократить ресурсы, связанные с хранением данных, однако требует дополнительного времени для формирования векторных пар значений $\{y_{k+n-1}, s_{k+n-1}\}$, в то время как классический метод БФГШ и так содержит в себе весь полный объем данных и не требует никаких дополнительных генераций. На практике используют тот или иной алгоритм в зависимости от размерности получаемой приближенной обратной матрицы Гессе M_{k+n-1}. Если размер данной матрицы невелик, то считаем целесообразным использовать классический алгоритм, он обладает наибольшей производительностью.

Существенной проблемой при использовании методов БФГШ и БФГШ с ограниченной памятью является то, что в некоторых случаях можно получить отрицательно определенную обратную матрицу Гессе M_{k+n-1}. В общем случае, в процессе выполнения БФГШ и БФГШ с ограниченной памятью полагаем, что для получения положительно определенной матрицы M_{k+n-1} необходимо выполнение следующего неравенства:

$$y_{k+n-1}^{T} s_{k+n-1} > 0. \qquad (4.147)$$

Однако на практике встречаются случаи, когда это неравенство не выполняется, в частности, при реализации методов БФГШ и БФГШ с ограниченной памятью на ЭВМ. Это в первую очередь связано с тем, что параметр ρ_{k+n} может продолжать соответствовать значению спуска, однако матрица M_{k+n-1} уже не

будет положительно определенной. С учетом того, что количество итераций алгоритмов БФГШ и БФГШ с ограниченной памятью может быть достаточно большим, сложно определить, когда именно матрица $M_{k+n-1} < 0$. В таком случае для предотвращения дальнейших погрешностей необходимо вернуться к исходной единичной матрицы, тем самым вся накопленная информация относительно кривизны функции $f(x)$ будет потеряна, а все итерации необходимо повторить заново. Для решения этой проблемы нами предполагается использование факторизации на основе разложения Холецкого. В общем случае число операций с использованием LDL^T разложения для матрицы $M_{x_{k+n}}$ будет сопоставимо со сложностью расчета искомой матрицы $M_{x_{k+n}}$. Вычисляя разложение матрицы $M_{x_{k+n}}$ с использования разложения Холецкого, можно получить оценки обусловленности. В частности, если определить d_{min} и d_{max} в качестве минимального и максимального диагональных элементов D_{k+n}, то можно получить следующее уравнение обусловленности для матрицы $M_{x_{k+n}}$:

$$cond(M_{x_{k+n}}) \geq d_{max}/d_{min}. \qquad (4.148)$$

В таком случае оценка числа обусловленности $cond(M_{x_{k+n}})$ на этапе завершения алгоритмов БФГШ и БФГШ с ограниченной памятью позволит дать заключение о допустимости найденного приближенного решения для обратной матрицы Гессе $M_{x_{k+n}}$. Используя факторизацию Холецкого, мы сможем гарантировать положительность матрицы $M_{x_{k+n}}$. Если значение диагональных элементов для D_{k+n} будут положительными, то и сама матрица $M_{x_{k+n}}$ будет положительно определенной. Следовательно, контроль D_{k+n} на каждой итерации методов БФГШ и БФГШ с ограниченной памятью позволит нам всегда знать знак матрицы и своевременно принять меры при потери ее положительной определенности. Такой подход позволит исключить повторный пересчет $M_{x_{k+n}}$ в том случае, если возникнет какая-либо погрешность в процессе выполнения БФГШ и БФГШ с ограниченной памятью, в том числе на ЭВМ.

Алгоритм БФГШ можно достаточно просто адаптировать для поиска оценок интенсивностей λ и μ. Запишем выражение для расчета обратной матрицы с

использованием метода БФГШ, математическое описание которого представлено в виде формулы (4.141)

$$M_{x_{k+n}} = (I - K \times \Delta X \times \Delta J^T)^T \times M_{x_{k+n-1}}$$
$$\times (I - K \times \Delta J \times \Delta X^T) + K \times \Delta X \times \Delta X^T,$$
$$\Delta X = \begin{pmatrix} \lambda_{i+n} \\ \mu_{i+n} \end{pmatrix} - \begin{pmatrix} \lambda_{i+n-1} \\ \mu_{i+n-1} \end{pmatrix}, \quad (4.149)$$
$$\Delta J = J_{i+n} - J_{i+n-1},$$
$$K = \frac{1}{\Delta J \times \Delta X^T}.$$

Рассмотренные нами модели синхронизации на основе применения СМО и алгоритмов Сузуки-Касами позволяют существенно расширить возможности обработки разделяемых данных распределенных вычислительных систем для решения различных прикладных задач. Использование в рамках решения задач обучения, вероятностного вывода ДБС позволяет оптимизировать данные процессы, повысить интенсивность обработки и порождения выборок без возникновения проблем, связанных с некорректным распределением разделяемых ресурсов между вычислительными процессами.

Предложенные подходы оптимизации, в частности, на основе статистического анализа параметров СМО, позволяют производить настройку системы с учетом специфики функционирования распределенных систем, провести корректировку основных параметров с целью достижения максимальной производительности и устойчивости функционирования СМО. Применение алгоритма БФГШ позволяет снизить риски получения ошибочных оценок для параметров СМО, в частности, оценок интенсивностей входного и выходного потока $\hat{\lambda}$ и $\hat{\mu}$, за счет исключения ситуаций попадания в локальные оптимумы функции правдоподобия. Такой подход позволяет повысить точность формирования оценок интенсивностей $\hat{\lambda}$ и $\hat{\mu}$. В результате мы получаем обоснованное использование систем обслуживания для моделирования сложных процессов синхронизации распределенных вычислительных систем, адаптированных для решения задач обучения и вероятностного вывода в ДБС.

4.4. Выводы

В четвертой главе работы:

1. По результатам анализа организации распределенных вычислительных систем, выработаны направления повышения эффективности процессов синхронизации в распределенных вычислительных системах.

2. В рамках оптимизации ресурсной эффективности разработан метод синхронизации на основе подхода Сузуки-Касами, базирующийся на применении инструментов теории массового обслуживания. Важной особенностью данного метода является возможность обеспечения условия строгой консистентности, что обеспечивает исключение возникновения взаимных блокировок между процессами.

3. Разработан подход к адаптации предложенных алгоритмов синхронизации для существующих распределенных систем с общей памятью на примере платформы Spark.

4. Разработана методика оценки показателей интенсивностей потоков для модели синхронизации на основе метода максимального правдоподобия. Реализация данной методики направлена на поиск наиболее оптимальных значений интенсивностей при распределении ресурсов между элементами вычислительной системы.

Глава 5. Вычислительный эксперимент по применению предложенных в работе моделей, методов и алгоритмов для тестирования основных классов ошибок веб-приложений

5.1. Используемые подходы распараллеливания вычислений в предложенных в работе методах обучения и вероятностного вывода

Большинство вычислительных и параллельных систем используют похожий набор принципов распараллеливания отдельных операций и процедур работы с переменными, передаваемые входе выполнения таких операций. Структурно каждый из возможных алгоритмов может рассматриваться в виде множества таких операций, выполняемых в определенном порядке. Параллелизм заключается в реализации одновременного выполнения данных операций, за счет их распределения между доступными ресурсами вычислительной системы или отдельного компьютера. Одной из особенностей построения алгоритмов для решения задач, связанных с использованием численных методов, является возможность разбиения множества входных данных на группы, которые потенциально могут быть обработаны одновременно. В большинстве случаев параллелизм численных методов связан с блочно-циклическим распределением этапов выполнения алгоритма, где каждый блок для каждой итерации цикла будет выполняться параллельно. Блочно-циклический подход может быть легко преобразован к классическому циклическому подходу, если определить число необходимых итераций равным единице. Применение блочно-циклического подхода позволяет упростить представление алгоритма в виде программного кода и оптимизировать балансировку загруженности процессора, что дает возможность повысить стабильность и интенсивность выполнения программы, использующей данный алгоритм. С точки зрения параллелизма возможно использование подхода MapReduce, реализованного в системе Spark – вся память представляет собой распределенное множество данных RDD. Наиболее оптимальным подходом является сокращение передачи синхронизирующих событий и реализация блочного разделения обучающих данных,

между параллельными блоками – каждый параллельный блок будет иметь ограниченную область видимости данных в рамках решения задач обучения. В текущей главе представлены параллельные алгоритмы обучения структуры, параметров ДБС, алгоритмы для оптимизации расчета параметров СМО с использованием блочных матричных алгоритмов, а также методы, направленные на решение задач вероятностного вывода, включая использование описанных в предыдущих главах оптимизационных механизмов.

Рассмотрим процедур оптимизации расчета матрицы Гессе, используемой для поиска оценок интенсивностей λ и μ в процессе синхронизации на основе выражения (4.149). С учетом того, что приближенную матрицу Гессе можно представить в виде $H^* = J \times J^T$, интерес представляет реализация параллельных алгоритмов умножения матриц. В свою очередь для расчета матрицы $M_{x_{k+n}}$ и достижения условия ее положительности $M_{x_{k+n}} > 0$ методом БФГШ с ограниченной памятью, используется факторизация на основе модифицированного метода Холецкого (3.30). В процессе расчета H^* нет необходимости хранить по отдельности матрицы J и J^T, можно вычислять матрицу J^T сразу же в процессе выполнения умножения и вычисления транспонированной матрицы $J_{mn} = J_{nm}^T$. Расчет результирующих компонентов h_{mn} матрицы H^* размерностью $m \times n$ может быть представлен в следующем виде:

$$h_{ij} = \sum_{k=0}^{n-1} J_{ik} \times J_{kj}^T = \sum_{k=0}^{n-1} J_{ik} \times J_{jk}\,, 0 \le i < m, 0 \le j < m. \qquad (5.1)$$

Из формулы (5.1) получаем, что каждый элемента матрицы Гессе h_{ij} может быть представлен в виде скалярного произведения [159]

$$h_{ij} = \left(J_i, \left(J_j^T\right)^T\right) = \left(J_i, J_j\right), J_i = \left(J_{i0}, J_{i1}, J_{i2}, \dots, J_{in-1}\right),$$
$$J_j^T = \left(J_{0j}, J_{1j}, J_{2j}, \dots, J_{n-1j}\right)^T. \qquad (5.2)$$

В дальнейшем будем полагать, что при разработке параллельных алгоритмов матрицы J и J^T будут квадратными размерностью $n \times n$. Если исходные матрицы не являются таковыми (имеют размерность $m \times n$), то можно осуществить приведение матриц за счет расширения выборки S путем создания ее копий θ,

число которых будет задаваться в соответствии с условием $\theta m \geq n$. Это гарантирует что обе матрицы J и J^T можно будет привести к квадратным. Исходя из выражения (5.2), операции умножения для элементов матриц J и J^T являются попарно независимыми, В таком случае их можно выполнять независимо друг от друга в параллельном режиме. Наиболее простым способом для обеспечения распараллеливания умножения матриц является разделение матрицы J на строки, после чего перемножение каждой строки на столбцы матрицы J^T выполняется параллельно, используя отдельное задание. С учетом того, что в процессе перемножения матриц получается набор однотипных операций, можно оптимизировать параллельный алгоритм за счет укрупнения блоков (обрабатывать несколько строк J в рамках одного параллельного процесса). Такой подход исключает формирование достаточно большой очереди заданий в рамках параллельного алгоритма и позволяет оптимально организовать вычисление в тех случаях, когда матрицы J и J^T имеют достаточно большой размер. Наряду с классическими (ленточными) алгоритмами перемножения матрицы, можно использовать различные блочные алгоритмы, позволяющие сократить время вычисления $T_{\text{общ}}$. Наиболее распространенными блочными алгоритмами являются алгоритмы Кэнона и Фокса. По аналогии с ленточным алгоритмом (5.2), будем рассматривать квадратные матрицы $A = J$ и $B = J^T$ размерностью $n \times n$. В блочных алгоритмах исходные матрицы J и J^T разбиваются на блоки, имеющие размерность $k \times k, k = n/s$, n – размерность исходных квадратных матриц J и J^T, s – число вертикальных и горизонтальных блоков. Исходя из этого, можно записать операцию блочного матричного умножения в следующем виде [72,116]:

$$\begin{pmatrix} A_{00}A_{01} \dots A_{0s-1} \\ A_{10}A_{11} \dots A_{1s-1} \\ \dots \\ A_{s-10}A_{s-11} \dots A_{s-1s-1} \end{pmatrix} \times \begin{pmatrix} B_{00}B_{01} \dots B_{0s-1} \\ B_{10}B_{11} \dots B_{1s-1} \\ \dots \\ B_{s-10}B_{s-11} \dots B_{s-1s-1} \end{pmatrix} = \begin{pmatrix} H_{00}H_{01} \dots H_{0s-1} \\ H_{10}H_{11} \dots H_{1s-1} \\ \dots \\ H_{s-10}H_{s-11} \dots H_{s-1s-1} \end{pmatrix}, \tag{5.3}$$

$$H_{ij} = \sum_{q=1}^{s} A_{iq} B_{qj} = \sum_{q=1}^{s} J_{iq} J_{qj}^{T} = \sum_{q=1}^{s} J_{iq} J_{jq}, i, i = 1, \dots, s.$$

В рамках реализации распараллеливания процедуры блочного умножения полагаем, что имеем p вычислительных процессоров, каждый из которых отвечает за вычисление блоков H при этом в качестве исходных данных выступают блоки A'_{ij} и B'_{ij}, формируемые из матриц A и B (по одному блоку из каждой матрицы). Доступ к остальным блокам будет формироваться с помощью передачи сообщений. Такой подход применяется для исключения дублирования данных (блоков матриц A и B) и повышает производительность алгоритма. Каждый процессор p_{ij} будет отвечать за решение подзадачи по расчету H_{ij}. В таком случае процессоры будут формировать квадратную решетку $s \times s$, соответствующую блочному представлению результирующей матрицы H. Рассмотрим последовательность выполнения алгоритма Фокса [12]. На этапе инициализации алгоритма определяется число процессоров, доступных в рамках вычислительной среды и осуществляется передача сообщений на каждый из процессоров p_{ij} с информацией относительно блоков A_{ij} и B_{ij}. Далее выполняется цикл по $0 \leq l \leq s$ в блоке которого реализуются следующие операции:

для каждой строки $1 \leq i \leq s$ из решетки процессоров $s \times s$ ставится в соответствие блок A_{ij} на той же строке i, индекс процессора m, задающий его положение в строке i решетки $s \times s$ определяется как $m = (i + l - 1) \bmod (\mathrm{s} + 1)$;

каждый блок A_{im} отправляется на выполнение на каждый элемент строки i, $A'_{ij} = A_{im}$;

полученные блоки A'_{ij} и B'_{ij}, соответствующие процессору p_{ij}, перемножаются и суммируются с H_{ij}

$$H_{ij} = H_{ij} + A'_{ij} \times B'_{ij};$$

полученные блоки B'_{ij}, ассоциируемые с p_{ij}, пересылаются в соседний сверху процессор p_{i-1j} (блоки, связанные с первой строкой процессорной решетки, пересылаются в процессоры, находящиеся в последней строке).

Общее время, необходимое для вычисления операции $A \times B$ в соответствии с алгоритмами Фокса, определяется в следующем виде:

$$T_{\text{общ}} = s\left(\frac{n^2}{p}\left(\frac{2n-1}{s-1}+1\right)\right)\tau, \tag{5.4}$$

где p – число параллельных процессоров, τ – время выполнения одной скалярной операции.

Далее рассмотрим алгоритм Кэннона. Отличительной особенностью данного алгоритма по сравнению с алгоритмом Фокса является выбор начального распределения блоков A'_{ij} и B'_{ij}. Число блоков выбирается в соответствии с доступным числом процессоров p в вычислительной системе. В таком случае процедура определения начального положения блоков A'_{ij} и B'_{ij} может быть выполнена без дополнительной передачи данных. Процедуру умножения матриц на основе алгоритма Кэннона можно представить в виде совокупности следующих операций:

на каждый процессор p_{ij} осуществляется рассылка блоков A_{ij} и B_{ij}, при этом результирующая матрица H_{ij} первоначально обнуляется;

для i строки и j столбца матрицы процессоров выполняется сдвиг матрицы A_{ij} на i позиций влево и на j позиций вверх;

через $\sqrt{p}$ операций получаем матрицу H_{ij} на каждом из процессоров p_{ij} и пересылаем эти блоки главному процессу для формирования результирующей матрицы H.

Рассмотренные блочные алгоритмы Кэннона и Фокса обладают хорошей производительностью и могут быть адаптированы для решения задач, связанных с перемножением матриц большой размерности за счет использования принципов блокового распределения и реализации подхода без непосредственного копирования всего содержимого матриц A и B в адресное пространство процесса, связанного с обработкой параллельного блока на процессоре. Использование данных алгоритмов для формирования MapReduce заданий распределенных вычислительных систем, в частности Spark, не представляет трудности, так как

блоки матриц будут хранится внутри RDD, а отдельные процессоры, выполняющие операции с блоками A_{ij} и B_{ij} будут использовать только фрагменты матриц A и B, что существенно повышает эффективность использованием данных алгоритмов в рамках создания параллельных программ на Spark. В свою очередь рассылка блоков A_{ij} и B_{ij} осуществляется внутри связанных блоков RDD, что существенно снижает затраты, связанные с распределением данных между узлами Spark, а RDD будет восприниматься процессорами в виде единого доступного адресного пространства.

Для реализации обучения и вероятностного вывода ДБС в рамках процесса тестирования требуется введение следующих требований, предъявляемых к разрабатываемым программам. Данные требования связаны с необходимостью взаимодействия модулей фаззинга и модуля ДБС, что позволяет осуществлять непрерывную коррекцию и настройку тестовых данных. Первое требование связано с тем, что параллельные алгоритмы должны быть адаптированы для решения задач онлайн обучения и вероятностного вывода, так как выполнение тестирования требует постоянной адаптации алгоритма, в частности, для учета переменных, появляющихся на новых срезах ДБС в процессе функционирования. Второе требование связано с решением задачи оптимизации хранения промежуточных данных, используемых в процессе проведения вычислений (модель перехода, восприятия, начальное распределение условных вероятностей, обучающая выборка). Это необходимо для обработки ДБС со сложной топологией и большим числом переменных. Для повышения эффективности алгоритма имитации отжига, а также в процессе реализации тестов на условную независимость на основе статистического критерия G^2 предполагается использование концепции разделения вычислительной задачи на ряд подпроцессов с целью достижения требуемой скорости алгоритма и интенсивности статистической обработки выборки, используемой в процессе обучения структуры и параметров динамической байесовской сети. В свою очередь обучение параметров динамических и статических байесовских сетей может быть выполнено с использованием тех же алгоритмов, а число временных срезов в данной ситуации ограничивается необходимой точностью получения

апостериорных данных. Обучение структуры подразумевает наличие транзитивных связей между временными срезами и необходимостью проведения тестов на условную независимость узлов, располагающихся в смежных временных срезах ДБС. Отметим, что в процессе решения задачи определения направленности связей между узлами ДБС, имеющими временные связи, предполагается, что мы имеем однонаправленную связь между узлами сети, находящихся на t и $t+1$ срезах. Для решения задач обучения разработан параллельный алгоритм МПСО, включающий в себя алгоритм имитации отжига.

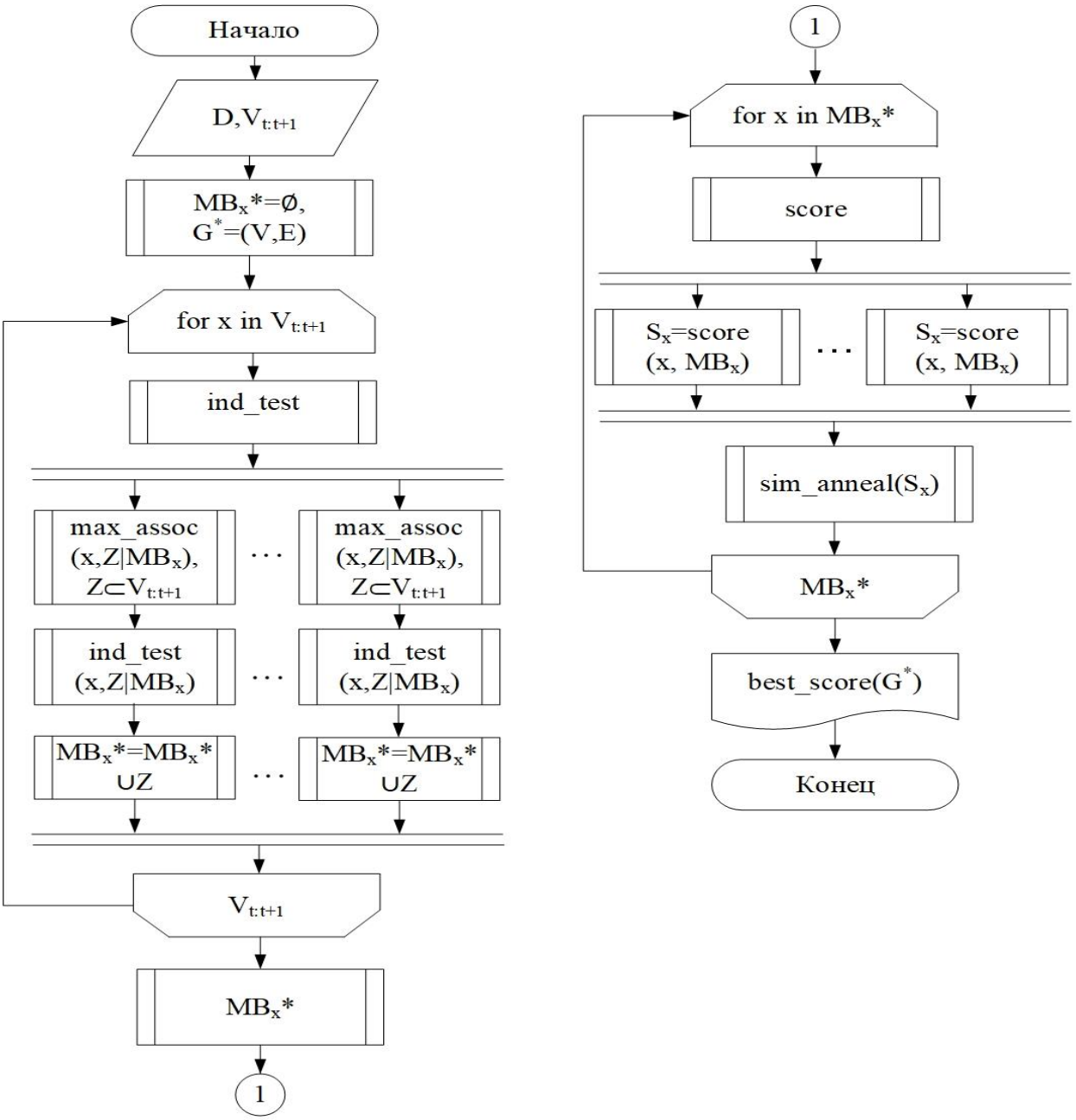

Рисунок 5.1. Структурная схема параллельного алгоритма обучения структуры ДБС на основе разработанного метода

Применительно к процедуре обучения параметров ДБС методом ОМ, алгоритм Гиббса позволяет обойти ряд ограничений, присущих базовому алгоритму ОМ. Выполнение каждого из этапов алгоритма ОМ в полной мере не позволяет учесть специфику динамической байесовской сети. Это приводит к снижению производительности алгоритма, так как необходимо производить расчет не только текущего состояния, но и состояний, взятых в некотором хронологическом порядке. Использование алгоритма Гиббса в рамках алгоритма ОМ позволяет оптимизировать первоначальную выборку параметров ДБС. В свою очередь одним из недостатков алгоритма Гиббса является необходимость генерации достаточно большого числа выборок для получения требуемой точности. Для оптимизации данных операций предполагается разработать и использовать параллельный алгоритм Гиббса. Данный алгоритм является частью параллельного алгоритма ОМ с использованием выборок Гиббса. Структура данного алгоритма разделена на два этапа. Первый предусматривает формирование начального распределения выборок для ДБС на основе метода Монте-Карло с применением цепей Маркова. На следующем этапе происходит обучение параметров сети с использованием метода ожидания максимизации. Отметим, что для оптимизации структуры ДБС будем использовать алгоритм построения дерева сочленения. В связи с тем, что основой ОМ является метод максимального правдоподобия, то в исследовании предполагается оптимизация данного метода за счет применения методологии бустинга, в частности, алгоритма AdaBoost, рассмотренного в главе 4.

Оптимизация алгоритма ОМ с использованием бустинга позволяет произвести подбор классификатора с целью минимизации ошибок для каждой из выборок, входящей в состав обучающего множества ДБС $S_i \subset D$. В результате получаем наиболее согласованный набор выборок. В качестве классификатора могут быть использованы следующие критериальные функции: логарифм максимального правдоподобия, эквивалентная метрика Байеса-Дирихле, критерии Шварца и Акаике. Параллельный алгоритм ожидания максимизации с

применением бустинга и выборок Гиббса [181], позволяющий оптимизировать процедуру обучения параметров ДБС представлен на рисунке 5.2

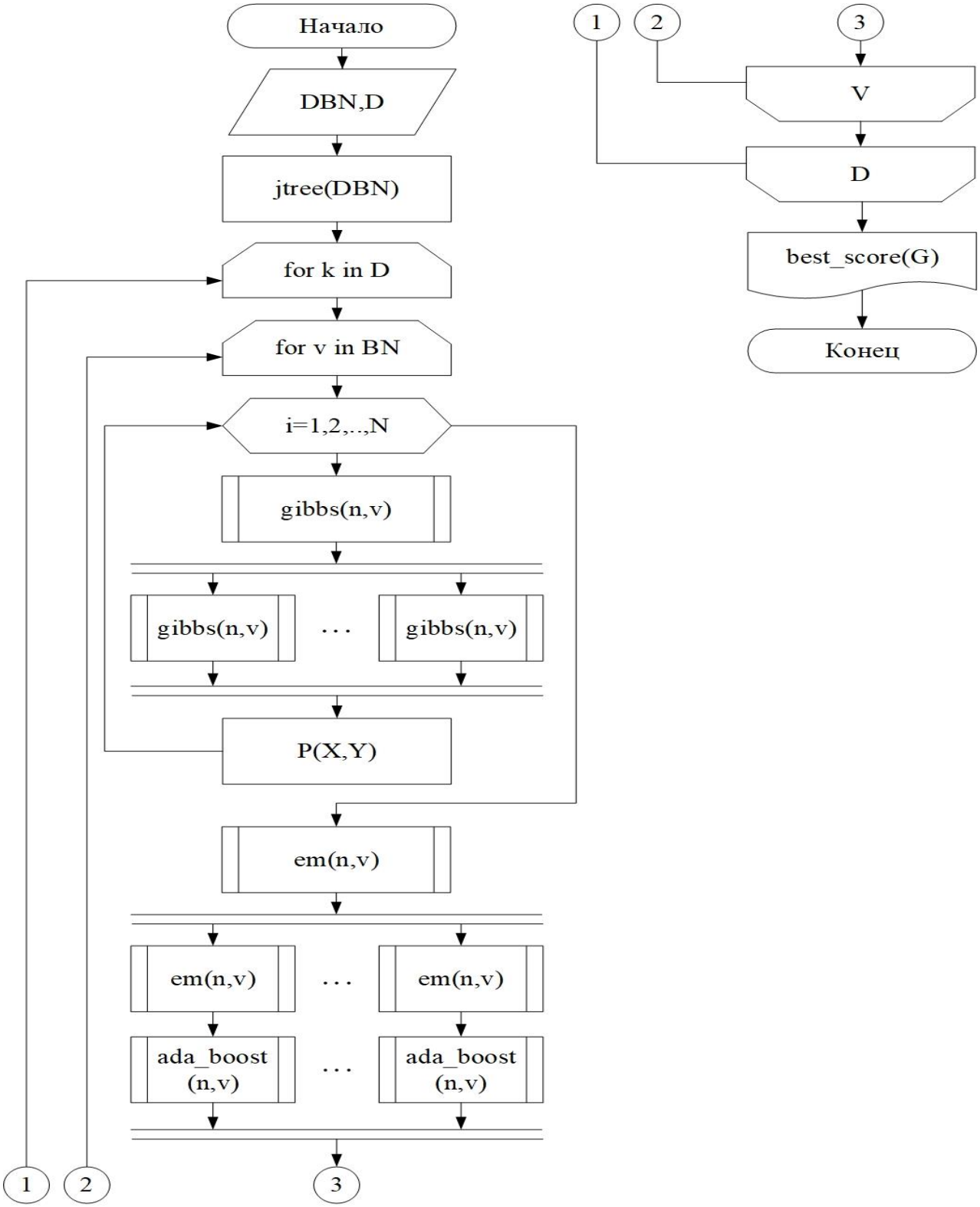

Рисунок 5.2. Структурная схема параллельного алгоритма обучения параметров ДБС на основе разработанного метода

Из рисунка 5.2 видно, что исходной структурой для алгоритма Гиббса является дерево сочленений (ДС), представляющее собой оптимизированную

топологию ДБС. Рассмотрим принципы создания параллельных алгоритмов ДС, алгоритмов ОМ, бустинга и Гиббса. Возможные пути распараллеливания алгоритма построения дерева сочленений, соответствующего произвольной ДБС связаны с необходимость оптимизации построения морального, триангулярного графа, а также дерева объединений. В таком случае распараллеливание алгоритма может быть сведено к разделению операций добавления, удаление и объединения узлов графа ДБС, а также реализацию параллельной процедуры опроса сети. Тогда формирование вышеописанных графов может быть разделено на блоки, соответствующие каждой из вершин сети или совокупности вершин, если сеть имеет достаточно разветвленную структуру. Формирование результирующих графов будет выполнятся на этапе операции отображения за счет объединения узлов и соответствующих им множеств родительских и дочерних вершин, построенных по результатам морализации и триангуляции графа исходной ДБС. В качестве основы для создания параллельного алгоритма для построения ДС может быть использована модель Pregel [51]. Данная модель представляет собой последовательность S итераций (блоков модели) и является разновидностью блочно-синхронной параллельной модели (БСПМ) [88]. В Pregel и в БСПМ используется синхронизация, ожидающая завершения выполнения всех параллельных блоков (блоков модели). Отметим, что в системах, построенных по модели RDD, синхронизация будет выполняется на стадии отображения, когда выполнен каждый из блоков модели. Это обеспечивает проведение расчетов каждым из блоков, а в случае возникновения сбоя, обеспечивает повторный перерасчет блока, завершившегося с ошибкой. В процессе выполнения блока i для каждого узла графа N формируется заранее определенная функция F, содержащая набор операций, которые необходимо выполнить для текущего узла. Такой набор операций может быть выполнен параллельно для каждого из узлов. Функция F описывает порядок взаимодействия вершины N и блока модели $S-1$, осуществляет чтение сообщений M_i, поступающих на вход N в процессе выполнения S, и рассылку сообщений M_{i+1} на другие узлы, которые будут обработаны блоком $S+1$. При этом алгоритм не гарантирует соблюдение порядка

отправки сообщений от узла, инициирующего передачу. Очередь сообщений для каждого узла формируется в порядке получения сообщений целевым узлом. Алгоритм лишь гарантирует отсутствие дубликатов сообщений, даже в случае возникновения ошибки или исключения [12]. Реализация модели Pregel с использованием Spark RDD представлена на рисунке 5.3

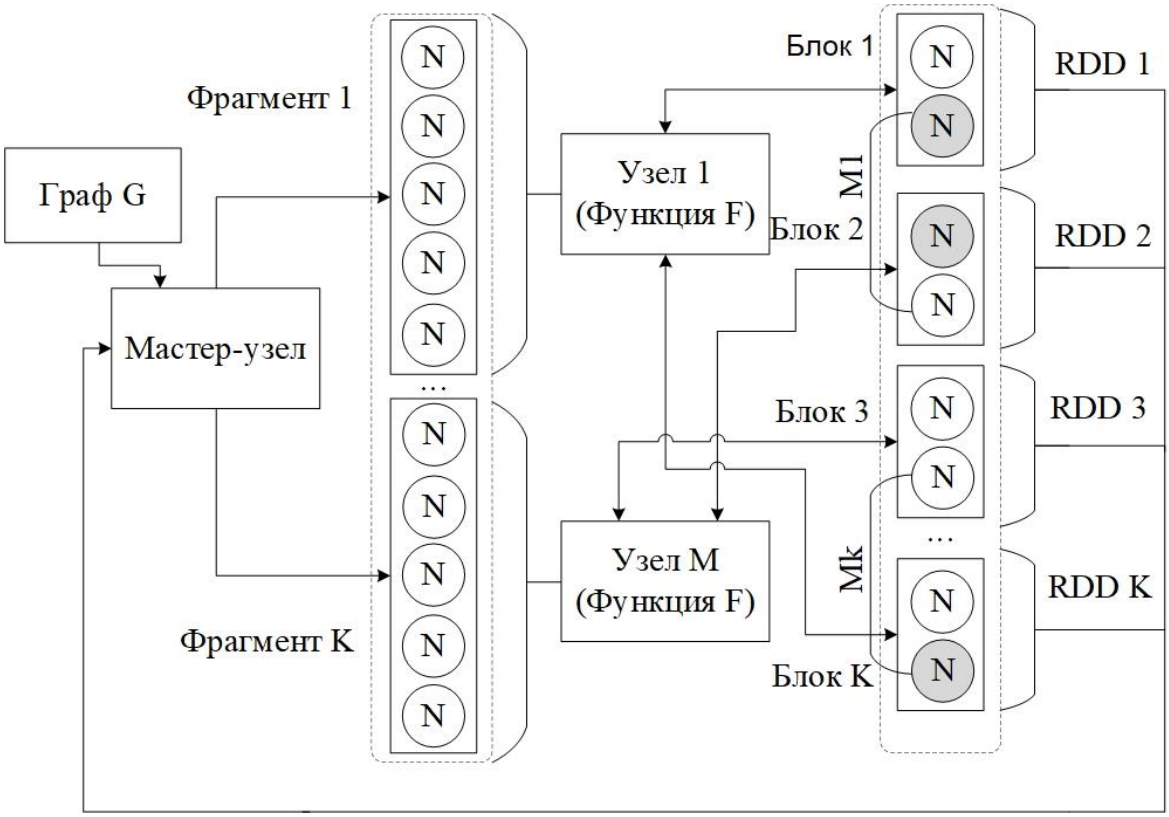

Рисунок 5.3. Параллельный алгоритм обработка графа на основе модели Pregel с использованием Spark RDD

Из рисунка 5.3 видно, что в рамках реализации модели Pregel с использованием RDD, происходит разделение графа на отдельные подграфы для каждого из узлов которого будет применена функция F. При этом узлы, находящиеся в разных RDD, также могут осуществлять пересылку сообщения между собой. Решая задачу построения ДС для ДБС, в процессе использования параллельного алгоритма на основе модели Pregel возникает необходимость построения триангулярного и морального графа, что требует непосредственного изменения топологии исходного графа ДБС. Для исключения конфликтных ситуаций, связанных с удалением или добавление объединенных вершин, необходимо определить порядок выполнения операций с блоками модели, ребрами

и узлами графа ДБС. В таком случае удаление блоков модели происходит прежде всего, так как они могут содержать вершины из различных RDD. Ребра удаляются только после удаления вершин. Следовательно, для обеспечения непротиворечивости после исключения вершин, удаляются все ребра, связанные с данной вершиной. При этом алгоритмом допускается удаление, добавление исходящих из узла ребер, а также удаление вершины в процессе выполнения блока модели, отвечающего за обработку этого узла. Для оптимизации хранения данных в RDD, будем использовать разделение информации относительно узлов и ребер графа ДБС на три типа RDD. В первом типе RDD будет хранится таблица маршрутов, представляющая собой множество пар значений идентификатора узла и множества ребер, связанных с данной вершиной. Во втором типе RDD будет хранится информация относительно вершин, представляемая в виде пар идентификатора узла и данных, связанных с ним. Последний тип RDD предназначен для хранения информации относительно ребер графа. В нем содержится информация относительно структуры (узлы, связанные с данным ребром), а также информация для каждого ребра. Данный тип RDD можно представить в виде кортежа значений: идентификаторов узла-источника и конечного-узла, функции F и идентификатора раздела. Каждый раздел формируется путем разделения таблицы всех доступных узлов на равные части с целью исключения ее полного копирования и использования информации относительно того узла, который в текущий момент будут ассоциирован с данным разделом. В процессе реализации параллельного алгоритма ДС с использованием алгоритма Гиббса наибольший интерес вызывает процедура построения триангулярного графа, дерева кликов, а также расчета распределения $\mathrm{P}(X_{i+1}|X_i)$ для каждой клики $\mathrm{C}_i \in \mathrm{C}$, формируемой в соответствии с алгоритмом построения ДС для ДБС, представленным в параграфе 3.2. Запишем формулы вычисления $\mathrm{P}(X_{i+1}|X_i)$ в соответствии с формулой Гиббса (2.121):

$$\mathrm{P}(X_{i+1}|X_i) = \frac{\prod_{i=1}^{n} \varphi_{C_i}\left(X_{C_{i+1}}\right)}{\sum_{X_i} \prod_{i=1}^{n} \varphi_{C_i}\left(X_{C_{i+1}}\right)}, \tag{5.5}$$

где C_i – клика, $\varphi_{C_i}\left(X_{C_{i+1}}\right)$ – потенциал, соответствующий клике C_i.

Из формулы (5.5) видно, что интерес с точки зрения распараллеливания формирования выборок Гиббса представляет распределение расчета произведения потенциалов для каждой из клик на любом из доступных вычислительных процессоров. Производится параллельное вычисление вероятности каждого перехода $X_i \to X_{i+1}$ марковской цепи. Для множества потенциалов, связанных с кликой C_i формируется функция свертки. Область памяти, ассоциируемая с процессором, будет содержать лишь набор потенциалов φ_{C_i} для данной клики. Такой подход позволяет исключить необходимость дублирования потенциалов для любого из процессов. Актуальные значения для распределения $\mathrm{P}(X_{i+1}|X_i)$ могут быть получены в результате выполнения операции отображения. Если для оптимизации структуры нами было использовано построение дерева сочленений, то вероятности $\mathrm{P}(X_{i+1}|X_i)$ в алгоритме ОМ с использованием выборок Гиббса будут формироваться в соответствии с формулой (5.5). При решении задачи распараллеливания алгоритма ОМ, этапы которого могут быть описаны в виде соответствующих выражений (2.54) и (2.60), интерес представляет процедура распараллеливания обучающей выборки, представляющей наборы параметров θ. В качестве входных данных достаточно указать исходный граф ДБС G, обучающую выборку D, а также задать множество скрытых узлов H и наблюдаемых узлов X. В результате, вычисление логарифма $L(\theta)$ для каждого из параметров (шаг ожидания) будет выполняться на этапе свертки параллельного алгоритма, а вычисление нижней границы логарифма правдоподобия θ' (шаг максимизации) на этапе отображения. В таком случае на этапе свертки получаем искомое распределение по всем скрытым переменным $q(\theta; H)$.

Создание параллельных алгоритмов для решения основных задач вероятностного вывода связано с необходимостью блочной обработки выборок S, формируемых в результате выполнения алгоритмов на основе метода Монте-Карло. В свою очередь задача усложняется с ростом числа временных срезов – общее число выборок будет зависеть от их числа. На практике интерес представляют два основных подхода к распараллеливанию алгоритмов стохастического вывода на основе последовательного метода Монте-Карло, в

частности, алгоритма многочастичного фильтра. Первый способ подразумевает распараллеливание процедуры формирования N выборок.

При разработке параллельных алгоритмов вероятностного вывода необходимо также учитывать рост размеров таблиц условных вероятностей при усложнении топологии ДБС и увеличение общего числа переменных внутри нее. В таком случае в процессе выполнения алгоритма МЧФ возникает необходимость формирования достаточно большого числа выборок. Основным преимуществом алгоритма МЧФ с точки зрения параллелизма является то, что формирование каждой из N выборок производится независимо. Следовательно процедура распараллеливания может быть достаточно просто применена для решения рассматриваемой задачи.

Применительно к модели MapReduce, формирование выборок можно выполнить на этапе операции свертки, при этом результирующие значения переменных ДБС будет храниться в RDD. На этапе отображения будет определяться искомое распределение по всем переменным запроса $P(E_{t+k}|X_{t+k})$, соответствующим состоянию ДБС в момент времени $t+k$, с учетом всех полученных свидетельств. Второй способ подразумевает распараллеливание процедуры формирования весов $W_{t+k}^{i} = P(E_{t+k}|X_{t+k} = S_i)$, а именно распространения выборки S_i по всем переменным наблюдениям при фиксированных значениях переменных свидетельств E_{t+k}.

Данный способ наиболее применим, если ДБС имеет достаточно сложную структуру, а число генерируемых выборок существенно мало. Такой способ распараллеливания может быть использовал в случае применения теоремы РБК и ЛШ в алгоритме МЧФ. Тогда для расчета искомого распределения $P(X_{t+k}|E_{t+k})$ будут использованы лишь выборки, обладающие наибольшими значениями весов. Для исключения выборок будет использоваться теорема РБК, представленная выражением (3.113). Введем следующие обозначения для весов $W' = W(V_{0:t+1}|E_{1:t+1})$ и $W'' = W(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})$. Общая схема параллельной генерации выборок на основе МЧФ с применением теоремы РБК приведена на рисунке 5.4.

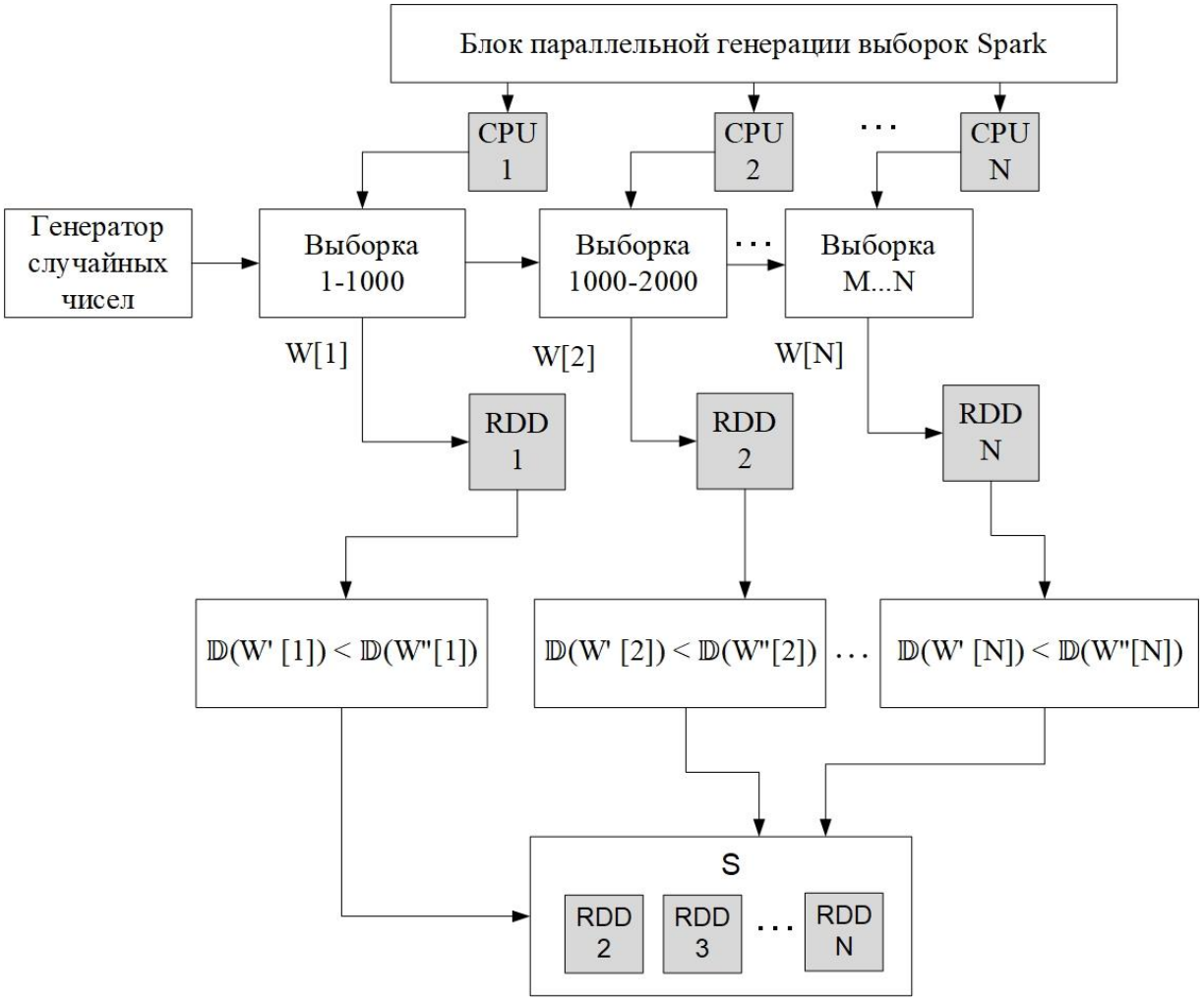

Рисунок 5.4. Параллельная генерация выборок МЧФ-фильтра с использованием Spark RDD и теоремы РБК

Из рисунка 5.4 следует, что генерация выборок производится k – блоками, при этом размер блока подбирается с учетом доступных вычислительных ресурсов системы Spark. В процессе выполнения параллельных фрагментов алгоритма, блоки выборок с минимальными значениями весов $\mathrm{W}[i]$ будут отброшены (блок выборок 1) и в дальнейшем не будут вносить вклад в распределение вероятностей $P(X_{t+k}|E_{t+k})$. Генерация выборок внутри каждого блока выполняется на основе метода Монте-Карло, в качестве генератора случайных чисел используется набор функции операционной системы. Преимущество генерации выборок с использованием РБК в отличии от классического алгоритма МЧФ заключается в том, что мы исключаем шаг повторной генерации выборок, необходимой для достижения согласованности выборок в соответствии с весами $\mathrm{W}[i]$, полученными на первой генерации, так как РБК исключает блоки выборок с низкими весами

Следовательно, в массиве весов $W[i]$ будет находиться только та часть выборок, которая гарантированно согласуется со свидетельствами E_{t+k}. Алгоритм параллельной генерации выборок является основным компонентом оптимизации работы МЧФ, позволяющим повысить точность, оптимизировать процедуру формирования большого числа выборок, а также снизить общий объем итераций алгоритма МЧФ за счет применения РБК. Наряду с применением РБК, интерес представляет применение теоремы ЛШ, сущность которой заключается в поиске ННО на основе неравенства информации (3.144) и построении информационной матрицы $I(S|X,E)$ в соответствие с выражением Фишера (3.142). Тогда матрица Фишера $I(S|X,E)$ будет содержать веса выборок, которые взвешиваются на основе их правдоподобия, в соответствии с поступающими свидетельствами $E_{1:t+k}$. Использование теоремы ЛШ применительно к алгоритму МЧФ направлено на поиск ННО, может быть сведено к решению задачи поиска минимального значения дисперсии $\mathbb{D}\big(W(V_{0:t+1}, U_{0:t+1}|E_{1:t+1})\big)$ и включать в себя использование теоремы РБК в виде критерия поиска ННО.

В таком случае получаем значительное сокращение выборок S, так как учитываться будут лишь та часть из них, которая формируется исходя из требования ННО и обладает минимальным значение дисперсии весов. В процессе выполнения параллельного алгоритма генерируется M/n параллельных выборок, которые затем проходят через алгоритм МЧФ. В таком случае каждый из блоков выполняется параллельно на вычислительной платформе Spark, состоящей из k процессоров. Общее число одновременно обрабатываемых блоков будет равно числу k. Суммарное быстродействие параллельного алгоритма МЧФ с применением теоремы ЛШ будет определяться следующим выражением:

$$B = \frac{\delta + m}{\delta + \frac{m}{K}}, \tag{5.6}$$

где $\delta = \frac{N}{m}$ – общее число циклов параллельного алгоритма, m – размер блока выборки, K – общее число доступных процессоров для обработки выборок.

Параллельный алгоритм вероятностного вывода ДБС на основе МЧФ с применением теорем РБК и ЛШ представлен на рисунке 5.5.

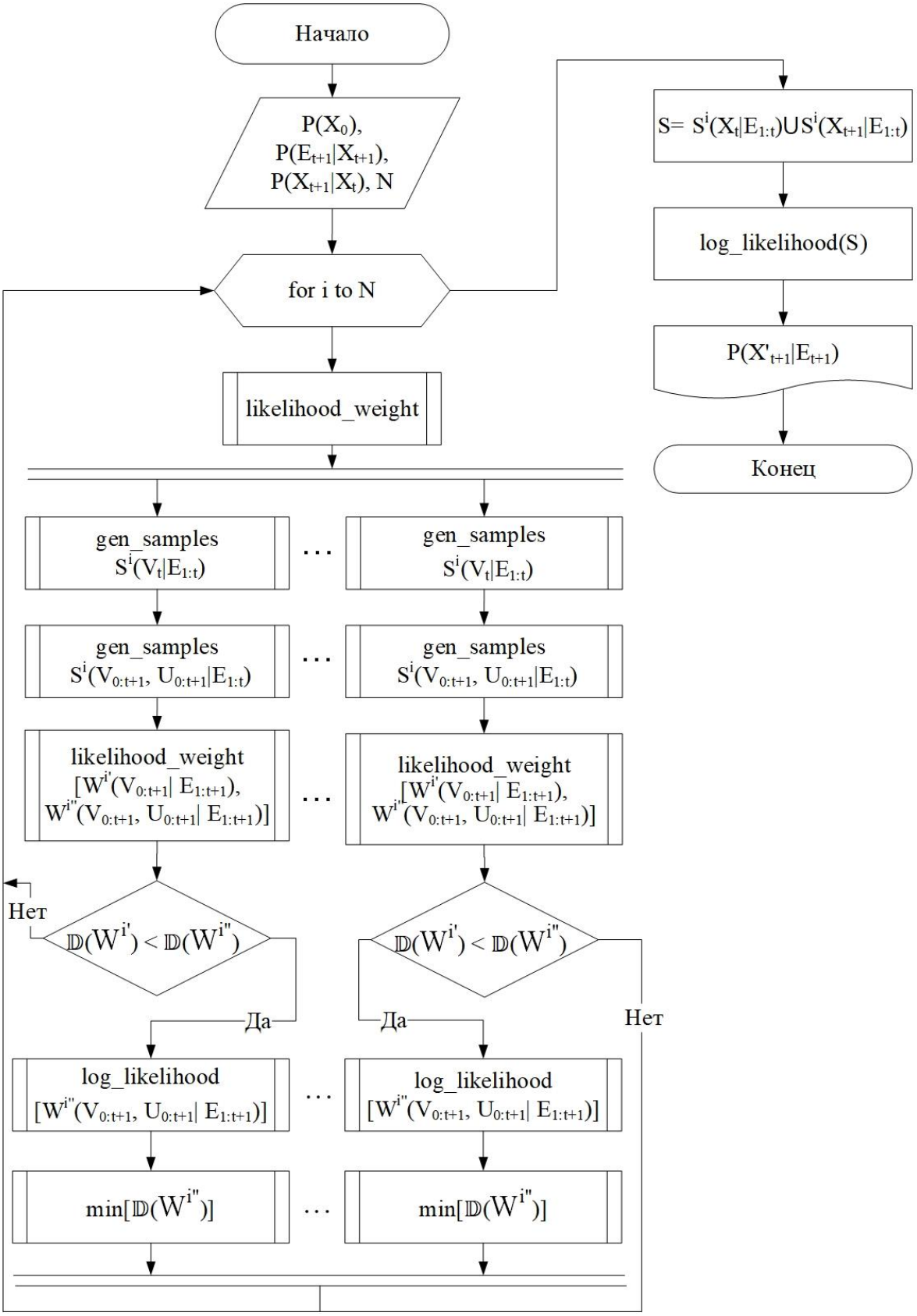

Рисунок 5.5. Параллельный алгоритм МЧФ с применением теорем РБК и ЛШ

Отметим, что в качестве решения задачи распараллеливания и оптимизации МЧФ используется алгоритм взвешивания на основе правдоподобия, так как другие алгоритмы не позволяют применить теорему РБК совместно с теоремой ЛШ. При выполнении алгоритм МЧФ все блоки сгенерированных выборок, а также их веса помещаются в RDD, а указатели на них постоянно доступны внутри алгоритма. Веса W^i, неудовлетворяющие условию РБК, удаляются из RDD. Удаление выборок S^i, соответствующих весам W^i, производится автоматически. Это дает возможность снизить требовательность алгоритма к ресурсам для хранения данных, так как Spark размещает все RDD в оперативной памяти каждого из узлом системы для достижения требуемого уровня производительности. После объединения выборок S^i, имеющих наибольшие веса W^i, по аналогии с алгоритмами обучения, используется метод максимального правдоподобия для получения результирующего распределения по всем переменным наблюдения $P(X_{t+k}|E_{t+k})$. Блочное распределение выборок показывает наилучшие показатели в сочетании с параллельной системой Spark. Это связано с тем, что разработанные алгоритмы исключают дополнительную пересылку данных и создают максимальные условия для использования процессорного времени в распределенной системе. В рамках представленных параллельных алгоритмов, разработаны компоненты отслеживания появления исключений, возникающих в процессе работы параллельных алгоритмов. Блоки данных, которые не обработаны из-за возникновения ошибок времени выполнения системы Spark, проверяются и отправляются повторно. Это достигается за счет создания дополнительных контрольных точек в каждом из параллельных алгоритмов обучения и вероятностного вывода, что гарантирует целостность выборок. Если возникла ошибка времени выполнения, то алгоритм возвращается к наиболее близкой контрольной точке (блоку алгоритма).

Предложенные в работе параллельные алгоритмы обучения и вероятностного вывода направлены на оптимизацию сложных математических процедур, а также повышение ресурсной эффективности за счет применения процедуры разделения отдельных алгоритмических блоков и блоков обучающих

данных на отдельные параллельные вычислительные процессоры. Применительно к динамическим байесовским сетям, предложенные алгоритмы позволяют повысить точность определения структуры сети, определить направленность связей между узлами сети, а также снизить временные затраты на определение начального распределения сети, моделей перехода и восприятия динамической байесовской сети.

Использование алгоритмов рандомизации выборки на основе методов Монте-Карло с применением цепей Маркова позволяет оптимизировать процедуру генерации выборок, а также повысить эффективность процедуры обучения параметров ДБС на основе алгоритма ОМ, за счет моделирования временных связей в виде марковских цепей. Такой подход позволяет эффективно решать задачи обучения ДБС, узлы которой могут иметь дискретное и непрерывное распределение вероятностей. Представленные алгоритмы обладают высокой производительностью и адаптированы для обучения динамических байесовских сетей со сложной иерархической структурой и значительным числом переменных, как в рамках одного временного состояния, так и в рамках достаточно большого числа временных состояний, взятых в хронологическом порядке. Параллельные алгоритмы позволяют использовать ДБС в различных направлениях по реализации процедур тестирования методом фаззинга. Рост числа разнородных приложений, увеличение числа модулей и взаимодействующих компонентов несомненно ведет к усложнению ДБС в рамках решения задач фаззинга. В таком случае использование процедур распараллеливания позволяет создавать предобученные модели, позволяющие оптимально производить поиск и выявления ошибок внутри веб-приложений, а также своевременно сигнализировать о возможных аномалиях, которые потенциально могут привести к возникновению ошибок в будущем. Такие модели могут быть применены в системах обнаружение вторжений, тестирования программ различного назначения, межсетевых экранах веб-приложений (WAF), а также в любой системе, реализующей механизмы фильтрации и блокирования сетевой активности внутри информационной системы.

5.2. Построение структуры стенда для проведения вычислительного эксперимента

Рассматривая возможность применения моделей ДБС в рамках проведения фаззинг-тестирования, немаловажно уделить особое внимание различным математическим и программным аспектам, необходимым для эффективной организации комплексного анализа ошибок веб-приложений с применением методов фаззинга. Наряду с компонентами фаззинга, направленными на поиск и локализацию программных ошибок, определяемых согласно классификации OWASP, необходимо разработать элементы, отвечающие за взаимодействие между модулями фаззинга и обучения ДБС, а также целевыми веб-приложениями, подвергаемыми тестированию. Решение задач обучения и вероятностного вывода в процессе реализации различных процедур тестирования веб-приложений на основе методов фаззинга формализуется в его представление в виде ДБС, развертываемой на заданном временном интервале. Каждое рассматриваемое временное состояние модели характеризует процесс тестирования определенной группы приложений в заданный момент времени. Под группой приложений подразумевается совокупность приложений, имеющих общие признаки. В данном случае – схожий набор программных модулей, библиотек и компонентов. Среди компонентов можно выделить системы управления сайтами (СУС-системы), объединяющие в себе средства обработки запросов, визуализации, шаблонизации, работы с базами данных, сетевыми сервисами и управления контентом. Среди систем управления сайтами особую популярность получили СУС-системы: WordPress, Joomla, 1С Битрикс, OpenCart, Drupal. Данные системы обладают расширенными возможностями, позволяют создавать информационные порталы, интернет-магазины и другие веб-приложения за короткий промежуток времени, без непосредственной разработки программных компонентов. Приложения на основе 1С Битрикс обладают широкими возможностями для интеграции с другими сервисами платформы 1С, в частности, 1С предприятие и бухгалтерия. Наряду с

СУС-системами, при разработке приложений используются фреймворки, позволяющие упростить написание приложений. Наиболее популярными из которых являются ASP.NET, Spring MVC, Laravel, Django, Ruby on Rails, Angular JS. Компоненты данных фреймворков используются для создания модулей доступа к данным, авторизации и аутентификации пользователей, а также пользовательского интерфейса, сетевых служб и сервисов.

Структура среды тестирования веб приложений представляет собой совокупность: виртуальной среды, состоящей из компонентов отвечающих за работу тестовых приложений, элементов фаззинга и моделей обучения и вероятностного вывода на основе ДБС. Виртуальная среда создана при помощи системы контейнеризации Docker, развернутой на операционной системе Debian. Данная среда состоит из отдельных виртуальных контейнеров веб-серверов и серверов-приложений (Apache Http Server, Nginx, Tomcat) с программными модулями языков: C#, PHP, Python, Java и Ruby; серверов баз-данных Oracle 21c XE, PostgreSQL v.16, MySQL Community v.8, SQL Server 2022 Express; межсетевым экраном WAF ModSecurity; сервером Selenium Grid, с предустановленными различными версиями веб-браузерами Google Chrome, Firefox, Microsoft Edge и Opera. Платформа Selenium Grid используется для тестирования программных ошибок, в частности MCC, возникающих в результате выполнения динамического кода JavaScript, а также исходя из специфики обработки html-страниц определенным браузером.

Для выявления и анализа ошибок методом фаззинга используется набор веб-приложений, содержащих ошибки различного класса и уровня критичности, разрабатываемых в рамках проекта OWASP, а также СУС-приложения для некоторых наиболее популярных систем управления сайтами. Данные приложения можно разделить исходя из веб-технологий, используемых в процессе создания данных приложений. Это необходимо для того, чтобы осуществлять настройку тестов для анализа особенностей обработки параметров каждой из веб-технологий. Для моделирования фаззинга во времени с использование ДБС предполагается изменение поведения веб-приложений (усиление политик безопасности), а также

включение и отключение межсетевых экранов безопасности (WAF). Такой подход позволяет имитировать выпуск обновлений (патчей) приложений, закрывающих определенную группу ошибок или набор разнородных ошибок. Полный список веб-приложений, используемых в процессе проведения фаззинг-тестирования, приведен в таблице 5.1

Таблица 5.1

Список приложений, используемых в рамках проведения тестирования методом фаззинг с использованием ДБС

Веб-технология	Приложение	Версия
ASP.NET	WebGoat.NET	1.5
PHP	1С Битрикс	16.5,16.5.4
	Joomla	1.5.15, 2.5
	WordPress (myGallery 1.2.1, Spreadsheet WordPress 0.9)	4.0,4.1
	Mutillidae	2
	OrangeHRM	2.6
	Bricks	1.6
	WebCalendar	1.3
	Damn Vulnerable Web Application	2.0.1
	OWASP Vicnum	1.7
	Magical Code Injection Rainbow	1.1
	Ghost	1.2
	gtd-php	0.7
	WackoPicko	0.9
	Peruggia	1.5
	Cyclone Transfers	1.0
	GetBoo	1.04
	gtd-php	0.7
	RailsGoat	5.0
	Gallery	2.2
	TikiWiki	1.9.5
	WIVET	3
Ruby on Rails	ESAPI	1.0
	Hackxor	12.7
Java	BodgeIt	1.2.6
	OWASP CSRFGuard	3.1
	OWASP Wavsep	1.5
	Mandiant Struts Forms	0.9
	Yazd	2.0
	ZAP Wave	0.2
	WebGoat	8.1.0

В рамках проведения экспериментальной части исследования предполагается, что каждый тип веб-приложений в таблице 5.1 будет формировать группу однородных веб-приложений. Такой подход дает возможность выстроить процедуру

тестирования методом черного и серого ящика и позволяет произвести настройку инструментов фаззинга под тестирование определенного набора программных компонентов, входящих с состав данных приложений, а также определить механизмы межмодульного и межпрограммного взаимодействия с другими компонентами и системами. В процессе анализа фаззинг-тестирования веб-приложений и специфики обработки входных параметров, используемой веб-серверами, можно реализовать ряд подходов, направленных на обход некоторых фильтров СУС и приложений, использующих веб-фреймворки. Одним из таких подходов является механизм смешивание входных параметров (СВП). Для наглядности приведем таблицу, отражающую особенности обработки http-запросов.

Таблица 5.2

Особенности обработки параметров веб-приложений и http-серверов на основе подхода СВП

Веб технология/ HTTP сервер	Результат преобразования	Результат
JSP, Servlet/Oracle Application Server 11g	Выбирается первое вхождения параметра	$par_1 = val_1$
PHP/Apache	Выбирается последнее вхождения параметра	$par_1 = val_2$
IceWarp	Выбирается последнее вхождения параметра	$par_1 = val_1$
JSP, Servlet/Jetty	Выбирается первое вхождения параметра	$par_1 = val_1$
ASP.NET/IIS	Объединение параметров символом «,»	$par_1 = val_1, val_2$
mod_perl, libapreq2/Apache	Выбирается последнее вхождения параметра	$par_1 = val_1$
PHP/Zeus	Выбирается последнее вхождения параметра	$par_1 = val_2$
IBM Lotus Domino	Выбирается последнее вхождения параметра	$par_1 = val_2$
JSP, Servlet/Apache Tomcat	Выбирается первое вхождения параметра	$par_1 = val_1$
Perl CGI/Apache	Выбирается первое вхождения параметра	$par_1 = val_1$
IBM HTTP Server	Выбирается первое вхождения параметра	$par_1 = val_1$
Python/Zop	Массив параметров	$['val_1', 'val_2']$
ASP/IIS	Объединение параметров символ «,»	$par_1 = val_1, val_2$

Рассматривая результаты обработки параметров на основе СВП, приведенные в таблице 5.2. Видно, что различные веб-приложения и http-сервера имеют свой набор правил обработки параметров. Данная особенность позволяет использовать различные механизмы СВП для обхода логики приложения и алгоритмов фильтрации, применяемых на различных уровнях функционирования веб-приложений и систем управления сайтами. Наряду с применением СВП через непосредственную отправку смешанных параметров на веб-сервер, СВП могут быть использованы на стороне клиента, оптимизированы для внедрения JavaScript-кода и применены для анализа ошибок межсайтового скриптинга. Исходя из классификации МСС, представленной в главе 1, в СВП для совместного использования с МСС можно выделить следующие функциональные компоненты:

1. DOM СВП. Позволяет использовать DOM-модель документа и механизмы языка JavaScript для создания и внедрения СВП.

2. Отраженные СВП. Реализуют механизм отправки СВП через различные сервисы и ресурсы, в частности, через сторонние сервисы и сообщения электронной почты;

3. Хранимые СВП. Дают возможность хранения кода СВП в различных системах хранения как на целевом сервере, так и внутри веб-браузера пользователя (база данных, хранилище сеансов, кэш-хранилище).

Используя анализ ошибок на основе СВП, можно определить аномальное поведения веб-приложений в процессе обработки групп параметров $Par = (par_1, par_2, \dots, par_N)$, а также параметров, принимающих на входе массив значений $Par_1 = (par_1[1] = val_1, par_1[2] = val_2, .., par_1[N] = val_N)$.

Формирование результирующих значений для Par_1 будет выполняться в соответствии с таблицей 5.2. Такой подход может быть использован для разделения тестовых генераций и обнаружения определенного типа ошибок. По результатам обработки каждой пары $par_i[1] = val_i$ и $par_i[1] = val_j$ получим объединение значений $par_i[1] = val_i, val_j$. Использование данного подхода позволяет создать разделенные тестовые данные, объединяемые в процессе обработки входных параметров конкретным веб-сервером или сервером-приложений. Учитывая

особенности тестирования методом фаззинга, приведем обобщенную модель процесса анализа программных ошибок веб-приложений на основе фаззинга

Рисунок 5.6. Модель тестирования методом фаззинга

Из рисунка 5.6 видно, что механизмы фаззинга включают два основных этапа: сканирование и эксплуатацию. На этапе сканирования производится определение входных параметров приложения, методов передачи данных с использование протокола http, а также происходит генерация и отправка тестовых выборок в целевое приложение с целью определения факта наличия программной ошибки внутри веб-приложения. На этапе эксплуатации используем ошибки, найденные на этапе сканирования с целью выполнения кода, позволяющего эксплуатировать данную ошибку. Это позволяет оценить, насколько критична ошибка и какие функциональные возможности по раскрытию данных она предоставляет. Основное направление применения сканирования и эксплуатации – это определение насколько могут быть нарушены показатели целостности,

конфиденциальности и доступности информации, обрабатываемой веб-приложением. Статистическая информация будет содержать только выборки, включающие в себя параметры модулей сканирования и эксплуатации, которые наиболее эффективно показали себя в процессе фаззинга веб-приложений. После получения выборок происходит обучение структуры и параметров ДБС $P(\mathrm{X_t})$ в соответствии с каждой группой программных ошибок. Далее происходит решение задач вероятностного вывода: фильтрация $P(X_{t+1}|E_{1:t+1})$, прогнозирования $P(X_{t+k+1}|E_{1:t})$ и сглаживания $P(X_{t+k}|E_{1:t})$. По результатам вероятностного вывода с использованием МЧФ формируется N выборок для каждой из ДБС в соответствии со своим классом ошибок. Полученные выборки передаются в блок фаззинга и используются для дальнейшего тестирования различных веб-приложений. Такой подход позволяет сделать процесс тестирования более осмысленным и реализовать стратегию тестирования методом серого ящика. В процессе фаззинга происходит постоянная коррекция тестовых данных, исходя из выборок, полученных для каждой из ДБС по результатам вероятностного вывода. В следствие чего, происходит настройка среды тестирования для поиска и локализации определенных групп программных ошибок веб-приложений. Далее производится адаптация системы к обнаружения аномального поведении веб-приложений в процессе проведения тестирования. В рамках фаззинга веб-приложений согласно рисунку 5.6 возникает необходимость детальной проработки каждого из компонентов фаззинга в соответствии с группами ошибок, рассмотрения различных способов и механизмов распараллеливания вычислений, связанных с обучением и вероятностным выводом ДБС, определения структуры и топологии построения информационной системы, элементами которой могут выступать веб-приложения и веб-сервисы, а также системы хранения и обработки информации на основе баз данных. Такой подход позволяет реализовать процедуру комплексного тестирования, направленную на решение основных задач фаззинга. Обобщенная схема процесса тестирования веб-приложений методом фаззинга с использованием ДБС приведена на рисунке 5.7

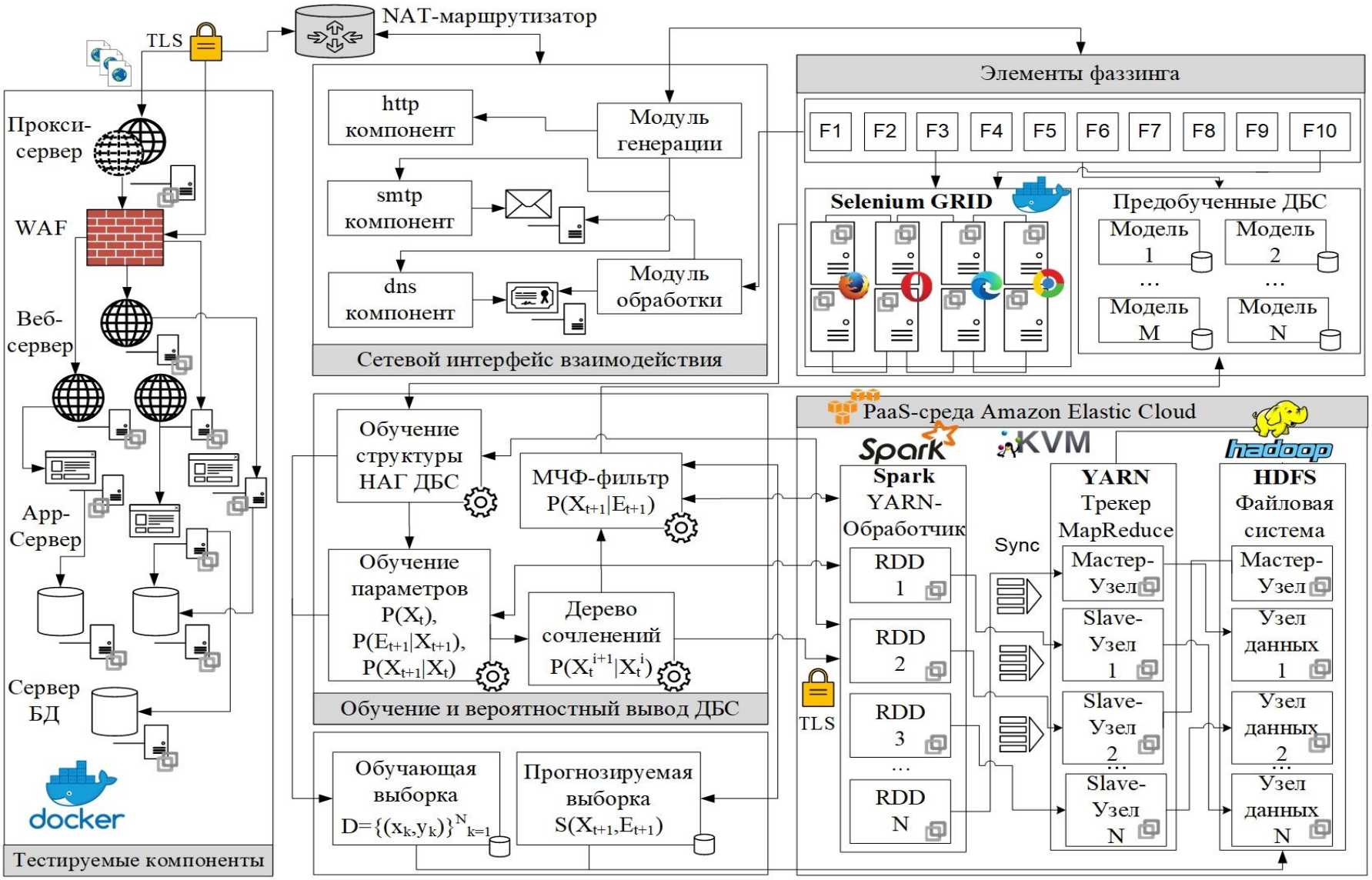

Рисунок 5.7. Структура аппаратного и программного обеспечения процесса тестирования веб-приложений методом фаззинга

В соответствии с рисунком 5.7, для оптимизации задач обучения ДБС в рамках исследования используются вычислительные мощности и инфраструктура сервиса Amazon Elastic Cloud (EC2). Данный сервис обладает достаточно гибкой архитектурой и содержит набор образов для автоматизации развертывания распределенных вычислительных систем на основе Hadoop и Spark. EC2 используется как PaaS-сервис, следовательно для выполнения задач обучения и вероятностного вывода, тестовые генерации, прошедшие верификацию в процессе выполнения фаззинга методом черного ящика попадают в обучающее множество ДБС, ассоциируемое с процессом обнаружения определенной группы программных ошибок и сохраняются на распределенную файловую систему HDFS, развернутую на платформе EC2. При этом каждая из ДБС, моделируемая для любой OWASP-группы ошибок веб-приложений, ассоциируется со своей обучающей выборкой, хранящейся в виде отдельного файла на HDFS. Программа, решающая задачи обучения и вероятностного вывода, подгружается в параллельную систему Spark и после заполнение выборок с использованием отдельных модулей фаззинга $F = (F_1, F_2, \dots, F_{10})$ производит последовательный запуск процедур обучения структуры и параметров, а также вероятностного вывода. По результатам выполнения данных процедур формируемся множество выборок $S = (S_1, S_2, \dots, S_N)$, вычисляется распределение вероятностей $P(X_{t+k}, E_{t+k})$. Применение такого подхода позволяет формировать набор предобученных моделей ДБС, которые могут быть использованы не только в фаззинге, но и для построения WAF, а также любой системы, направленной на детектирование как определенной группы ошибок, так и комплексного выявления всех групп ошибок [187,193]. Наборы обученных моделей ДБС с соответствующими таблицам условных вероятностей могут хранится как в виде файлов, так и в форме записей баз данных. В рамках работы разработан модуль загрузки и использования моделей ДБС в рамках межсетевого экрана ModSecurity. Данный модуль является достаточно эффективным, однако для достижения требуемого уровня гибкости возникает необходимость формирования достаточно большого числа правил, описывающих специфику обнаружения ошибок. Правила

задаются в виде регулярных выражений для соответствующих блоков и заголовков HTTP-запроса. ModSecurity может использовать обученные модели ДБС в виде самостоятельного элемента, исключая необходимость использования процессов фаззинга. В данном случае считаем, что предобученные модели содержат полный набор правил, необходимых для своевременного детектирования и блокирования ошибок веб-приложений. Сформированные ДБС модели категорируются в соответствии с группами ошибок OWASP, записываются в отдельные блоки конфигурации ModSecurity и могут быть использованы после его перезагрузки. Такой подход позволяет упростить процедуру обновления данных ModSecurity – параметры моделей могут быть обновлены для конкретной группы ошибок, без необходимости перезаписи уже имеющихся данных, предназначенных для обнаружения ранее выявленных ошибок устойчивости функционировании.

Наряду с анализом классических SQL-инъекций, в рамках структуры фаззинга, приведенной на рисунке 5.7, реализована возможность анализа SQL-инъекций, использующих альтернативные каналы взаимодействия на ряду с протоколом HTTP. В данном случае тестовые генерации SQL запросов, формируемых модулем, будут включать расширенный набор системных команд для СУБД Oracle (utl_inaddr.get_host_address, utl_inaddr.get_host_address, utl_smtp.open_connection, utl_mail.send), Microsoft SQL Server (sp_OACreate, msdb.dbo.sp_send_dbmail), PostgreSQL (copy from). СУБД будет отправлять DNS или SMTP запрос на соответствующие DNS или SMTP компоненты фаззинга, имеющие встроенные обработчики событий. При каждом поступлении блока данных осуществляется непосредственное взаимодействие с модулем фаззинга и обработчиком запросов. В данном случае задача обработчика выделить фрагмент данных, полученный в результате выполнения SQL-запроса с использованием DNS или SMTP запроса. Отметим, что если объем данных в таблице является достаточно большим, то используются лимитированные выборки с поэтапным циклическим смещением вдоль всего массива данных. Типовая схема проведения тестирования SQL-инъекции [174,168] на примере команд СУБД ORACLE представлена на рисунке 5.8:

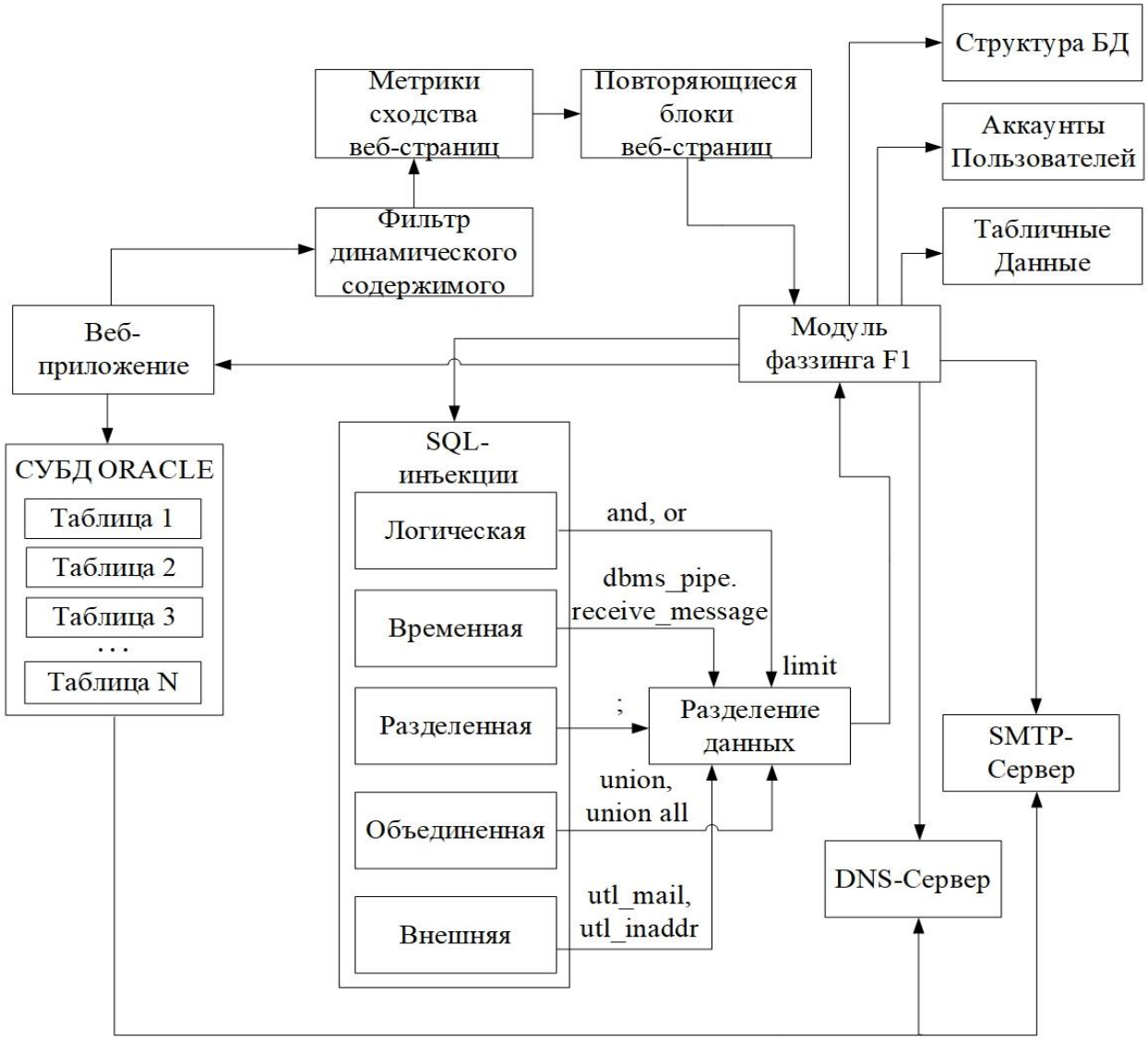

Рисунок 5.8. Модель фаззинга SQL-инъекции на примере СУБД ORACLE

Рисунок 5.8 показывает, что для получения данных и метаданных, хранящихся внутри СУБД, целесообразно использовать фрагментированные запросы для ограничения единовременного получения блоков данных с использованием SQL-инъекции. В частности, использование внешней DNS SQL инъекции ограничивается размером блока данных поля LN, входящего в структуру DNS-имени, в 63 символа согласно документу RFC 1035. В связи с чем, в дополнение к процедуре построчного ограничения записей, целесообразно использовать оператор извлечения подстроки «substr». Максимальный объем подстроки будет равен длине поля LN DNS-имени. Такая процедура особенно важна, если производится извлечения текстовых и бинарных полей большой размерности. В качестве временной инъекции можно использовать функцию

dbms_pipe.receive_message(cmd, sleep_time), при этом наличие задержки будет подтверждением присутствия инъекции. Рассматривая особенность применения логической инъекции, возникает ряд проблем, связанных с тем, что большинство современных приложений используют динамическую генерацию содержимого, вследствие чего, факт детектирования таких ошибок может быть затруднен. В процессе анализа логической SQL-инъекции, в общем случае, происходит контроль изменения содержимого веб-страницы. Если веб-приложение имеет динамическое содержимое, то процедура определения инъекции усложняется. К динамическому содержимому относятся изменяющиеся блоки скриптов, таблиц стилей и html-тегов. Для решения данной проблемы в исследовании предлагается использование различных метрик определения схожести текста после выполнения нескольких запросов к веб-странице. Одной из таких метрик является $tfidf$. Данная метрика устанавливает связь между частотами появления определенных слов (термов), расположенных внутри фиксированного документа (текстового фрагмента). Документ d представляет собой множество слов $W = (w_1, w_2, \dots, w_n)$. Метрика $tfidf$ состоит из двух параметров: частоты термов $tf(w, d)$ и инверсного значения частоты документа $idf(w, D)$. Пусть $N(w, d)$ – частота появления слова t в документе d, $N_{w,d}$ – общее число слов в документе d, N – общее число рассматриваемых документов, $N(w)$ – число документов, в которых присутствует слово w. Тогда результирующее значением метрики $tfidf$ определяется в виде произведения двух множителей – $tf(w, d)$ и $idf(w, D)$ [75]:

$$tf(w, d) = \frac{N(w, d)}{N_{w,d}},$$
$$idf(w, D) = log\frac{N}{N(w)}, \quad (5.7)$$
$$tfidf = tf(w, d)idf(w, D)$$

где $D = (d_1, d_2, \dots, d_N)$ – массив документов размерностью N.

Рассмотрим применение метрики $tfidf$ к решению задачи определения схожести двух строк A и B, представляющих собой исходный тексты веб-страницы.

Для этого разобьем каждую из строк на множество слов W_A и W_B. Тогда значение метрики $tfidf$ можно записать в следующей форме [35]:

$$tfidf(A,B) = \frac{\sum_{w \in A \cap B} tfidf(w,A) tfidf(w,B),}{\sqrt{\sum_{t' \in W_A} tfidf(t',A)^2} \sqrt{\sum_{t' \in W_B} tfidf(t',B)^2}} \quad (5.8)$$

Из формулы (5.8), определяющей метрику видно, что данные простейшие метрики эффективны лишь при минимальных размерах наборов текстов A и B, однако в условиях повышения объема анализируемых данных возникает сложность перерасчета частот каждого из сравниваемых наборов документов. Для решения данных проблем можно использовать алгоритмы на основе нечеткого поиска. В таком случае процедуру расчета частот можно заменить на расчет метрик, определяющих расстояния между отдельными словами двух веб-страниц. В качестве таких метрик наибольший практический интерес представляют метрики Левенштейна, $soft_{tfidf}$ и Дамерау-Левенштейна (ДЛ). Под метрикой (расстоянием) Левенштейна понимаем минимальный набор действий по преобразованию строки A в строку B. Отличительные особенностью метрик ДЛ и Левенштейна является: добавление операции транспозиции (перестановок двух смежных символов строки), наряду с описанным в работах Левенштейна [153] операциями добавления, удаления и замещения символов.

В работах Коена и Равикумара [21] представлена оптимизированная метрика $soft_{tfidf}$, включающая в себя максимально допустимое расстояние и характеризующая появление слова w в строках A и B. Пусть существует два слова $a \in A$ и $b \in B$, тогда можно задать функцию их сходства $sim(a,b)$. Рассмотрим множество $close(\theta, A, B)$, состоящее их тех элементов $a \in A$, для которых значение функции $sim(a,b)$ больше порогового значения – $sim(a,b) > \theta$. Обозначим $N(a,B) = \max_{b \in B} sim(a,b)$, тогда результирующую метрику $soft_{tfidf(A,B)}$ можно определить в следующем виде:

$$soft_{tfidf(A,B)} = \sum_{a \in close(\theta,A,B)} tfidf(a,A) \times tfidf(b,B) \times N(a,B), \quad (5.9)$$

Расчет $sim(a,b)$ в формуле (5.9) можно произвести на основе вычисления расстояния Дамерау-Левенштейна для двух слов a и b.

Рассмотрим алгоритм вычисления метрики ДЛ двух строк A и B. Для этого необходимо определить расстояние D_{ij} между символом i, входящим в состав строки A, и символом j, входящим в строку B. Далее происходит вычисление минимального расстояния между символами i и j с применением индикаторной функции $S(A_i, B_j)$, принимающей значение нуля в случае равенства $A_i = B_j$ и значение единицы во всех остальных случаях. Выражение для расчета расстояния $D_{i,j}$запишем в виде следующей системы [23]:

$$D_{i,j} = min\begin{cases} 0, i = j = 0 \\ D_{i-1,j} + 1, i > 0 \\ D_{i,j-1} + 1, j > 0 \\ D_{i-1,j-1} + S, i, j > 0 \\ D_{i-2,j-2} + 1,\ A_i = B_{j-1}, A_{i-1} = B_j, i, j > 0 \end{cases}, \quad (5.10)$$

$$S = \begin{cases} 1, \text{если } A_i = B_j \\ 0, \text{иначе} \end{cases}$$

где $D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}, D_{i-2,j-2}$ – действие, характеризующие удаление, вставку, замену и транспозицию символов между строками A и B.

Из формулы (5.10) расчета расстояния ДЛ следует, что $D_{i,j}$ представляет собой двумерную матрицу каждый элемент которой есть минимальное значение среди показателей $D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}, D_{i-2,j-2}$. Исключая показатель транспозиции $D_{i-2,j-2}$, каждый элемент $D_{i,j}$ будет являться расстоянием Левенштейна. Для расчета расстояния согласно формуле (5.10) может быть использован алгоритм Вагнера-Фишера (ВФ) [89]. Сущность подхода, предложенного в алгоритме ВФ, заключается в построчном обходе матрицы $D_{i,j}$ и вычислении расстояния ДЛ. Временная сложность алгоритма ВФ будет пропорциональна размеру матрицы $D_{i,j}$. Для оптимизации расчетов матрицы расстояний $D_{i,j}$ могут быть использованы различные подходы. Первый подход заключается в параллельном вычислении матрицы $D_{i,j}$. Производится разбиение матрицы на отдельные строки и параллельное вычисление

$\min\left(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}, D_{i-2,j-2}\right)$, с последующим формированием результирующей матрицы расстояний ДЛ.

Альтернативный подход заключается в использовании метода отсечений Укконена-Хиро (МОУХ) для оптимизации расчета матриц $D_{i,j}$ расстояний Левенштейна и ДЛ. Укконен показал, что разность между соседними элементами матрицы $D_{i,j}$, располагаемыми в одной строке или столбце может принимать значения $\{-1,0,+1\}$, в то время как между диагональными элементами она будет принимать значения 0 или 1 [86]. В тоже время Хиро доказал, что для расстояния ДЛ разность между диагональными элементами матрицы $D_{i,j}$ может быть равна 2 [33]. Тогда для расчета расстояния Дамерау-Левенштейна с использованием алгоритма МОУХ требуется ввести некоторое пороговое значений K. В таком случае, если расстояние между строками $d(A,B) \leq K$, то A и B совпадают, в противном случае нет. Для диагональных элементов справедливо правило отсечения, если они представляют собой неубывающую последовательность: $D_{i+1,j+1} - D_{i,\text{j}} > 0$. Следовательно, если $D_{i,j} > K$ и диагональные элементы $D_{i+k,j+k}, k \geq 0$ будут превышать порог K, их вклад в результирующее расстояние можно не учитывать.

Пусть задана переменная U, соответствующая порядковому индексу отсечения для столбца j. Элементы, имеющие индексы заведомо большие U исключаются. Для первого столбца матрицы $D_{i,j}$ значение U будет равно пороговому значению. Тогда повторяя цикл по всем столбцам $j = 1 \ldots U$ на основе алгоритма ВФ, переменная U увеличивается на 1. При условии, что для элемента, стоящего в новом столбце $D_{i,U} > K$, значение U уменьшается до тех пор, пока $D_{i,U} \leq K$. Если все элементы больше порогового значения $D_{i,U} > K$ в текущем столбце U, то выполнение алгоритма останавливается, а значения элементов в остальных столбцах исключаются и формирование матрицы расстояний D_{ij} на основе метода отсечений Укконена-Хиро завершено [87]. На рисунке 5.9 представлены матрицы расстояний и соответствующее расстояние ДЛ для двух слов A – «injection» и B – «inspection», вычисляемые в соответствии с алгоритмами

Вагнера-Фишера и Укконена-Хиро. Для алгоритма Укконена-Хиро элементы, отмеченные как «*», являются отсеченными согласно правилу $D_{i,U} > K$, так как не вносят вклад в вычисление расстояния ДЛ. Закрашенной областью отмечено расстояние ДЛ, вычисляемое в процессе формирования матрицы D_{ij}.

		i	**n**	**s**	**p**	**e**	**c**	**t**	**i**	**o**	**n**
	0	1	2	3	4	5	6	7	8	9	10
i	1	0	1	2	3	4	5	6	7	8	9
n	2	1	0	1	2	3	4	5	6	7	8
j	3	2	1	1	2	3	4	5	6	7	8
e	4	3	2	2	2	2	3	4	5	6	7
c	5	4	3	3	3	3	2	3	4	5	6
t	6	5	4	4	4	4	3	2	3	4	5
i	7	6	5	5	5	5	4	3	2	3	4
o	8	7	6	6	6	6	5	4	3	2	3
n	9	8	7	7	7	7	6	5	4	3	2

а)

		i	**n**	**s**	**p**	**e**	**c**	**t**	**i**	**o**	**n**
	0	1	2	3	4	*	*	*	*	*	*
i	1	0	1	2	3	*	*	*	*	*	9
n	2	1	0	1	2	*	*	*	*	*	*
j	3	2	1	1	2	*	*	*	*	*	*
e	4	3	2	2	2	*	*	*	*	*	*
c	*	*	*	*	*	*	*	*	*	*	*
t	*	*	*	*	*	*	*	*	*	*	*
i	*	*	*	*	*	*	*	*	*	*	*
o	*	*	*	*	*	*	*	*	*	*	*
n	*	*	*	*	*	*	*	*	*	*	*

б)

Рисунок 5.9. Матрицы расстояний, полученные с помощью методов ВФ (а) и МОУХ (б) при заданном пороге $K = 3$.

Из рисунка 5.9 видно, что расстояние ДЛ будет идентично для двух алгоритмов и равно 2. В свою очередь, метод МОУХ в значительной степени позволяет оптимизировать вычисление матрицы расстояний за счет исключения из матрицы элементов, значение расстояния для которых больше установленного порогового значения K. Следовательно, для расчета расстояния ДЛ с использованием МОУХ достаточно сформировать квадратную матрицу, имеющую размерность $n \times n$ элементов. Наряду с решением задач расчета матрицы расстояний $D_{i,j}$ методом МОУХ, можно использовать алгоритмы распараллеливания данной матрицы на основе алгоритма ВФ. В этом случае происходит формирование n – параллельных заданий для каждого символа строки A. Тогда число символов, для которых будет параллельно формироваться элементы $D_{i,j}$ будет кратно числу доступных процессоров в вычислительной системе. Одним из недостатков алгоритма МОУХ является необходимость предварительного

задания порогового значения K, позволяющего отсечь лишние элементы матрицы $D_{i,j}$. Выбор данного параметра необходимо осуществлять эмпирически, так как в случае малых значений K, алгоритм может дать неправильные значения расстояний ДЛ, так как мы заведомо исключаем элементы, которые могли бы быть изменены. При больших значениях $K \to \infty$, метод МОУХ сводится к классическому методу расчета ВФ, который требует формирования всех элементов $D_{i,j}$ для получения расстояний ДЛ.

В рамках исследования, для расчета расстояний ДЛ для двух веб-страниц, содержащих динамическое содержимое, будем использовать метод МОУХ в сочетании с распараллеливанием операций вычисления как действующих, так и отсеченных элементов матрицы $D_{i,j}$, предварительно совершив подбор порогового значения K. Классический подход ВФ ограничен в силу того, что веб-страниц являются неоднородными по структуре и включают в себя html-теги наряду с данными, что приводит к увеличению общего числа слов на странице. В качестве примера приведем оценки эффективности алгоритма МОУХ с пороговым значением $K = 3$ по сравнению со стандартным алгоритмом ВФ в зависимости от роста объема данных, передаваемых по результатам выполнения http-запросов в СУС типа WordPress 4.0, 4.1 и 1С Битрикс 16.5,16.5.4, обрабатываемых на процессоре Intel Xeon 3.7 ГГц с 4 ядрами, объем памяти 32 ГБ

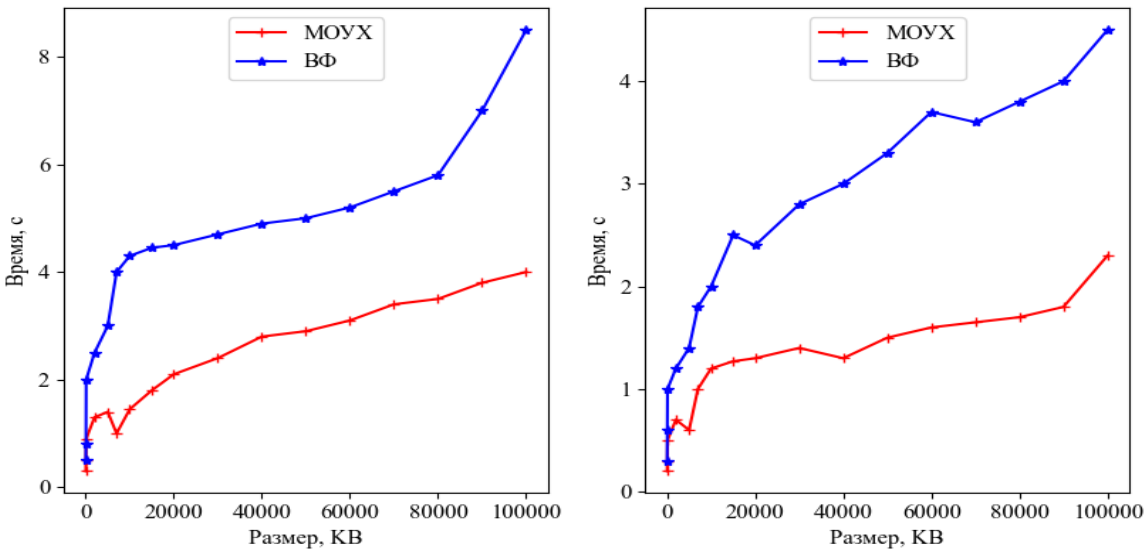

Рисунок 5.10. Оценка эффективности получения расстояния ДЛ с помощью методов ВФ и МОУХ в однопоточном и параллельном режиме

Из рисунка 5.10 видно, что в однопоточном режиме наблюдается снижение эффективности алгоритма МОУХ при небольших объемах данных. Это связано с необходимостью эмпирического определения значения порога K. В многопоточном режиме подбор данного порога осуществляется значительно быстрее и алгоритм показывает хорошие показатели при работе в параллельном режиме. На приведенном рисунке, максимальный объем обрабатываемых данных взят размером 100 МБ, так как веб-страницы большего объема не могут быть загружены браузером, а следовательно веб-приложения ограничивают размер единоразово загружаемой веб-страницы для повышения их интерактивности – это приводит к снижению ресурсных затрат не только веб-браузера, но и уменьшает сетевой трафик веб-приложения в целом.

В процессе тестирования MCC-ошибок нет необходимости отслеживать динамическое содержимое, так как JavaScript-событие обрабатывается веб-браузером и ассоциируется с определенным элементов DOM-модели документа или модального диалогового окна. Для достижения гибкости и повышения точности тестирования ошибок связанных с динамическим выполнения кода JavaScript на стороне пользователя в работе используется платформа Selenium GRID, представляющая собой распределенную систему управления веб-браузерами, построенную на основе Selenium WebDriver и предназначенную для автоматизации тестирования веб-приложений с динамическим содержимым. Применение данной платформы позволяет выполнять http-запросы через веб-браузер в фоновом режиме и получать доступ к результатам выполнения таких запросов. Ключевой особенностью Selenium WebDriver является возможность получения доступа к DOM-модели html-страницы, а также к переменным, структурам и функциям языка JavaScript, ассоциированных с данной веб-страницей. В свою очередь Selenium WebDriver позволяет запускать одновременно несколько веб-браузеров, что является достаточно полезным для тестирования особенностей обработки запросов определенным типом веб-браузера. Для запуска отдельных веб-браузеров используется контейнеризация. Каждый веб-браузер запускается в Docker-контейнере с соответствующим типом Selenium WebDriver.

Интерфейс Selenium GRID позволяет контролировать создание необходимого числа браузеров определенной версии и типа, осуществлять передачу и выполнение всех необходимых запросов напрямую в веб-браузере, а также извлекать результаты обработки запросов в приложении, непосредственно взаимодействующим с Selenium. Общий сценарий проведений тестирования ошибок межсайтового скриптинга МСС приведен на рисунке 5.11

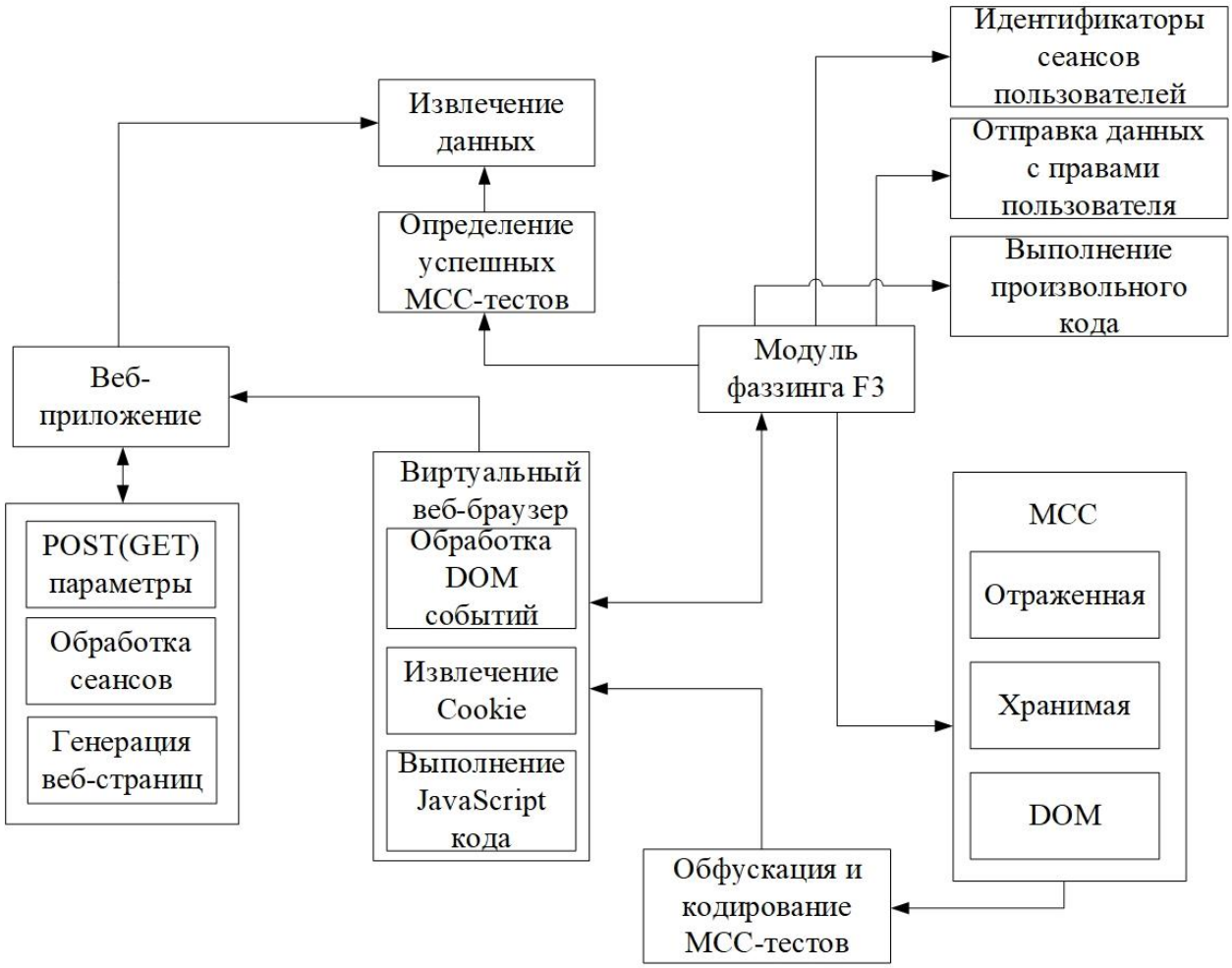

Рисунок 5.11.Структурная схема фаззинга МСС-ошибок

Рисунок 5.11 показывает, что в процессе тестирования МСС, в качестве дополнительных механизмов используется обфускации, заключающаяся в запутывании кода, а также различные механизмы кодирования. Использование данных механизмов направлено на определение эффективности функционирования механизмов фильтрации входных данных, блокировании недокументированных действий и снижении сложности тестирования веб-приложений с динамическим содержимым. Функциональной единицей Selenium

GRID является виртуальный веб-браузер, запускаемый с определенной версией и модулем обработки веб-страниц. В такой схеме модуль фаззинга выполняет роль управляющей программы, координируя действия веб-браузеров, функционирующих внутри Selenium GRID посредством удаленных вызовов (RPC-вызовов), а также осуществляя синхронизацию событий, обрабатываемых в процессе анализа факта наличия МСС в целевом веб-приложении. В этом случае, каждый сеанс браузера будет иметь соответствующий набор атрибутов, а также свою область хранения промежуточных данных. Такой подход позволяет формировать цепочки МСС тестов, сочетающие в себе различные механизмы тестирования МСС, а также оптимизировать данные тесты для анализа ошибок определенных модулей обработки веб-страниц (Gecko, WebKit). В рамках конвейера Docker можно запустить N коллекций однотипных (разнотипных) веб-браузеров для одного GRID-узла. В таком случае каждый GRID-узел может содержать коллекцию веб-браузеров лишь определенной версии или с одним модулем обработки. Тем самым можно повысить интенсивность тестирования МСС и оптимизировать распределение N тестовых генераций между несколькими GRID-узлами, одновременно формируя тесты для конкретных типов веб-браузеров. Использование данных подходов в рамках моделирования процессов фаззинга позволяет повысить точность обнаружения ошибок, а также исключить из выборок, формируемых по результатам выполнения тестов, ту область данных, которая является несогласованной и не может быть использована в рамках решения задач тестирования с использованием ДБС.

5.3. Анализ результатов вычислительного эксперимента

Результаты экспериментальной части исследования затрагивают все вышеизложенные методы и алгоритмы обучения структуры, параметров, а также вероятностного вывода. ДБС используются для моделирования процессов тестирования ошибок согласно требованиям OWASP. Каждый узел ДБС описывает соответствующие механизмы тестирования, а также вспомогательные средства, реализующие функционал преобразования тестовых данных с целью оценки

правильности фильтрации входных параметров приложений. В результате выполнения алгоритмов тестирования методом фаззинга с использованием сценариев тестирования, представленных на рисунках 5.8 и 5.11 получим обучающую выборку необходимую для построения направленного графа G, лежащего в основе ДБС и позволяющую получить начальное распределение вероятностей $P(X_0)$, модели $P(X_{t+1}|X_t)$ и $P(E_{t+1}|X_{t+1})$. Формирование ДБС происходит на основе статистических данных, формируемых в результате тестирования приложений, представленных в таблице 5.1 на $t+k$ временных срезах. Построение ненаправленного графа ДБС происходит в предположении об условной независимости $I(X;Y|Z)$ на основе определения марковского покрытия $M(X)$ и проверки G^2-тестов. Результаты статистического анализа данных используются для принятия решения о добавлении, удалении узлов в состав родительских (дочерних) вершин для переменной X. Применение алгоритмов поиска наибольших (наименьших) оценок позволяет оценить правдоподобие связей между узлами сети за счет добавления, удаления и инверсии направленности дуг между узлами ДБС, включая транзитивные связи. Данная процедура, согласно представленному алгоритму МПСО, сводится к решению экстремальной задачи. В процессе изменения структуры ДБС меняется значение локальной оценки для каждой из дуг сети на основе критериев максимального правдоподобия, эквивалентного критерия Байеса-Дирихле, Шварца или Акаике.

Для проведения практической части определим аппаратную конфигурацию, построенную на основе платформы Amazon EC2. Данная платформа представляет собой IaaS/PaaS среду, с реконфигурируемым числом физических узлов, предназначенную для быстрого развертывания облачных сервисов обработки данных, в частности Hadoop и Spark.

Данная конфигурация представлена 10 узлами, имеющими следующие вычислительные характеристики:

1. 2 процессора Intel Xeon Gold 2.8 ГГц с 16 ядрами, 128 ГБ ОЗУ;
2. жесткий диск 10 ТВ для файловой системы HDFS;

С использованием данной аппаратной платформы развернута система распределенных вычислений Hadoop и Spark. Хранение данных организовано посредством распределенной файловой системы HDFS. Параллельные задачи Spark запускаются в виде последовательности MapReduce-задач Hadoop. Чтение обучающей выборки производится непосредственно из HDFS в RDD-коллекции Spark. Такой подход позволяет оптимизировать обработку большого объема обучающей выборки за счет предварительной ее загрузки в виде RDD-сущностей в память кластера серверов Amazon EC2 для последующего кэширования. Непосредственно перед практическим выполнением экспериментов по обучению и вероятностному выводы ДБС, а также в процессе анализа эффективности данных алгоритмом, оценим разработанный алгоритм синхронизации. Данный алгоритм является расширением существующих подходов синхронизации Spark на основе использования СМО. В качестве абстрактной сущности синхронизации используется понятие «широковещательной переменной». Данная переменная применяется в параллельных заданиях Spark. Это дает возможность предварительной инициализации переменных для исключения повторной инициализации а процессе распределения заданий. Переменные хранятся в памяти каждого узла кластера в виде RDD-сущностей [92]. Программная реализация описанного в главе 4 алгоритма на основе СМО реализована в виде отдельного класса, унаследованного от базового класса Spark, отвечающего за широковещательный обмен данными между узлами кластера, в процессе выполнения набора параллельных задач.

Приведем модель системы синхронизации, описанную в виде модели массового обслуживания, состоящей из P_n вычислительных ядер процессоров, функционирующих в рамках кластера распределенной системы обработки и выполняющих обработку заданий в k параллельных потоках, соответствующих числу доступных вычислительных ядер распределенной системы Spark. В процессе выполнения синхронизации выполняется широковещательная рассылка информационных сообщений между потоками-получателями с целью исключения взаимных блокировок.

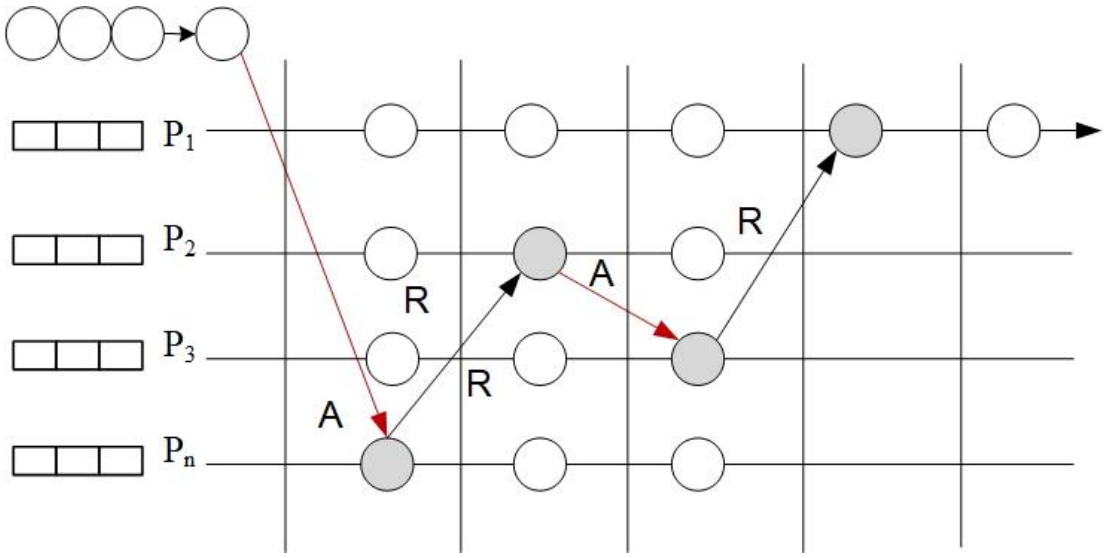

Рисунок 5.12. Процесс синхронизации в виде модели СМО.

Приведем общую характеристику рассматриваемых алгоритмов в рамках исследования временных показателей существующих и разработанных алгоритмов синхронизации в таблице 5.3

Таблица 5.3. Описание сравниваемых алгоритмов синхронизации ресурсов

№ п/п	Алгоритм	Описание алгоритма
1.	BitTorrent	Реализация децентрализованного протокола BitTorrent (встроенный алгоритм Spark).
2.	YARN [5]	Алгоритм синхронизации, встроенный в программную модель YARN Hadoop (5 реплик).
3.	YARN [10]	Алгоритм YARN (10 реплик).
4.	СКСМО	Разработанная система синхронизации на основе алгоритма Сузуки-Касами и системы массового обслуживания.

Далее приведем результаты численных экспериментов для алгоритмов синхронизации, приведенных в таблице 5.3 (время выполнения в зависимости от объема данных, требующих синхронизации). Зависимость времени синхронизации от числа потоков-получателей приведем для объема передаваемых данных в 100 ГБ.

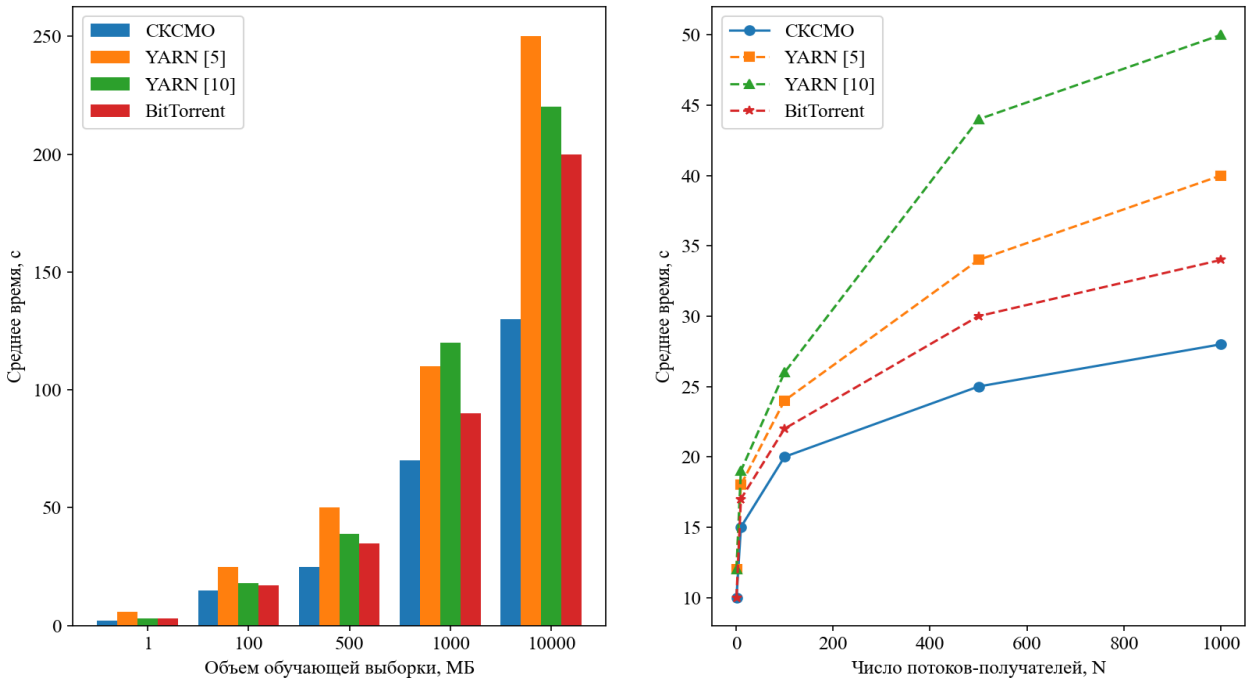

Рисунок 5.13. Результаты работы алгоритмов синхронизации для Spark.

Из рисунка 5.13 видно, что разработанный алгоритм СКСМО позволяет значительно сократить время синхронизации одного и того же объема данных по сравнению с алгоритмами BitTorrent, YARN [5], YARN [10] на одном и том же объеме передаваемых данных.

Применение разработанного алгоритма предоставляет возможность оптимизации процедуры распределения данных между RDD-сущностями Spark. При этом наблюдаем значительную устойчивость алгоритма к росту числа потоков-получателей, что свидетельствует об адаптируемости разработанного подхода к решению задач синхронизации в процессе распределения разделяемых ресурсов большого объема при росте числа доступных ядер процессоров в кластере и масштабируемости параллельной платформы Spark. Алгоритм СКСМО позволяет настраивать максимальное время ожидания синхронизации, что достаточно важно, если объем памяти узлов ограничен, а число широковещательных переменных достаточно велико. При единовременной инициализации нескольких широковещательных переменных, планировщик заданий не будет производить запуск параллельных шагов до того момента, пока все переменные не переместятся в RDD-области каждого из узлов кластера.

Spark может быть использован для хранения сложной структуры динамических моделей ДБС в оперативной памяти при реализации механизмов фаззинга. В процессе экспериментальной части тестирования веб-приложений методом фаззинга на основе применения моделей ДБС в качестве широковещательных переменных на этапе обучения структуры будет выступать множество переменных сети, используемых для формирования как направленного, так и ненаправленного графа ДБС. На этапах обучения параметров и вероятностного вывода будем рассматривать модель ДБС с набором переменных X_t, X_{t+1}, E_{t+1} и соответствующим им набором таблиц условных вероятностей $P(X_0), P(X_{t+1}|X_t)$ и $P(E_{t+1}|X_{t+1})$.

Далее перейдем к численному эксперименту процедур обучения и вероятностного вывода ДБС фаззинга. Рассмотрим эксперимент по обучению структуры и параметров ДБС фаззинга «Инъекции» и «Межсайтовый скриптинг» на основе разработанного алгоритма МПСО и ОМ по схеме Гиббса (ОМГ).

В начале выполним оценку правильности определения структуры байесовской сети в соответствии с алгоритмом МПСО [171]. Для этого возьмем множество существующих сетей с заранее известной структурой: HAILFINDER (223 узла, 338 ребер, 1157 параметров), DIABETES (56 узлов, 2656 параметров), ANDES (223 узла, 1157 параметров), LINK (724 узла, 14211 параметров). Рассмотрим процедуру оценки CPX всех описанных ранее сетей в зависимости от общего размера выборки, используемой в процессе обучения. Установим фактор зависимости точности алгоритмов обучения от общего размера обучающей выборки S. Правдоподобие структуры обратно пропорционально дистанции CPX и максимум правдоподобии структуры будет достигаться при минимальных значениях CPX.

На рисунке 5.14 приведена зависимость численной оценки CPX от объема обучающей выборки N каждой из сетей HAILFINDER , DIABETES и LINK для алгоритмов МПСО, минимаксного восхождения (ММВ), возрастания-сокращения (ВС) и алгоритм Питера-Кларка (ПК) [185].

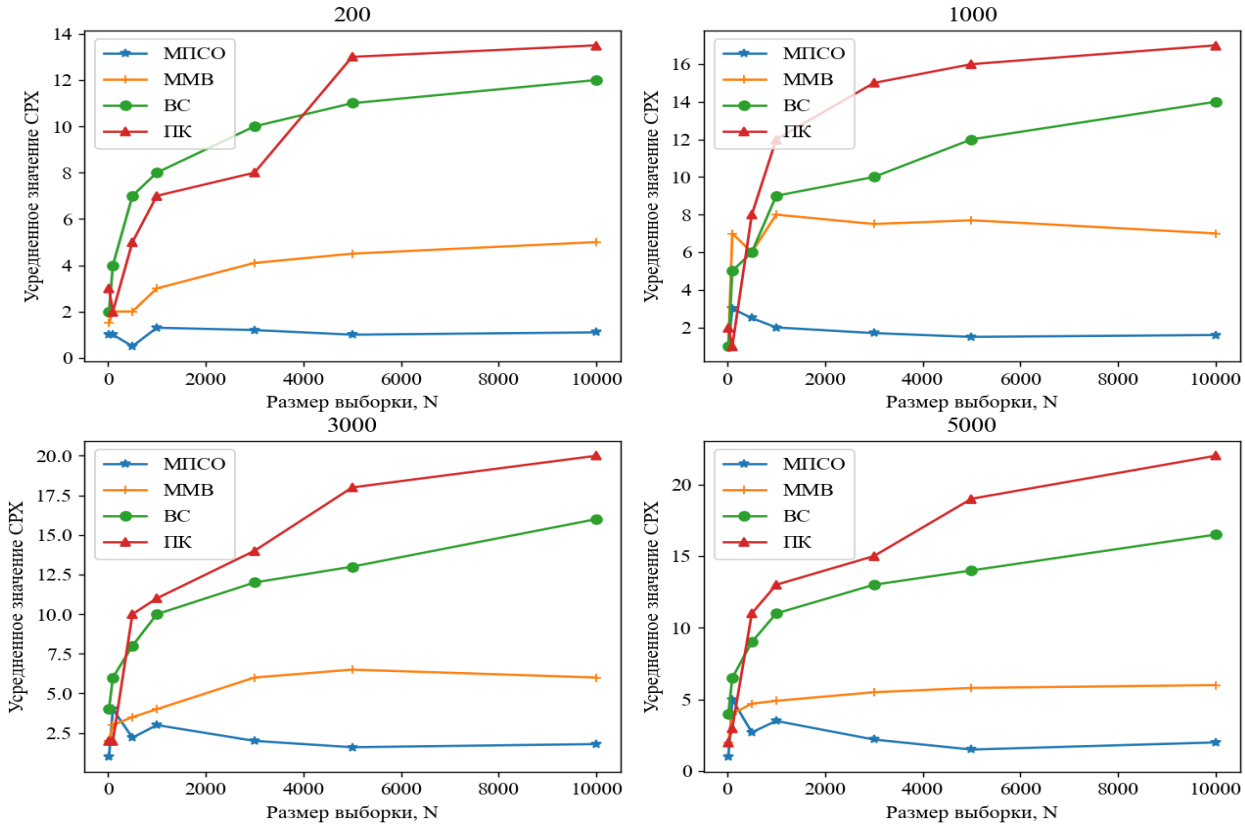

Рисунок 5.14. Усредненное значение структурного расстояния Хэмминга для числа параметров сети 200, 1000, 3000 и 5000.

Из рисунка 5.14 следует, что представленный в исследовании алгоритм МПСО обладает наименьшим значением СРХ, что является показателем точности определения структуры БС и ДБС за счет реализации инструментов статистического оценивания и решения задачи по поиску максимума критериальной функции, представленной одним из следующих критериев: логарифм правдоподобия, информационный критерий Акаике (АИК), эквивалентная метрика Байеса-Дирихле (БДЭ) или Байесовский информационный критерий (БИК). В рамках экспериментальной части работы будет применяться разработанный алгоритм МПСО для построения структуры ДБС, предназначенной для моделирования процессов тестирования веб-приложений с применением фаззинга, в частности, для анализа ошибок типа «инъекция» и «межсайтовый скриптинг». Переходя к процедуре численной оценки алгоритмов построения структуры ДБС для анализа ошибок методом фаззинга, определим состав и размер обучающей выборки, необходимой для проведения эксперимента.

Обучающая выборка представлена двумя файлами, имеющими размер 100 ГБ, полученных по результатам фаззинга «Инъекций» и «Межсайтового скриптинга» (МСС) веб-приложений в соответствии с таблицей 5.1. В процессе реализации процедуры обучения структуры и параметров, каждая строка выборки обрабатывается в распределенном режиме, общее время обучения будет пропорционально числу доступных процессоров N распределенной вычислительной системы. В процессе проведения эксперимента по обучению ДБС «Инъекции» и «Межсайтовый скриптинг» проведем анализ временных и количественных показателей разработанных алгоритмов МПСО и ОМГ. Отметим, что в случае полного наблюдения, алгоритм ОМГ может быть заменен классическим алгоритмом максимального правдоподобия, так как схема Гиббса используется для заполнения скрытых (пропущенных) данных в обучающей выборки. В процессе реализации обучения параметров ДБС установим ограничение по числу выборок, которые необходимо сгенерироваться в процессе выполнения алгоритма для ДБС. Эмпирическим путем нами установлено, что наиболее оптимальным значением будет является число выборок $N = 1000000$.

Для оценки точности обучения параметров на основе разработанного алгоритма ОМГ воспользуемся перекрестной энтропией (ПЭ) для распределений $P(X)$ и $Q(X)$, соответствующих алгоритмам обучения параметров на основе ОМГ и МП соответственно. Выразим ПЭ через метрику Кульбака-Лейблера (2.61) определенную нами в параграфе 2.2 для дискретного случая распределений $P(X)$ и $Q(\mathrm{X})$. Получим следующее выражение [44]:

$$H\big(P(X), Q(X)\big) = H\big(P(X)\big) + D_{KL}\big(P(X), Q(X)\big) = \sum_{X_1, X_2, \dots, X_n} P(X_1, X_2, \dots, X_n) \frac{log\big(P(X_1, X_2, \dots, X_n)\big)}{log\big(Q(X_1, X_2, \dots, X_n)\big)}. \quad (5.11)$$

В соответствии с формулой (5.11), чем меньше значение $H\big(P(X), Q(X)\big)$, тем выше точность формирования распределения Q(X), полученного по результатам выполнения алгоритма ОМГ. Для проверки корректности вычисленных значений возьмем структуры для БС HAILFINDER, DIABETES и LINK, обученные на основе

алгоритма МПСО и произведем обучение параметров методом МП и ОМГ, после чего вычислим значения ПЭ. Для этого перепишем выражение (5.11) с учетом семантики ДБС

$$H\big(P(X),Q(X)\big)=\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}P(X_i=k|Y_i=j)\,ln\frac{P(X_i=k|Y_i=j)}{Q(X_i=k|Y_i=j)},$$
$$q_i=\prod_{X_l\in Y_i} r_l. \qquad (5.12)$$

где Y_i родительская вершина, ассоциируемая с переменной X_i, r_i – число возможных состояний, принимающих X_i, q_i – число состояний для Y_i.

Значение ПЭ напрямую зависит от точности определения структуры, так распределение $P(X_i=k|Y_i=j)$ формируется с учетом родительских вершин. При проведении практической оценки точности расчета распределений $P(X_0)$, $P(X_t|X_{t-1})$ и $P(E_t|X_t)$ для HAILFINDER, DIABETES и LINK будем использовать усредненное значение энтропии ПЭ. На рисунке приведем зависимость усредненной ПЭ от размера обучающей выборки

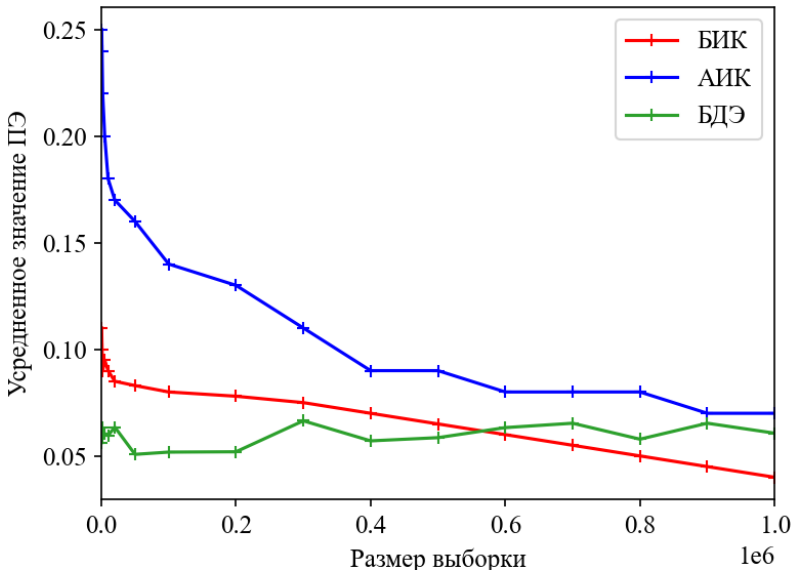

Рисунок 5.15. Усредненное значение ПЭ для обученных сетей HAILFINDER, DIABETES и LINK

Рисунок 5.15 показывает, что при обучении ДБС в независимости от объема выборок наиболее целесообразно использовать метрику БДЭ или критерий БИК. В этом случае получаем минимум несоответствия распределений $P(X_0)$, $P(X_t|X_{t-1})$ и

$P(E_t|X_t)$. Перейдем к численным экспериментам для ДБС фаззинга ошибок. Для этого проведем сравнение значений критериев АИК и БИК относительно эквивалентной метрики Байеса-Дирихле в процессе построения структуры ДБС «Инъекции» и «Межсайтовый скриптинг». Отметим, что данные критерии используются в процессе определения направленности связей с помощью алгоритма имитации отжига. В этом случае важно оценить, насколько целесообразно использовать оценочные функции, формируемые на основе логарифма правдоподобия (L), в процессе обучения структуры ДБС фаззинга ошибок веб-приложений. Оценку погрешности L, АИК и БИК, формируемых на основе метода имитации отжига, будем сравнивать относительно метрики БДЭ с разными значениями параметра распределения Дирихле $\alpha = \{0.1; 1; 10\}$. Максимальное число обучающих выборок $D = 1000000$. Значения результатов для разных оценочных функций приведены на рисунке 5.16

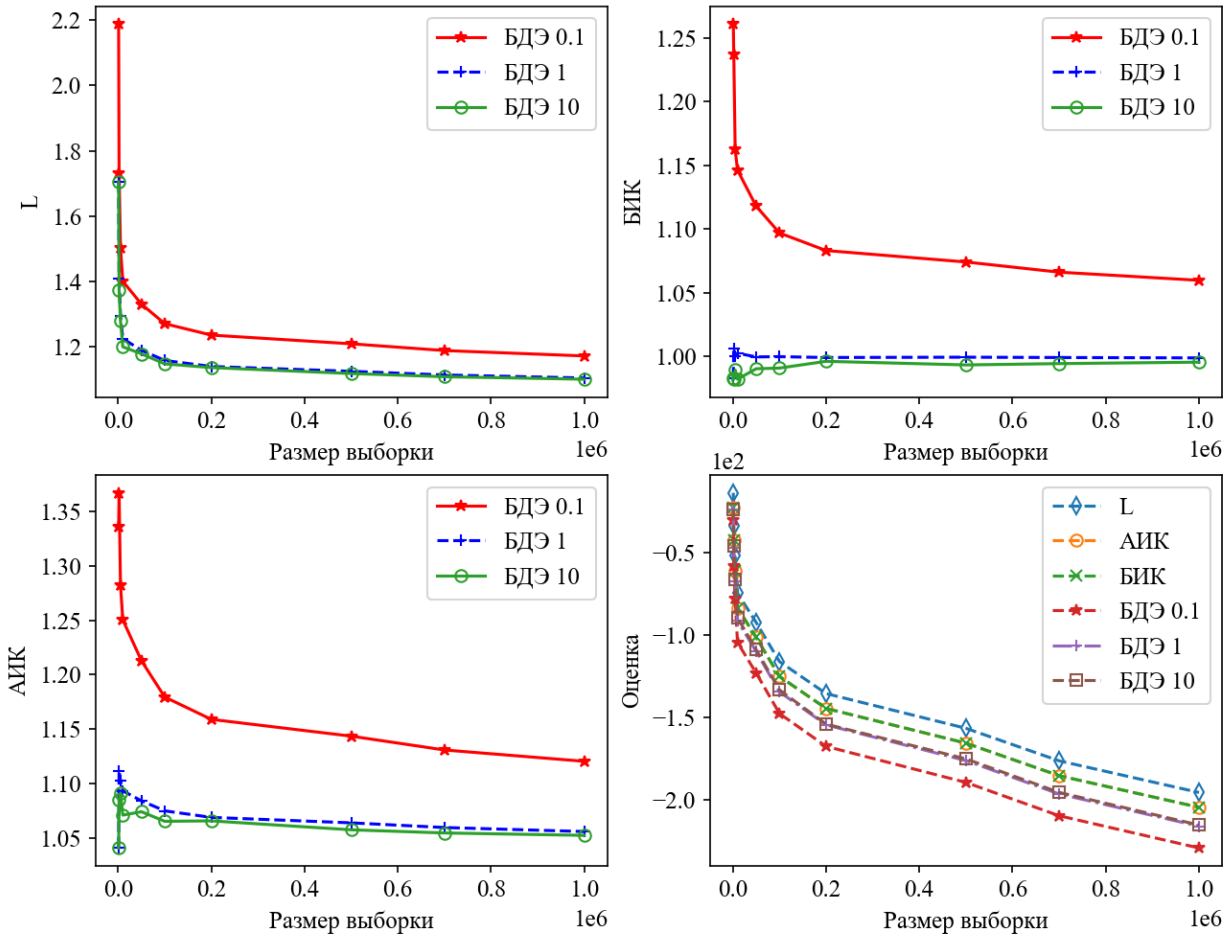

Рисунок 5.16. Сравнение результатов, полученных при разных оценочных функциях и при разных объемах выборок в процессе определения структуры ДБС

Практические эксперименты показывают, что оценки на основе БИК и АИК в процессе обучения структуры дают близкие результаты. Проведем сравнение числа оценок, формируемых на основе БИК и БДЭ, необходимых для выполнения разработанного алгоритма МПСО, с существующими алгоритмами ММВ, ВС для вычисления оценок ненаправленных графов «Инъекции» и «Межсайтовый скриптинг», необходимых для определения направленности связей в промежуточном графе, полученном по результатам статистического анализа обучающей выборки. Приведем данное сравнение на рисунке 5.17

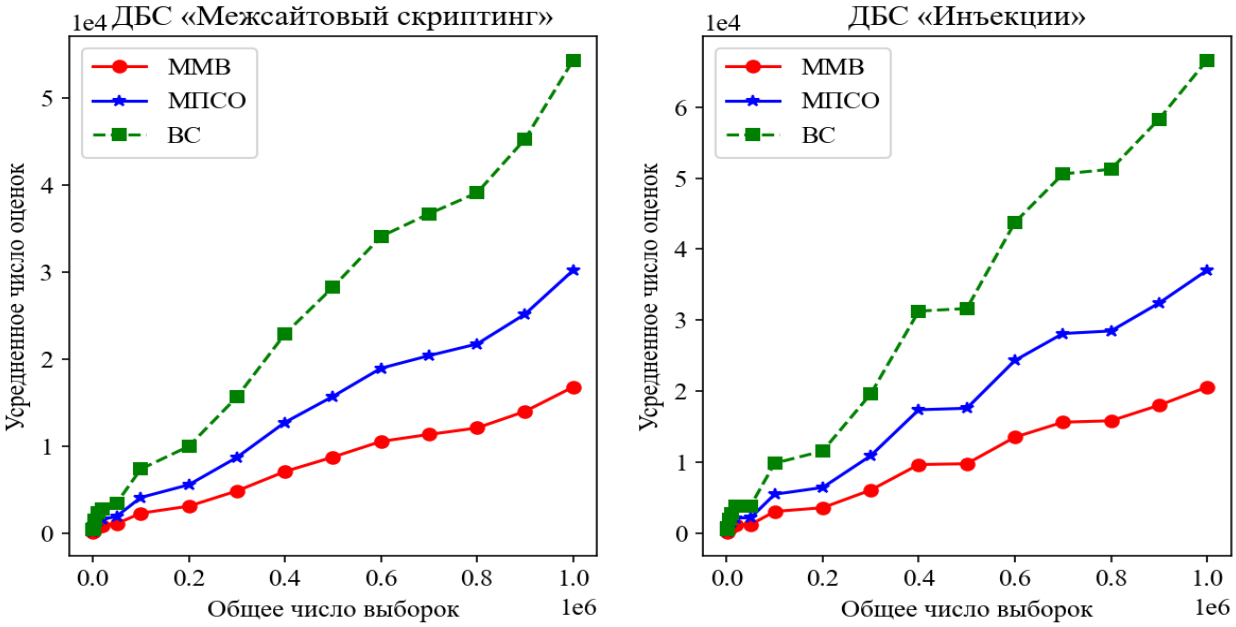

Рисунок 5.17. Усредненное значение числа БИК-оценок от обучающей выборки для ДБС «Инъекции» и «Межсайтовый скриптинг»

Для определения структуры ДБС потребуется выполнить $N = kn - \frac{k(k-1)}{2}$ статистических тестов для алгоритмов ММВ и ВС Для разработанного алгоритма МПСО потребуется $N = kM(V) - \frac{k(k-1)}{2}$ тестов. В состав марковского покрытия размерностью $M(V)$ попадают родительские вершины с минимальным показателем статистики G^2, определяющей условную независимость вершины v от множества остальных узлов $v_i \in V$, входящих в состав ДБС.

Приведем полученные модели ДБС «Инъекции» [189] и «Межсайтовый скриптинг» [191,192], полученные на основе применения разработанного гибридного метода обучения структуры сети.

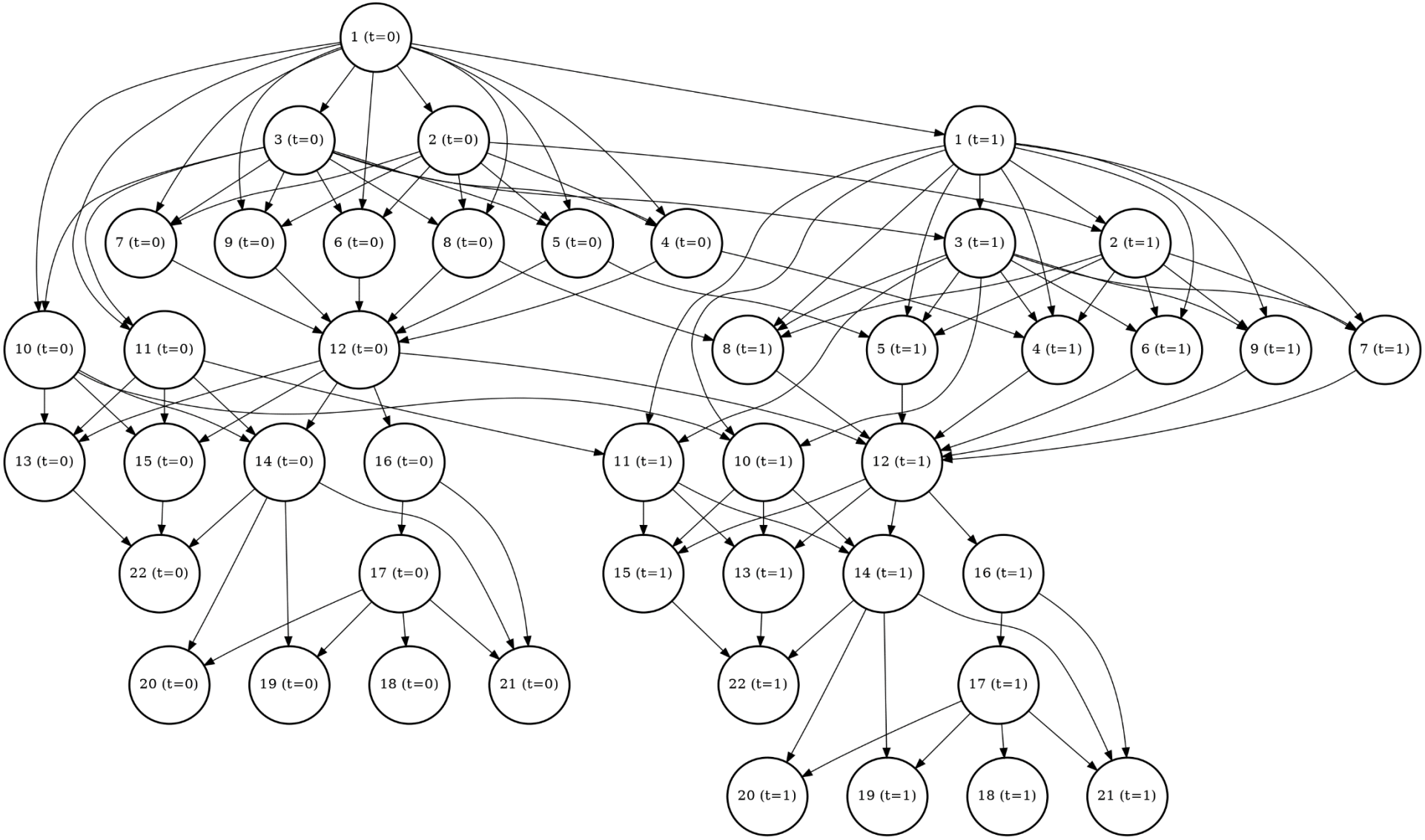

Рисунок 5.18. ДБС тестирования инъекций методом фаззинга

В таблицах 5.4 и 5.5 приведем описание узлов, характерных для ДБС типа «Инъекции» и «Межсайтовый скриптинг». Каждый узел наделяется функционалом, предназначенным для решения определенного типа задач фаззинга. В состав узла входит генератор тестовых данных, предназначенный для тестирования определенного типа ошибок характерных для «Инъекции» и «Межсайтового скриптинга».

Таблица 5.4

Характеристика узлов ДБС фаззинга инъекций

№ п/п	Название узла	Характеристика
1.	Inject Type	Выявление типа инъекции: SQL, команд, кода.
2.	Http Parameter Pollution	Механизмы «смешивания» параметров http запроса (Таблица 5.2).
3.	Encoder	Преобразования определенных параметров, отправляемых в веб-приложение.
4.	Union Injection	Объединенная инъекция.
5.	Boolean Based Blind	Логическая инъекция.
6.	Time Blind	Временная инъекция.
7.	Error Blind	Инъекция на основе анализа ошибок.
8.	Stacked Time	Разделенная инъекция.
9.	Out of Band	Внешняя инъекция.
10.	Code	Инъекция кода.
11.	Command	Инъекция команд.
12.	DBMS Fingerprint	Извлечение типа и версии СУБД для оптимизации процедуры формирования тестов
13.	Network	Получение доступа к сетевым компонентам через межпрограммные интерфейсы СУБД и операционной системы.
14.	Cmd Execution	Выполнение произвольных команд посредствам СУБД и командной строки операционной системы.
15.	File System	Реализация механизмов чтение и запись файлов через встроенные функции СУБД.
16.	Db Structure	Определение структуры баз данных и таблиц
17.	Table Data	Извлечение табличных данных.
18.	Confidentiality	Конфиденциальность.
19.	Authentication	Аутентификация.
20.	Authorization	Авторизация.
21.	Integrity	Целостность.
22.	Availability	Доступность.

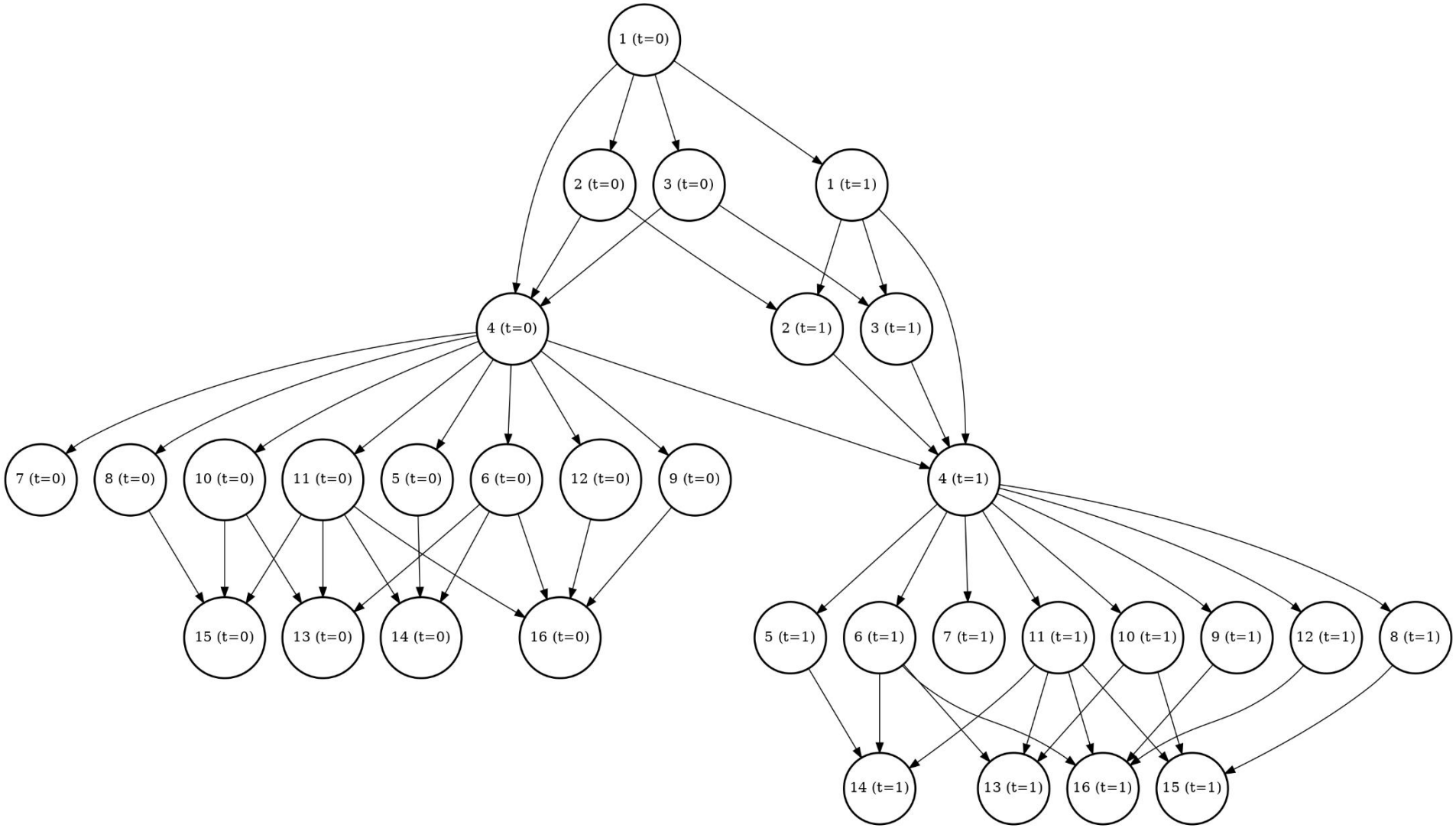

Рисунок 5.19. ДБС тестирования ошибок межсайтового скриптинга методом фаззинга

Таблица 5.5

Характеристика узлов ДБС фаззинга межсайтового скриптинга

№ п/п	Название узла	Характеристика
1.	XSS Type	Выявление типа межсайтового скриптинга.
2.	Encoder	Преобразования определенных параметров, отправляемых в веб-приложение.
3.	Evasion	Механизмы «запутывания» (преобразования) полезной нагрузки.
4.	Xss Payload	Разновидность полезной нагрузки (html-тэги, обработчики событий)
5.	Keylogger Module	Запоминание комбинаций клавиш, нажатых пользователем.
6.	Spy Eye Module	Получения «снимка» html-страниц активных вкладок и окон веб-браузера пользователя.
7.	DDos Module	Выполнение атаки отказа в обслуживании на внешние ресурсы и сервисы.
8.	Port Scanner Module	Определение открытых портов на компьютере пользователя.
9.	Network Scanner Module	Сканирование локальной сети пользователя.
10.	Nat Pinning Module	Обход сетевых правил маршрутизатора NAT (проникновение в локальную сеть)
11.	Drive By Download Module	Перенаправление пользователя на ресурсы, содержащие вредоносные программы и вирусы.
12.	Browser Fingerprint	Определение типа веб-браузера, списка установленных модулей и компонентов.
13.	Authentication	Аутентификация.
14.	Authorization	Авторизация.
15.	Integrity	Целостность.
16.	Availability	Доступность.
17.	Confidentiality	Конфиденциальность.

Используя PaaS-среду Amazon EC2, проведем оценку ускорения разработанных параллельных алгоритмов при обучении структуры и параметром ДБС «Инъекции» и «Межсайтовый скриптинг». Для этого будем использовать изменение числа используемых ядер кластера EC2 путем конфигурирования соответствующих параметров параллельной платформы Spark: «spark.driver.core», «spark.cores.max» и «spark.executor.core». Для эмуляции однопоточного режима в Spark будем использовать значение параметра конфигурации «spark.driver.core=1».

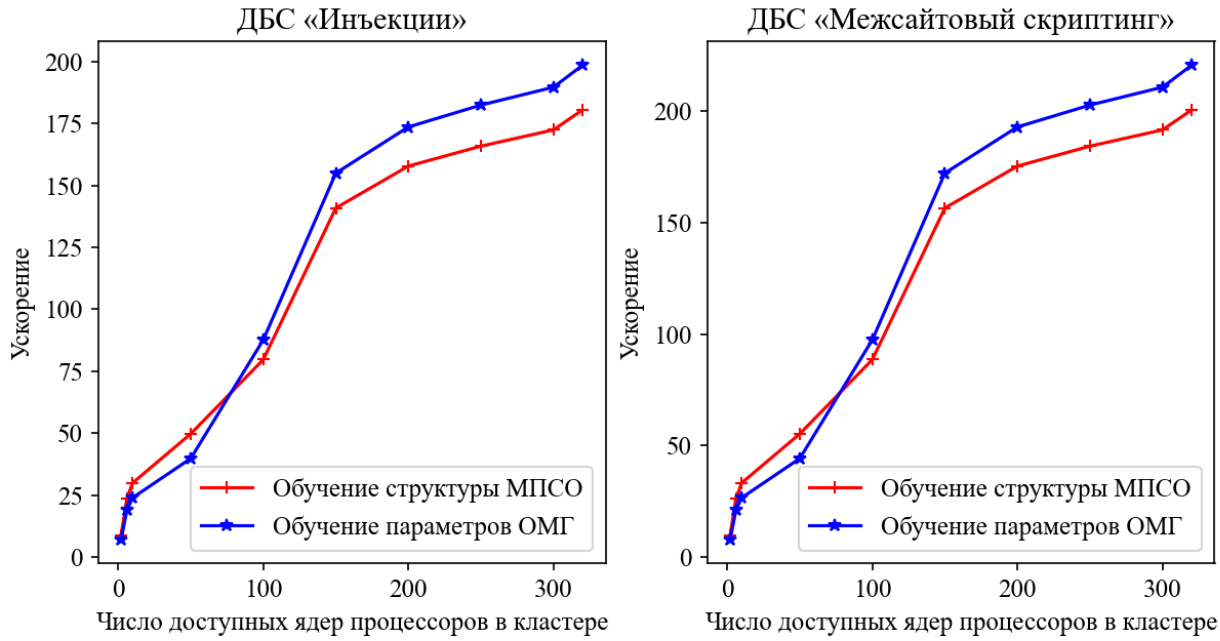

Рисунок 5.20. Зависимость ускорения параллельных алгоритмов МПСО и ОМГ в процессе решения задач обучения структуры и параметров ДБС «Инъекции» и «Межсайтовый скриптинг»

Рисунок 5.17 показывает, что наибольшим ускорением обладает алгоритм ОМГ, это достигается за счет того, что в процессе выполнения алгоритма нет необходимости оценки условной независимости вершин, а также применения алгоритма имитации отжига для решения задачи поиска максимума оценочной функции, необходимого для определения направленной ДБС. Применение процедур распределенного обучения ДБС показывает свою высокую эффективностью. Зависимость производительности алгоритмов от повышения сложности топологии сети, а также объема обучающей выборки будет определяться производительностью аппаратной платформы и общим числом доступных ядер процессоров распределенной вычислительной системы.

В результате выполнения процедур обучения структуры и параметров ДБС «Инъекции» и «Межсайтовый скриптинг» получим необходимые распределения вероятностей: начальное распределение $P(X_0)$, распределения для модели перехода $P(X_{t-1}|X_t)$ и восприятия $P(E_t|X_t)$. Данные распределение приведены в приложении А. В связи с большим размером таблиц условных вероятностей исследуемых ДБС, для упрощения записи, в таблицах распределений параметров ДБС «Инъекции» и «Межсайтовый скриптинг» приведены только ненулевые вероятности с

соответствующими им индексами, форма записи имеет вид «{индекс: вероятность}». Априорные и апостериорные распределения вероятностей, соответствующие ДБС «Инъекции» и «Межсайтовый скриптинг» приведены в приложениях А и Б. Отметим, что в эксперименте учитывается полное совместное распределение вероятностей каждой из переменных ДБС.

Процедура выявления новых типов ошибок (аномалий) тесно связна с решением задачи вероятностного вывода в ДБС. Рассматривая характеристики узлов двух ДБС, представленные в таблицах 5.4 и 5.5, можно сформировать две основные группы параметров в соответствии с их функциональным назначением: сканирование и эксплуатация. Первая группа S используется для поиска программной ошибки, а также выявление уязвимых параметров. Вторая E направлена на получение доступа к информационной системе, в рамках которой функционируют приложения за счет эксплуатации определенного типа программных ошибок. Рассмотрим предлагаемый подход к анализу аномальных ошибок веб-приложений с использованием ДБС фаззинга, реализуемой с учетом спецификации OWASP. Данный подход заключается в решении задач вероятностного вывода на временном интервале $(t; \mathrm{t} + k)$ за счет использования стохастических алгоритмов на основе многочастичного фильтра. Рассмотрим особенности решения задач вероятностного вывода применительно к фаззингу ошибок веб-приложений.

В процессе фильтрации необходимо найти распределение $P(X_{t+1}|E_{1:t+1})$, характеризующее вероятность нахождения ошибок в момент времени $t + 1$. В таком случае процесс фильтрации будет включать в себя один шаг предсказания. Свидетельства $E_{1:t+1}$ характеризуют состояние системы: целостность, доступность и конфиденциальность [195,186]. Это позволяет оценить возможность обнаружения ошибок для текущего состояния системы.

Решение задачи предсказания позволяет вычислить апостериорное распределение вероятностей для всех ненаблюдаемых переменных X_{t+k} в будущих состояниях системы $P(X_{t+k+1}|E_{1:t+1})$. Процедура прогнозирования для ДБС фаззинга может быть определена в виде процесса обнаружения ошибок в будущем с учетом динамики их появления в прошлых состояниях. Такой подход позволяет

произвести настройку системы тестирования для обнаружения ошибок определенной группы приложений, имеющей схожий набор компонентов и модулей, а также способствует выявлению аномальных ошибок. Интерес с точки зрения корректировки тестов представляет решение задачи сглаживания. Качественное определение распределения $P(X_k|E_{1:t})$, характеризующего оценку системы в прошлых состояниях, дает возможность произвести исключение неинформативных с точки зрения обнаружения ошибок тестов. Также можно произвести оценку эффективности реализации процедур тестирования для каждого из прошлых состояний. Имитационную модель детектирования ошибок на основе вероятностного вывода ДБС можно представить в следующем виде

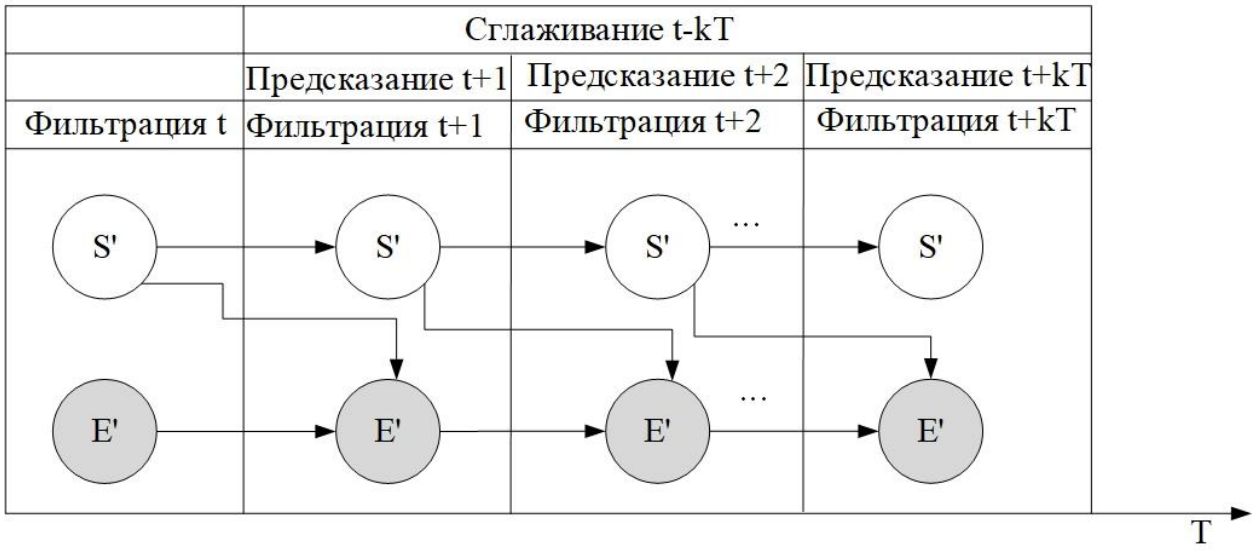

Рисунок 5.21. Модель перехода и восприятия процесса обнаружения ошибок

Если в результате выполнения процедуры сглаживания на срезе $\mathrm{t} - KT$ будет обнаружена новая ошибка, то свидетельства E_{t-KT} относительно нее будут включены в новое предсказание для шага $t + KT$ и все полученные ранее вероятности будут пересчитаны алгоритмом многочастичного фильтра с учетом данных свидетельств. Следовательно, задача выявления новых типов ошибок сводится к определению распределения вероятности $P(S_{t+\mathrm{KT}}|E_{1:t+KT})$ при наличии свидетельств $E_{1:t+KT}$ подтверждающих успешность их применения для обнаружения определенных типов программных ошибок [194].

Используя распределение вероятностей фильтрации и предсказания, можно получить оценку факта наличия определенного типа ошибок, опираясь на

результатах процедуры байесовского вывода для всех переменных запроса при наличии непрерывного потока свидетельств. Особую роль при обнаружение аномальных ошибок играет процедура предсказания, так она позволяет оценить динамику поведения программных ошибок на несколько срезов вперед. Следовательно, это дает возможность оценить факт локализации и исключения определенных ошибок в процессе выпуска патча или обновления для веб-приложения и сформировать возможные пути ее устранения в соответствии с тестовой выборкой, давшей положительный результат. Наряду с предсказанием и фильтрацией при реализации МЧФ необходимо учесть процедуру сглаживания. Отличительной особенностью сглаживания является возможность дать оценку правдоподобия прошлых состояний в процессе сканирования и эксплуатации, так как на момент kT уже получены все свидетельства [188,190].

При проведении численного эксперимента тестирования веб-приложений для верификации разработанного алгоритма рассмотрим две основополагающие методики тестирования веб-приложений для верификации предложенного алгоритма. Первая методика характеризуется реализацией тестирования на основе разработанного алгоритма МЧФ и ЛШ, вторая – тестирование, выполняемое на основе фаззинга методом черного ящика. Для интервала времени $t+1$ будем использовать межсетевой экран безопасности ModSecurity с предустановленными правилами, имитирующий закрытие ошибок веб-приложений за счет применения дополнительных средств защиты. Отметим, что для второй методики, результирующее распределение для фаззинга по методу черного ящика формируется путем обучения параметров моделей ДБС на основе алгоритма ОМГ или МП, в случае частичной или полной наблюдаемости. В процессе вероятностного вывода решаем задачу декодирования Витерби $\max_{X_1,X_2,\dots,X_t} P(X_1,X_2,\dots,X_t,X_{t+k}|E_{1:t+\mathrm{k}})$ вплоть до среза $t+\mathrm{k}$. Одновременно с решением данной задача, оценим правдоподобие распределений, полученных на основе МЧФ и МЧФ и ЛШ, путем вычисления расстояния КЛ (2.61) в сравнении с эталонными распределениями, полученными в результате практического тестирования методом черного ящика.

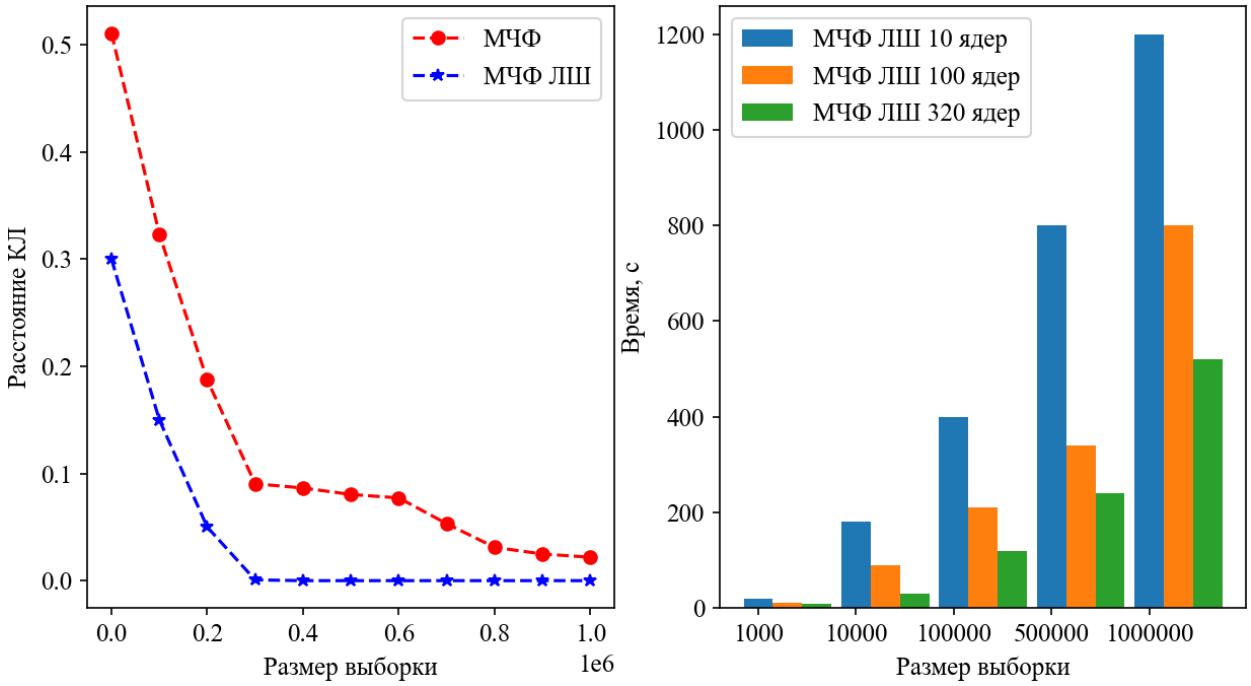

Рисунок 5.22. Оценка расстояния КЛ и производительности алгоритма МЧФ и ЛШ

Рисунок 5.22 показывает, что применение теоремы ЛШ позволяет получить минимальное значение расстояния КЛ даже на небольшом объеме выборок, в то время как для классического алгоритма МЧФ требуется значительное увеличение числа выборок для достижения того же значения расстояния ДЛ. Использование теоремы ЛШ дает возможность применения разработанных алгоритмов не только в кластерных многоядерных параллельных системах, но и персональных компьютерах, а также позволяет получить допустимую точность апостериорного распределения $P(X_{t+1}|E_{t+1})$. Альтернативным подходом является использование предобученных моделей фаззинга. Для примера в работе рассмотрена возможность использования полученных моделей в качестве основы для анализа ошибок веб-приложений в рамках нескольких программных продуктов с открытым исходным кодом: межсетевой экрана ModSecurity. Исследована возможность преобразования полученных моделей в соответствующие правила данного программного продукта. Такой подход расширяет возможности использования разработанного инструментария и позволяет оптимизировать процедуру обучения и обеспечить оптимальную настройку существующих средств обнаружения сетевых

программных ошибок, связанных с функционированием веб-приложений и взаимодействующих с ними компонентов.

Для верификации разработанного алгоритмического и программного обеспечения произведем оценку эффективности обнаружения 200 программных ошибок веб-приложений с 30 ранее неизвестными ошибками при тестировании 32 приложений. Сравним с результатами тестирования наиболее известными сканерами веб-приложений для выявления программных ошибок типа «Инъекции» и «Межсайтовый скриптинг: Acunetix, Nessus, OWASP ZAP, Arachi, Paros, Burp Suite Pro, W9scan.

Таблица 5.6

Сравнение результатов тестирования веб-приложений

№ п/п	Инструмент тестирования	Обнаруженные ошибки	Ложные срабатывания	Пропущено ошибок	Аномальные ошибки
1.	Acunetix	172	10	18	2
2.	Nessus	130	50	20	1
3.	OWASP ZAP	170	20	10	0
4.	Arachi	168	15	17	0
5.	Paros	160	5	35	0
6.	Burp Suite Pro	140	22	38	0
7.	W9scan	150	40	10	0
8.	Разработанный фаззер	190	3	7	20

Таким образом, в пятой главе описана структура и результаты комплексного вычислительного эксперимента по моделированию процессов фаззинг-тестирования основных групп ошибок функционирования веб-приложений на основе моделей ДБС. Результаты тестирования предложенного метода синхронизации СКСМО показали снижение времени реализации процесса вычислений по сравнении с алгоритмами YARN [5], YARN [10], BitTorrent при различном количестве вычислительных потоков соответственно в среднем в 2, 4 и 1.5 раза. Разработанный в рамках исследования метод обучения структуры ДБС позволил в сравнении с ММВ, ВС и ПК при вычислении СРХ снизить число ошибочно добавленных и удаленных вершин структуры ДБС соответственно 1.2,

3.6 и 4 раза. При решении задач вероятностного вывода в процессе прогнозирования аномальных ошибок методом МЧФ и ЛШ при 240 тыс. выборок достигается лучшее значение метрики КЛ, чем дает базовый алгоритм МЧФ при 800 тыс. выборок. Метод МЧФ и ЛШ продемонстрировал высокий темп прироста производительности с увеличением количества ядер при использовании инфраструктуры кластера EC2, при 320 ядрах достигнуто повышение ускорения распределенных вычислений по сравнению со 100 и 10 ядрами в 1,5 и 2,3 раза соответственно. В результате применения предложенных в исследовании инструментов организации процесса фаззинг-тестирования на основе моделей ДБС удалось выявить 20 аномальных ошибок, при этом другие инструменты анализа ошибок не выявили более 2 ошибок.

5.4. Выводы

В пятой главе работы:

1. Предложена структура параллельных алгоритмов обучения и вероятностного вывода для динамических сетей организации процесса фаззинг-тестирования веб-приложений. Проведена оценка возможности использования данных алгоритмов в рамках распределенных систем обработки данных.

2. Описана структура вычислительного эксперимента по применению предложенных в работе моделей, методов и алгоритмов организации процесса фаззинг-тестирования разных групп ошибок функционирования веб-приложений.

3. Предложены метрические показатели, позволяющие оценить эффективность разработанных методов обучения и вероятностного выхода применительно к моделям организации процесса фаззинг-тестирования веб-приложений.

4. Проведена оценка временных показателей предложенных методов в рамках масштабирования вычислительной системы. Показано, как суммарное число вычислительных процессоров влияет на общий прирост скорости алгоритмов обучения и логического вывода.

Заключение

В диссертационном исследовании разработаны новые модели, методы и алгоритмы решения задач фаззинга веб-приложений с применением динамических байесовских сетей. Разработаны новые подходы к оптимизации решения задач обучения и вероятностного вывода байесовских сетей на основе численной и статистической оптимизации. Решены задачи повышения эффективности тестирования при анализе комплексной защищенности информационных систем, функционирующих на основе веб-приложений и веб-сервисов. В работе представлены результаты, относящиеся к области синхронизации и применения распределенных систем обработки данных при решении задач обучения и вероятностного вывода. Представлен комплекс инструментальных средств в виде набора проблемно-ориентированных программ для решения, поставленных задач тестирования на основе математического аппарата ДБС.

В диссертационном исследовании получены следующие основные результаты.

1. Предложен новый подход к организации процедуры тестирования веб-приложений методом фаззинга на основе моделированием данного стохастического процесса с помощью комплекса ДБС, отражающий динамику шагов тестирования, моделирующий осуществленные элементы тестирования в виде переменных свидетельств на определенных временных срезах сетей и осуществляющий прогноз целевых переменные запроса планируемых в будущем или нереализованных ранее элементов тестирования на основе процедур вероятностного вывода.

2. Разработаны оригинальные модели ДБС фаззинг-тестирования OWASP-групп ошибок функционирования веб-приложений, отражающие временные и вероятностные связи между различными элементами тестирования.

3. Разработан метод обучения структуры ДБС на основе формирования марковского покрытия, учитывающий при обучении транзитные связи между соседними временными срезами сетей, использующий гибридные технологий

обучения, базирующиеся на статистических подходах оценки топологии байесовских сетей и применении оптимизационной процедуры имитация отжига для определения направленности связей между отдельными узлами сетей.

4. Разработан адаптированный к условиям неполных данных метод обучения вероятностных параметров ДБС, базирующийся на применении стохастического алгоритма ожидания максимизации и семплирования по методу Монте-Карло с использованием аппарата цепей Маркова.

5. Разработаны адаптированные к ДБС методы вероятностного вывода на основе многочастичного фильтра, базирующиеся на применении алгоритма Метрополиса-Гастингса на этапе повторной генерации выборок и теории достаточных статистик для сокращения количества выборок и оптимизации расчета весов генерируемых выборок, теоремы Лемана и Шеффе и технологии снижения сложности структуры динамических байесовских сетей на основе построения деревьев сочленений;

6. Разработан метод синхронизации большого объема данных, передаваемых в распределенной вычислительной системе в процессе решения задач обучения и вероятностного вывода для ДБС, базирующийся на методах Сузуки-Касами с применением инструментов вложенных цепей Маркова и статистической оптимизации для настройки параметров синхронизации.

7. Предложена структура параллельных вычислений в алгоритмах обучения и вероятностного вывода для ДБС с применением предложенных в исследовании инструментов повышения эффективности процедур обучения и вероятностного вывода.

8. Предложена структура вычислительного эксперимента, реализующего комплексный подход к организации на основе аппарата байесовских моделей процесса тестирования, выделенных по классификации OWASP и MITRE групп ошибок функционирования веб-приложений, направленный на оценку качества, временной и ресурсной эффективности разработанных в исследовании новых методов, моделей и гибридных алгоритмов обучения и вероятностного вывода.

Полученные результаты исследования соответствуют паспорту научной специальности «2.3.8. Информатика и информационные процессы».

Рекомендации по практическому использованию. Разработанные методы тестирования на основе моделей динамических байесовских сетей могут быть использованы для создания программных и аппаратных средств обнаружения ошибок в сетевых протоколах, системных и прикладных программах различного назначения, а также в качестве интеллектуальных элементов систем обнаружения вторжений и реагирования на инциденты в телекоммуникационных сетях.

Перспективы дальнейшего развития тематики диссертации связаны с оценкой возможности создания единого комплекса автоматизированного тестирования методом фаззинга на основе динамических байесовских сетей, с адаптацией предложенных методов обучения структуры, параметров и вероятностного вывода для ДБС с более сложной системой транзитивных связей между временными срезами сети, с проведением сравнительного анализа эффективности применения различных методов синхронизации данных, передаваемых в распределенной вычислительной системе в процессе решения задач обучения и вероятностного вывода для ДБС, с улучшением временных и ресурсных характеристик предложенных в работе алгоритмов, совершенствованием механизмов обнаружения аномальных ошибок за счет оптимизации процедуры вероятностного вывода и расширения числа функциональных подходов к тестированию и анализу программных ошибок в соответствии с требованиями OWASP и MITRE.

Список использованных источников

1. Anderson G. A Monotonicity Property of the Gamma Function / G. Anderson, S. Qiu // Proc. AMS, 1997. – Vol. 125(11). – pp. 3355-3362.

2. Andrieu C. Efficient particle filtering for jump Markov systems / C. Andrieu, M. Davy, A. Doucet // Proc. IEEE ICASSP, 2002. – No 2. – pp. 1625–1628.

3. Asmussen S. Applied probability and queue / S. Asmussen. – New York: Spinger, 2004. – 439 p.

4. Avriel M. Nonlinear programming: analysis and methods / M. Avriel. – N.J.: Prentice-Hall, 2003. – 847 p.

5. Bahadur R. R. Examples of inconsistency of maximum likelihood estimates / R.R. Bahadur // The Indian J. Of Statistic, 1957. – No 20. – pp. 207—210.

6. Barankin E.W. A note on functional minimality of sufficient statistics / E.W. Barankin // The Indian J. Of Statistic, 1961. – No 23. – pp. 401-404.

7. Barankin E.W. Localy best unbeased estimates / E.W. Barankin // Ann. Math. Statististic, 1949. – No 20 – pp. 477-501.

8. Barankin E.W. Suffitient statistics of minimal dimention / E.W. Barankin // The Indian J. Of Statistic, 1959. – No 21 – pp. 217-246.

9. Bernardo J.M. Bayesian Theory / J.M. Bernardo, A.F. Smith. – New York: Wiley, 2000. – 611 p.

10. Bhat U.N. An introduction to queueing theory / U.N. Bhat. – Boston: Birkhauser, 2015. – 353 p.

11. Bishop C. Pattern Recognition and Machine Learning / C. Bishop. – New York: Springer, 2006. – 738 p.

12. Bisseling R. Parallel Scientific Computation / R. Bisseling. – New York: Oxford University Press, 2004. – 324 p.

13. Bocharov P. P. Analysis of a two-phase queueing system with a Markov arrival process and blocking / P. P. Bocharov, R. Manzo, A. V. Pechinkin // Journal of Mathematical Sciences, 2006. – Vol. 132(5). – pp. 578–589.

14. Borgelt K. Graphical Models:Representations for Learning, Reasoning and Data Mining / K Borgelt, R Kruse, M Steinbrecher. – California: Wiley, 2009 – 387 p.

15. Boudreau J. F. Applied Computational Physics / J.F. Boudreau, E. S. Swanson. – Oxford: Oxford Univeristy Press, 2018. – 193 p.

16. Boyen X. Tractable inference for complex stochastic process / X. Boyen, D. Koller // Proc. of 14 th Conf. Uncertainty in AI, 1998. – pp. 33–42.

17. Brooks S. Handbook of Markov Chain Monte Carlo / Brooks S., A. Gelman, G. L. Jones, Xi Meng. – Boca Raton: CRC Press, 2011. – 592 p.

18. Chickering D.M. A Transformational Characterization of Equivalent Bayesian Network Structures / D.M. Chickering // Proc. UAI. – NY: Morgan Kaufman, 1995. – pp. 87-98.

19. Chowdhury M. Managing Data Transfers in Computer Clusters with Orchestra / M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, I. Stoica // SIGCOMM Computer. Communication. Review, 2011. – Vol 41(4). – pp. 98-109.

20. Clarke A.B. Maximum likelihood estimates in simple queues / A.B. Clarke // Annals of Math. Statistic, 1957. – Vol. 27. – pp. 1036-1040.

21. Cohen W.W. A Comparison of String Distance Metrics for Name-Matching Tasks / W.W. Cohen, P. Ravikumar, S.E. Fienberg // Proc. of the IJCAI, 2003. – pp. 73-78.

22. Cox D.R.. Principles of Statistical Inference / D.R. Cox. – New York: Cambridge University Press, 2006. – 281 p.

23. Damerau F.A. Technique for Computer Detection and Correction of Spelling Errors / F.A. Damerau // Communications of the ACM, 1964 – Vol. 7 – No. 3. – pp. 171–176.

24. Dean J. MapReduce: Simplified data processing on large clusters / J. Dean, S. Ghemawat // Communications Of The Acm, 2008. – Vol. 51(1). – pp.107-113.

25. Doucet A. Rao-Blackwellized Particle Filtering for Dynamic Bayesian Networks / A. Doucet, N. Freitas, K.P. Murphy, S. Russel // Proc. of 16th Conf. Uncertainty in AI, 2000 – pp. 176–183.

26. Doucet A. Sequental Monte Carlo Methods in Practice / A. Doucet, N. Freitas, N. Gordon. – New York: Springer, 2001. – 581 p.

27. Fraser D. The optimum linear smoother as a combination of two optimum linear filters / D. Fraser, J Potter // IEEE Transactions on Automatic Control, 1969. – No 14(4) – pp. 387-390.

28. Goodfellow I. Deep Learning / I. Googfellow, Y. Bengio, A. Courville. – Cambridge: MIT Press, 2016. – 785 p.

29. Grewal M.S. Kalman filtering: Theory and practice using mathlab / S. M. Grewal, A. P. Andrews. – New York: Wiley, 2001. – 401 p.

30. Halpern J. Reasoning about uncertainty / J. Halpern. – Cambridge:The MIT Press, 2003. – 483 p.

31. Heckerman D. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data / D. Heckerman, D. Geiger, D. M. Chickering // Machine Learning, 1995. – No 20 – pp 197-243.

32. Hill R. Guide to secure assurance for cloud computing / R. Hill, S.Y Zhu, M. Trovati. – L.: Springer, 2015. – 229 p.

33. Hyyro H. Practical Methods for Approximate String Matching: PhD Thesis / H. Hyyro. – Finland, 2003. – 96 p.

34. Jensen F. V. Bayesian Networks and Decision Graphs / F. V. Jensen. – Berlin :Springer-Verlag, 2001. – 268 p.

35. Jurasky D. Speech and Language Processing. Second Edition / D. Jurasky, J.H. Martin. – New Jersey: Prentice Hall, 2008. – 1032 p.

36. Kalman R.A. New Approach to Linear Filtering and Prediction Problems // J. of Basic Engineering, 1960. – No 82 (Series D). – pp 35-45.

37. Kasella G. Statistical Inference.Second Edition / G.Kasella, R.L.Berger. – Pacific Grove:Duxbery, 2002. – 660 p.

38. Kendall M. The Advanced Theory of Statistics – Vol. II / M. Kendall, A. Stuart. – London: Charles Griffin & Co., 1961. – 676 p.

39. Kitagawa G. Non-Gaussian state-space modeling of nonstationary time series / G. Kitagawa // J. American Statistical Association,1987. – No 82(400). – pp. 1032-1041.

40. Koch K.R. Introduction to bayesian statistics / K.R. Koch. – Berlin:Springer, 2007. – 250 p.

41. Koller D. Probabilistic graphical models. Principles and Techniques / D. Koller, N. Friedman. – Cambridge: MIT Press, 2009. – 1231 p.

42. Kolmogorov A.N. Unbiased estimates / A. N. Kolmogorov // Izv. Akad. Nauk

SSSR Ser. Mat., 1950. – Vol. 14 – No 4 – pp. 303–326.

43. Korb K.B. Bayesian Artificial Intelligence / K.B.Korb, A.E. Nicholson. – Boca Raton:CRC Press, 2004. – 491 p.

44. Kullbak S. Information and Sufficiency / S. Kullbak, R. Leibler // Ann. Math Statistics,1951. – Vol. 22. – No 1. – pp. 79–86.

45. Lamport L. Time, Clocks, and the Ordering of Events in a Distributed System / L. Lamport // Comunications of the ACM, 1978. – Vol. 21. – No 7. – pp. 558–565.

46. Lamport L. Using Time Instead of Timeout for Fault-Tolerant Distributed Systems / L. Lamport // ACMTransactionson Programming Languages and Systems, 1984. – Vol.6 – No. 2. – pp. 254-280.

47. Lehman E.L. Completeness, Similar Regions and Unbiased Estimations – Part I / E.L. Lehman, H. Sheffe // The Indian J. Of Statistic, 1950. – Vol. 10(4). – pp.233-268.

48. Lehman E.L. Completeness, Similar Regions and Unbiased Estimations – Part II / E.L. Lehman, H. Sheffe // The Indian J. Of Statistic, 1955. – Vol. 15(3). – pp. 269-286.

49. Leman E.L. Selected works by E.L. Leman / E.L. Leman. – New York:Springer, 2012. – 1109 p.

50. Lewis F.L. Optimal and robast estimation / F. L. Lewis, L. Xie, D. Popa. – Boca Raton: CRC Press, 2008. – 523 p.

51. Malewicz G. Pregel: a system for large-scale graph processing / G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, G. Czajkowski // Processing of ACM SIGMOD International Conference, 2010. – pp. 135-146.

52. Manning C.D. Introduction to Information Retrieval / C.D. Manning. – New York: Cambridge University Press, 2008. – 506 p.

53. Metropolis N. Equations of Stafe Colculations by fast computing machines / N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller // J. Chem. Phys., 1953. – Vol. 21. – pp. 1087-1092.

54. Moral P.D. Nonlinear Filtering: Interacting Particle Resolution / P.D. Moral // Markov Processing and Related Fields, 1996. – Vol. 2(4). – P. 555-580.

55. Moral P.D. On Adaptive Resampling Procedures for Sequential Monte Carlo Methods. P.D. Moral, A. Doucet, A. Jasra // Bernoulli, 2012. – Vol. 18(1). – P. 252-278.

56. Murphy K.P. Machine learning a probabilistic perspectives / K.P. Murphy. – Massachusetts: MIT Press, 2002. – 1067 p.

57. Nocedal J. Numerical optimization / J. Nocedal, S.J. Wright. – N.Y.: Springer, 1999. – 634 p.

58. Pearl J. Causality:Models, Reasoning and Inference / J. Pearl. – N.Y.: Cambridge University Press, 2009. – 484 p.

59. Pearl J. Evidential reasoning using stochastic simulation of causal models / J. Pearl // J. of Artificial Intelligence, 1987. – No 32. – pp. 247-257.

60. Pearl J. Fusion, propagation and structuring in belief networks / J. Pearl // Aritficial Intelligence, 1986. – Vol. 29. – pp. 241-288.

61. Pearl J. Probabilistic Reasoning in Intelligent Systems / J.Pearl. – NY: Morgan Kaufman, 1988. – 552 p.

62. Pinedo M.L. Scheduling: Theory, Algorithm and Systems. Third edition / M.L. Pinedo. – N.Y.:Springer, 2008. – 671 p.

63. Polukhin P.V. Advanced hybrid stochastic dynamic Bayesian network inference algorithm development in the context of the web applications test execution / P.V. Polukhin, T.V. Azarnova // Journal of Physics: Conference Series. – 2019. – Vol. 537. – Issue 5. – P. 052028.

64. Polukhin P.V. Application of junction tree clustering methods for solving dynamic Bayesian networks probabilistic inference tasks / P.V. Polukhin, T.V. Azarnova // Journal of Physics: Conference Series. – 2020. – Vol. 1479. – Issue 1. – P. 012096.

65. Polukhin P.V. Development of dynamic Bayesian models for web application test management / P.V. Polukhin, T.V. Azarnova, Yu.V. Bondarenko, I.L. Kashirina // Journal of Physics: Conference Series. – 2018. – Vol. 973. – Issue 1. – P. 012024.

66. Polukhin P.V. Distributed computing systems synchronization modeling for solving machine learning tasks / P.V. Polukhin, T.V. Azarnova // Journal of Physics: Conference Series. – 2021. – Vol. 1902. – Issue 1. – P. 012050.

67. Polukhin P.V. Fuzzing process modelling for Web Applications Robustness Testing via Dynamic Bayesian Networks / P.V. Polukhin, T.V. Azarnova, S.A. Barkalov // Proceedings GloSIC-2018. – 2018. – pp.1-6.

68. Polukhin P.V. Markov stochastic processes application tools to improve the characteristics of distributed systems synchronization procedures / P.V. Poluhnin, T.V. Azarnova, I.L. Kashirina // Proceedings SUMMA-2023. – 2023. – pp.1-6.

69. Polukhin P.V. The Hybrid parameter learning algorithm development for dynamic Bayesian network in the context of the Metropolis-Hastings approach tasks / P.V. Polukhin, T.V. Azarnova // Journal of Physics: Conference Series. – 2019. – Vol. 1203. – Issue 1. – P. 012084.

70. Quenouille M.H. Notes on bias on estimation / M.H. Quenouille // Biometrika, 1956. – No 43. – pp. 353-360.

71. Rauch H.E. Maximum likelihood of linear dynamic systems / H.E. Rauch, F. Tung, C.T. Striebel // J. of Amer. Inst. Aeronautic and Astronautics, 1965. – No 3(8). – pp. 1445-1450.

72. Reif J.H. Synthesis of Parallel Algorithms / J.H. Reif. – San Mateo: Morgan Kaufman, 1993. – 1011 p.

73. Robinson R.W. Counting unlabeled acyclic digraphs / R.W. Robinson // Lect. Notes Math, 1977. – No 622. – pp 28-43.

74. Ross S. M. Stochastic Processes. Second edition / S. M. Ross. – New York: Wiley, 1996. – 510 p.

75. Salton G. Introduction to Modern Information Retrieval / G. Salton, M.J McGill. – New York: McGraw-Hill,1983 – 448 p.

76. Sarkka S. Bayesian Filtering and Smoothing / S. Sarkka. – Cambridge: Cambridge University Press, 2013. – 256 p.

77. Shenoy PP. Binary join trees for computing marginals in the shefer-shenoy arhitecture / PP Shenoy // J. of Approx. Reasoning, 1997. – No 17 (2-3). – pp. 239-263.

78. Shenoy PP. Propagation Belief Functions with Local Computations / PP Shenoy, G. Shafer // IEEE Expert, 1986. – No 1 (3). – pp. 43-52.

79. Sherman J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix / J. Sherman, Morrison W.J. // Ann. Math. Statistics, 1950. – Vol. 21. – No 1. – p. 124-127.

80. Sherman J. Computations relating to inverse matrices / J. Scherman //

Simultaneous Linear Equations and the Determination of Eigenvalues. U.S. Nat. Bur. Standards. Appl. Math. Ser., 1953. – No 29. – pp. 123-124.

81. Spirtes P. Causation, Prediction and Search / P.Spirtes, C. Glymour, R. Sheines. – Cambridge: MIT Press, 2000. – 568 p.

82. Suzuki I. distributed mutual exclusion algorithm / I. Suzuki, T.A. Kasami // ACM Transactions on Computer Systems, 1985. – Vol. 3(4). – pp. 344–349.

83. Suzuki J. A. Theoretical Analysis of the BDeu Scores in Bayesian Network Structure Learning / J. Suzuki // Behaviormetrika, 1995. – Vol. 1(1). – pp. 1-20.

84. Takanen A. Fuzzing for Software Security Testing and Quality Assurance / A. Takanen, G. DeMott, C. Miller. – US: Artech House, 2008. – 312 p.

85. Tsamardinos I. The max-min hill-climbing Bayesian network structure learning algorithm / I. Tsamardinos, L.E. Brown, C.F. Aliferis // Machine Learning, 2006. – No 65. – pp. 31–78.

86. Ukkonen E. Algorithms for Approximate String Matching / E. Ukkonen // Information and Control, 1985 – No 64. – pp. 100–118.

87. Ukkonen E. Finding Approximate Patterns in Strings // Journal of Algorithms, 1985. – No. 6. – pp. 132–137.

88. Valiant L.G. A Bridging Model for Parallel Computation / L.G. Valiant // Comm. of ACM, 1990. – Vol 33. – No 8. – pp. 135-145.

89. Wagner R.A. The string-to-string correction problem / R.A. Wagner, M.J. Fischer // Journal of the Association for Computing Machinery, 1974. – Vol. 21. – No. 1. – pp. 168-176.

90. Wolfe P. Convergence conditions for ascent methods / P.Wolfe // Siam Review, 1969. – Vol. 11. – No. 2. – pp. 226-235.

91. Wolfe P. Convergence conditions for ascent methods. II: Some corrections / P.Wolfe // SIAM review, 1971. – Vol. 13. – No 2. – pp. 185-188.

92. Zaharia M. An Architecture for Fast and General Data Processing on Large Clusters: Dissertation doctor of philosophy in computer science / University of California, Berkeley, 2013. – 113 p.

93. Zaharia M. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-

Memory Cluster Computing / M. Zaharia, M. Chowdhury, T. Das, A. Dave, M. McCauley, M. Franklin, S. Shenker, I. Stoica // NSDI, 2012. – pp 1-15.

94. Zegzhda P.D. Detection of Anomalies in Behavior of the Software with Usage of Markov Chains / P. D. Zegzhda, S. S. Kort, A. F. Suprun // Automatic Control and Computer Sciences, 2015. – Vol. 49. – No 8. – pp. 820-825.

95. Аветисян А.И. Анализ характера изменений программ и поиск неисправленных фрагментов кода / М.С. Арутюнян, Г.С. Иванов, В.Г. Варданян, А.К. Асланян, Арутюнян М.С, Курмангалеев Ш.Ф // .Труды ИСП РАН, 2009. – Т 31 – №1. – С. 49-58.

96. Аветисян А.И. Технология статического и динамического анализа уязвимостей программного обуспечения / А.И. Аветисян, А.А. Белеванцев, И.И. Чукляев // Вопросы кибербезопасноти, 2014. – №3(4). – С. 20-28.

97. Айвазян С.А. Прикладная статистика: Классификация и снижение размерности / С.А. Айвазян, В.М. Бухштабер, И.С. Енюков, Л.Д. Мешанкин. – М.: Финансы и статистика, 1989. – 607 с.

98. Арустамов С.А. Применение динамической байесовской сети в системах обнаружения вторжений / С.А. Арустамов, В.Ю. Дайнеко // Научно-технический вестник информационных технологий, механики и оптики, 2012. – №3. – С. 128-133.

99. Афанасьева Л.Г. Случайные процессы в теории массового обслуживания и управления запасами / Л.Г. Афанасьева, Е.В. Булинская. – М.:Издательство МГУ, 1980. –110 с.

100. Балакришнан А.В. Теория фильтрации Калмана: Пер. с англ. / А.В. Балакришнан. – М.:Мир, 1988. – 168 с.

101. Барский А.Б. Параллелльные процессы в вычислительных системах: Планирование и организация / А.Б. Барский. – М: Радио и связь, 1990. – 256 с.

102. Бачура-Рид А.Т. Элементы теории марковских процессов и их приложение: Пер. с англ. / А.Т. Бачура-Рид. – М: Наука, 1969. – 512 с

103. Бейзер Б. Тестирование черного ящика / Б Бейзер. – Спб.: Питер, 2004. – 321 с.

104. Беляев Ю.К. Основы математической статистики. Ч.1 / Ю.К. Беляев, Е.В.

Чепурин. – М.: Изд-во Моск. ун-та, 1982. – 100с.

105. Бернс Б. Распределенные системы. Паттерны проектирования: Пер. с англ. / Б. Бернс. – СПб.: Питер, 2019.– 224 с.

106. Блюмин С.Л. Окрестностное моделирование сетей Петри: монография / С.Л. Блюмин, И.А. Седых, В.Ю. Филоненко. – Липецк: ЛЭГИ, 2010. – 124 с.

107. Борисов В.В. Нечеткие модели и сети / В.В. Борисов, В.В. Круглов, А.С. Федулов. – М.: Горячая линия-Телеком, 2012. – 284 с.

108. Боровков А.А. Вероятностные процессы в теории массового обслуживания / А.А. Боровков. – М.: Мир, 1972. – 368 с.

109. Боровков А.А. Математическая статистика / А.А. Боровков. – СПб.: Издательство «Лань», 2010. – 704 с.

110. Браницкий А.А. Обнаружение сетевых атак на основе комлексирования нейронных, имунных и нейронечетких классификаторов / А.А. Браницкий, И.В. Котенко // Информационно-управляющие системы, 2005. – №4. – с. 69-77.

111. Булдакова Т.И. Выбор технологий Data Mining для систем обнаружения вторжений в корпоративную сеть / Т.И. Булдакова, А.Ш. Джалалов // Инженерный журнал:наука и инновации, 2013 – №11. – с. 1-14.

112. Булинский А.В. Теория случайных процессов / А.В. Булинский, А.Н. Ширяев. – М.:Физматлит, 2005. – 400 с.

113. Вентцель Е. С. Прикладные задачи теории вероятностей / Е. С. Вентцель, А. А. Овчаров. – М. : Радио и связь, 1983. – 416 с.

114. Вержбицкий В.М. Численные методы (линейная алгебра и нелинейные уравнения) / В.М. Вержбицкий. – М.: Оникс 21 век, 2005. – 432 с.

115. Винклер Г. Анализ изображений, случайные поля и методы Монте-Карло на цепях Маркова. Математические основы: Пер. с англ. / Г. Виклер – Н.: ГЕО, 2008. – 440 с.

116. Воеводин В.В. Математические основы параллельных вычислений / В.В. Воеводин. – М.: Изд. Моск. ун-та,1991. – 345 с.

117. Воеводин В.В. Модели и методы в параллельных процессах / В.В. Воеводин. – М.: Наука, 1986. – 296 с.

118. Воеводин В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб.:БХВ-Петербург, 2002. – 608 с.

119. Воинов В.Г. Несмещенные оценки и их применение / В.Г. Воинов, М.С. Никулин. – М.:Наука, 1989. – 440 с.

120. Гергель В. Высокопроизводительные вычисления для многоядерных систем / В. Гергелью. – М.: Издательство Московского университета, 2010. – 544 с.

121. Гилл Ф. Практическая оптимизация: Пер. с англ. / Ф. Гилл, У. Мюррей, М. Райт. – М.: Мир, 1985. – 509 с.

122. Гнеденко Б.В. Введение в теорию массового обслуживания. / Б.В. Гнеденко, И.Н. Коваленко. – М.:Наука, 1966. – 431 с.

123. Голуб Дж. Матричные вычисления:Пер. с англ. / Дж. Голуб, Ч. Ван Лоун. – М.:Мир, 1999. – 549 с.

124. Городецкий В.И. Формирование непротиворечивых баз знаний с неопределенностью / В.И. Городецкий, А.Л. Тулупьев // Изв. РАН. Сер. Теория и системы управления. – 1997. – Т. 36. – № 5. – с. 33–42.

125. ГОСТ 17799-2005 Практические правила управления информационной безопасностью. – М.: Издательство стандартов, 2008. – 54 с.

126. ГОСТ 27.002-89 Надежность в технике. Основные понятия. Термины и определения. – М.: Издательство стандартов, 1990. – 37 с.

127. ГОСТ 28195-89 Оценка качества программных средств. – М.: Изд-во стандартов, 1989. – 38 с.

128. ГОСТ Р 34.321-96. Информационные технологии. Системы стандартов по базам данных. Эталонная модель управления данными.

129. ГОСТ Р ИСО 28640-2012. Статистические методы. Генерация случайных чисел. М: Стандартинформ, 2014. – 52 с.

130. Деменков Н.П. Статистическая динамика систем управления / Н. П. Деменков. – М.: Издательство МГТУ им. Баумана, 2017. – 146 с.

131. Дынкин Е.В. Необходимые и достаточные статистики для семейства распределений вероятностей / Е. В. Дынкин // Успехи матем. наук, 1960. – №6(1). – с. 68—90.

132. Дэнис Дж. Численные методы безусловной оптимизации и решения нелинейных уравнений: Пер. с англ. / Дж. Дэнис, Р. Шнабель. – М.: Мир, 1988. – 440 с.

133. Журавлев О.Г. Минимальные достаточные статистики для последовательности независимых случайных величин / О.Г. Журавлев // Теория вероят. и ее примен., 1963. –№8(2). – с. 236—238.

134. Закс Ш. Теория статистических выводов: Пер. с англ. / Ш. Закс. – М.:Мир, 1975. – 776 с.

135. Зиновьева Л.И., Терпугов А.Ф. Однолинейная система массового обслуживания с переменной интенсивностью, зависящей от времени ожидания / Л.И. Зиновьева, А.Ф. Терпугов //Автоматика и телемеханика, 1981. – №1. – с 27-30.

136. Иванов Н.Н. Обобщенные временные стохастические сети Петри / Н.Н. Иванов // Автомат. и телемех., 1996. – №10. – с. 159-164.

137. Ивченко Г.И. Введение в математическую статистику / Г.И. Ивченко, Ю.И. Медведев. – М.: Издательство ЛКИ, 2010. – 600 с.

138. Ивченок Г.И. Теория массового обслуживания / Г.И. Ивченко, В.А. Каштанов, И.Н. Коваленко. – М.: URSS, 2022. – 304 с.

139. Карлин С. Основы теории случайных процессов: Пер. с англ. / С. Карлин. – М.: Мир, 1971. – 536 с.

140. Кельберт М.Я. Вероятность и статистика в примерах и задачах. Т. II: Марковские цепи как отправная точка теории случайных процессов и их приложения: Пер. с англ. / М. Я Кельберт. – М.: МЦНМО, 2009.– 295 с.

141. Кендалл Д. Стохастические процессы, встречающиеся в теории очередей, и их анализ методом вложенных цепей Маркова / Д. Кендалл // Математика, 1959. – Т. 3. – №6. – с. 92-112.

142. Кендалл М. Статистические выводы и связи: Пер. с англ. / М. Кендалл, А. Стюарт. – М.: Наука, 1973. – 878 с.

143. Клейнрок Л. Теория массового обслуживания: Пер. с англ. / Л. Клейнрок. – М.: Машиностроение, 1979. – 432 с.

144. Климов Г.П. Теория массового обслуживания / Г. П. Климов. – М.:

Издательство Московского университета, 2011. – 312 с.

145. Кобзарь А.И. Прикладная математическая статистика / А.И.Кобзарь. – М.:Физматлит, 2006. – 816 с.

146. Кокс Д.Р. Теория очередей: Пер. с англ. / Д.Р. Кокс, У.И. Смит. – М.:Мир, 1966. – 218 с.

147. Колмогоров А.Н. Теория вероятностей и математическая статистика / А.Н. Колмогоров. – М.:Наука, 2005. – 581 с.

148. Колмогоров А.Н. Теория информации и теория алгоритмов / А.Н. Колмогоров. – М.:Наука, 2005. – 264 с.

149. Корнеев В.В. Параллельные вычислительные системы / В.В. Корнеев. – М: Нолидж, 1999. – 311 с.

150. Котенко И.В. Механизмы защиты компьютерных сетей от инфраструктурных атак на основе бионспирированного подхода «нервная система сети» / И.В. Котенко, А.В. Шоров // Техническая защита информации, 2013. – №2. – с. 57-66.

151. Котенко И.В. Применение графов атак для оценки защищенности компьютерных сетей и анализа событий безопасности / И.В. Котенко, А.А. Чечулин // Системы высокой доступности, 2013. - №3. – С. 103-110.

152. Крамер Г. Математические методы статистики: Пер. с англ. / Г.Крамер. – М.:Мир, 1975. – 648 с.

153. Левенштейн В.И. Двоичные коды с исправлением выпадение, вставок и замещений символов / В.И. Левенштейн // Доклады академии наук СССР, 1965. – Т. 163. – № 4. – с 845-848.

154. Леман Э. Проверка статистических гипотез: Пер. с англ. / Э.Леман. – М.:Наука, 1987. – 408 с.

155. Леман Э. Теория точечного оценивания: Математические методы статистики / Э.Леман. – М.:Наука, 1991. – 448 с.

156. Лившиц А.Л. Статистическое моделирование систем массового обслуживания / А.Л. Лившиц, Э.А. Мальц. – М.:Сов. Радио, 1978. – 248 с.

157. Липаев, В.В. Надежность программного обеспечения / В.В. Липаев – М.:

Радио и связь, 1998. – 200 с.

158. Льюг Л. Идентификация систем. Теория для пользователя: Пер. с англ. / Л. Льюг. – М.:Наука, 1991. – 432 с.

159. Магнус Я.Р. Матричное дифференциальное исчисление с приложениями к статистике и эконометрике: Пер. с англ. / Я.Р. Магнус, Х. Нейдеккер. – М.: Физматлит, 2002. – 496 с.

160. Миллер Б.М. Теория случайных процессов в примерах и задачах / Б.М. Миллер, А.Р. Панков. – М.: Физматлит, 2002. – 320 с.

161. Монахов О. Г. Параллельные системы с распределённой памятью: управление ресурсами и заданиями / О.Г. Монахов, Э.А. Монахова – Новосибирск : Издательство ИВМиМГ СО РАН, 2001. – 168 с.

162. Осовский С. Нейронные сети для обработки информации: Пер. с польского. / С Осовский. – М.: Горячая линия-Телеком, 2017. – 344 с.

163. Полухин П.В. Анализ эквивалентности байесовских сетей на основе асимптотических оценок Байеса-Дирихле / П.В. Полухин // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2022. – № 2. – С. 135-144.

164. Полухин П.В. Вариационные алгоритмы обучения и опроса байесовских сетей в условиях частичной наблюдаемости параметров / П.В. Полухин // Вестник кибернетики. – 2022. – № 2. – С. 74-84.

165. Полухин П.В. Динамические байесовские сети как инструмент тестирования веб-приложений методом фаззинга / П.В. Полухин, Т.В. Азарнова // Журнал вычислительной математики и математической физики. – 2021. – Т. 61. № 7. – С. 1125-1136.

166. Полухин П.В. Инструменты оптимизации многочастичного фильтра для вероятностных моделей динамических систем / П.В. Полухин // Системы управления и информационные технологии. – 2021. – № 4 (86). – С. 4-10.

167. Полухин П.В. Инструменты повышения эффективности численных алгоритмов обучения структуры динамических байесовских сетей / П.В. Полухин // Вестник Воронежского государственного университета. Серия: Системный

анализ и информационные технологии. – 2019. – № 4. – С. 132-140.
168. Полухин П.В. Исследование процесса фаззинга SQL-инъекций веб-приложений на основе динамической сети Байеса / П.В. Полухин, Т.В. Азарнова // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2014. – № 1. – С. 120-129.
169. Полухин П.В. Оптимизация алгоритмов вероятностного вывода в динамических вероятностных моделях на основе применения моделей динамики Гамильтона / П.В. Полухин // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2023. – № 2. – С. 135-144.
170. Полухин П.В. Оптимизация вычислительных процедур стохастических алгоритмов фильтрации и сглаживания, построенных на основе фильтра Калмана / П.В. Полухин // Вестник Российского нового университета. Серия: Сложные системы: модели, анализ и управление. – 2022. – № 1. – С. 3-15.
171. Полухин П.В. Оценка временной и ресурсной эффективности применения моделей динамических байесовских сетей для организации процедуры тестирования web-приложений методом фаззинга / П.В. Полухин // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2023. – № 4. – С. 141-151.
172. Полухин П.В. Применение байесовских методов для оптимизации обучения нейросетевых моделей тестирования приложений / П.В. Полухин // Вестник кибернетики. – 2022. – № 3. – С. 46-56.
173. Полухин П.В. Применение генетических алгоритмов для оптимизации решения задач фильтрации и прогнозирования в динамических системах тестирования веб-приложений / П.В. Полухин // Вестник Югорского государственного университета. – 2022. – № 4. – С. 120-132.
174. Полухин П.В. Применение динамических байесовских сетей для повышения эффективности процесса фаззинга SQL-инъекций веб-приложений / П.В. Полухин, Т.В. Азарнова // Системы управления и информационные технологии. – 2014. – № 1.1(55). – С. 106-112.

175. Полухин П.В. Применение методов теории массового обслуживания для оценки параметров синхронизации распределенных вычислительных систем / П.В. Полухин // Моделирование, оптимизация и информационные технологии. – 2022. – № 2. – С. 1-10.

176. Полухин П.В. Применение методов теории массового обслуживания для оценки параметров синхронизации распределенных вычислительных систем / П.В. Полухин // Моделирование, оптимизация и информационные технологии. – 2022. – № 2. – С. 1-11.

177. Полухин П.В. Разработка алгоритмов решения задач стохастического вывода в полумарковских моделях динамических байесовских сетей / П.В. Полухин // Инженерный вестник Дона. – 2022. – № 11. – С. 120-132.

178. Полухин П.В. Разработка вероятностных подходов к оптимизации процедуры обучения параметров моделей тестирования на основе метода ожидания-максимизации / П.В. Полухин // Вестник Югорского государственного университета. – 2022. – № 2. – С. 95-103.

179. Полухин П.В. Разработка гибридного алгоритма обучения структуры динамической байесовской сети на основе метода Левенберга – Марквардта / Т.В. Азарнова, П.В. Полухин, С.А. Баркалов // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника, 2018. – Т. 18. № 4. – С. 16-24.

180. Полухин П.В. Разработка концепции и инструментария моделирования процессов тестирования веб-приложений методом фаззинга с помощью динамических байесовских сетей / П.В. Полухин, Т.В. Азарнова // Моделирование, оптимизация и информационные технологии. – 2023. – № 4. – С. 1-11.

181. Полухин П.В. Разработка параллельных алгоритмов обучения вероятностных моделей тестирования веб-приложений / П.В. Полухин // Интеллектуальные системы в производстве. – 2022. – № 3. – С. 94-103.

182. Полухин П.В. Управление процессом тестирования веб-приложений методом фаззинга на основе динамических байесовских сетей / Т.В. Азарнова, П.В. Полухин, С.А. Баркалов // Вестник Южно-Уральского государственного

университета. Серия: Компьютерные технологии, управление, радиоэлектроника, 2017. – Т. 17. № 2. – С. 51-64.

183. Полухин П.В. Формирование оценочных функций для решения задачи построения направленной динамической байесовской сети / П.В. Полухин // Вестник Российского нового университета. Серия: Сложные системы: модели, анализ и управление. – 2023. – № 4. – С. 1-8.

184. Полухин П.В. Формирование структуры байесовской сети процесса тестирования надежности информационных систем / П.В. Полухин, Т.В. Азарнова, Н.Г. Аснина, Д.К. Проскурин // Вестник Воронежского государственного технического университета, 2017. – Т. 13. № 6. – С. 45-51.

185. Полухин П.В., Азарнова Т.В. Гибридные механизмы обучения и вероятностного вывода для статических и динамических байесовских сетей. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2021. № 2021666156 от 08.10.2021.

186. Полухин П.В., Азарнова Т.В. Комплексная методика тестирования уязвимостей нарушения конфигурации системы. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016615100 от 16.05.2016.

187. Полухин П.В., Азарнова Т.В. Комплексный подход к анализу уязвимостей небезопасных перенаправлений веб-приложений. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016616241 от 20.07.2016.

188. Полухин П.В., Азарнова Т.В. Математические модели, методы и алгоритмы тестирования раскрытия персональных данных с помощью фаззинга. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016616336 от 09.06.2016.

189. Полухин П.В., Азарнова Т.В. Методология тестирования инъекции веб-приложений с помощью фаззинга. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016615101 от 16.05.2016.

190. Полухин П.В., Азарнова Т.В. Механизмы анализа защищенности систем аутентификации и управления сессиями веб-приложений на основе фаззинга. –

Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016612407 от 21.03.2016.

191. Полухин П.В., Азарнова Т.В. Механизмы повышения эффективности и качества тестирования уязвимостей межсайтовой подделки запросов (CSFR) с помощью фаззинга. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016616243 от 20.07.2016.

192. Полухин П.В., Азарнова Т.В. Обнаружение и анализ уязвимостей межсайтового скриптинга (XSS) современных интернет-приложений. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016614976 от 20.06.2016.

193. Полухин П.В., Азарнова Т.В. Особенности выявления использования компонентов с уязвимостями на основе технологии фаззинга. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016616720 от 20.07.2016.

194. Полухин П.В., Азарнова Т.В. Оценка эффективности тестирования уязвимостей, связанных с нарушением управления доступа интернет-приложений. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 20166119513 от 20.09.2016.

195. Полухин П.В., Азарнова Т.В. Программа повышения качества оценки защищенности веб-приложений от небезопасных ссылок на объекты с помощью фаззинга. – Свидетельство о государственной регистрации программы для ЭВМ. М.: ФИПС, 2016. № 2016612433 от 20.07.2016.

196. Рао С.Р. Линейные статистические методы и их применение: Пер. с англ. / С. Р. Рао – М.: Наука,1968. – 548 с.

197. Рассел С. Искусственный интеллект: современный подход, 2 издание: Пер. с англ. / С. Рассел, П. Норвиг. – М.: Вильямс, 2006. – 1408 с.

198. Риордан Дж. Вероятностные системы обслуживания: Пер. с англ. / Дж. Риордан. – М.: Связь, 1966. – 184 с.

199. Самарский А.А. Математическое моделирование: Идеи. Методы. Примеры / А.А. Самарский. – М.: Физматлит, 2001. – 320 с.

200. Саттон М. Fuzzing: исследование уязвимостей методом грубой силы: Пер. с англ. / М. Саттон, А. Грин, П. Амини. – М.:Вильямс, 2009. – 560 с.

201. Синицин И.Н. Фильтр Калмана и Пугачева / И.Н. Синицин. – М:Логос, 2006. – 640 с.

202. Сироткин А.В. Байесовские сети доверия: дерево сочленений и его вероятностная семантика / А.В. Сироткин // Труды СПИИРАН , 2006. – № 3. – с. 228-239.

203. Таненбаум Э. Распределенные системы. Принципы и парадигмы/ Э. Таненбаум, М. Ванн Сеен. – Спб.: Питер, 2003. – 877 с.

204. Тарасюк И.В. Стохастические сети Петри – формализм для моделирования и анализа производительности вычислительных процессов / И.В. Тарасюк // Системная информатика , 2004. – №9. – с 135-194.

205. Тарков М.С. Вложение структур параллельных программ в структуры живучих распределенных вычислительных систем / М.С Тарков // Автометрия, 2003. – №3. – с.84-96.

206. Тихонов В.И. Марковские процессы / В.И. Тихонов, М.А. Миронов. – М.: Сов. Радио, 1977. – 488 с.

207. Топорков В.В. Модели распределенных вычислений / В.В. Топорков. – М.: Физматлит, 2004. – 320 с.

208. Тулупьев А.Л. Апостериорные оценки вероятностей в алгебраических байесовских сетях / А.Л. Тулупьев // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. – 2012. – № 2. – с. 51–59.

209. Тулупьев А.Л. Дерево смежности с идеалами конъюнктов как ациклическая алгебраическая байесовская сеть / А.Л. Тулупьев // Тр. СПИИРАН. – 2006. – Т.1. – №3. – с. 198-227.

210. Турчак Л.И. Основы численных методов / Л.И. Турчак, П.В. Плотников. – М.: Физматлит, 2003. – 304 с.

211. Уолрэнд Дж. Введение в теорию сетей массового обслуживания: Пер. с англ. / Дж. Уолрэнд. – М.:Мир, 1993. – 336 с.

212. Фильченко А.А. Алгоритм построения множества минимальных графов смежности при помощи клик владений / А.А. Фильченко // Труды СПИИРАН, 2010. – Вып. 1 (13). – с. 119-133.

213. Фильченко А.А. Анализ циклов в минимальных графах смежности алгебраических байесовских сетей / А.А. Фильченко, А.Л. Тулупьев // Труди СПИИРАН, 2011. – Вып. 2 (17). – с. 151-173.

214. Фильченко А.А. Связность и ацикличность первичной структуры алгебраической байесовской сети / А.А. Фильченко, А.Л. Тулупьев // Вестник Санкт-Петербургского университета. Серия 1: Математика. Механика. Астрономия. – 2013. – № 1. – с. 110–119.

215. Фишер Р.А. Статистические методы для исследователей / Р.А.Фишер. – М.:Гостатизд, 1958. – 267 с.

216. Хинчин А.Я. Математические методы теории массового обслуживания / А.Я Хинчин // Труды МИАН СССР, 1955. – Т. 53. – № 3. – с. 3-129.

217. Хорошевский В.Г. Архитектура вычислительных систем / В.Г. Хорошевский. – М. : Издательство МГТУ им. Н. Э. Баумана, 2008. – 520 с.

218. Шахтарин Б.И. Фильтр Винера и Калмана / Б.И. Шахтарин. – М: Горячая линия-Телеком, 2014. – 396 с.

Приложение А. Таблицы априорных вероятностей динамических байесовский сетей процесса фаззинга

Таблица 1

Начальное распределение $P(X_0)$ ДБС фаззинга инъекций веб-приложений

Узел	Родительские вершины	Таблица условных вероятностей
Inject Type	-	[{0:0.8215297450424929},{1:0.08404154863078375},{2:0.09442870632672333}]
Http Parameter Pollution (t)	Inject Type (t)	[{0:0.3103448275862069},{10:0.08045977011494253},{13:0.10344827586206896},{15:0.12643678160919541},{17:0.034482758620689655},{20:0.034482758620689655},{22:0.034482758620689655},{23:0.034482758620689655},{25:0.034482758620689655},{32:0.08045977011494253},{36:0.09195402298850575},{37:0.034482758620689655},{40:0.550561797752809},{48:0.2247191011235955},{50:0.2247191011235955},{80:0.7},{90:0.3},]
Encoder (t)	Inject Type (t)	[{0:0.9080459770114943},{12:0.034482758620689655},{17:0.034482758620689655},{18:0.022988505747126436},{23:0.7752808988764045},{43:0.2247191011235955},{46:1.0},]
Union Injection (t)	Inject Type (t), Encoder (t), Http Parameter Pollution (t)	[{0:0.8421052631578947},{1:0.15789473684210525},{20:0.5714285714285714},{21:0.42857142857142855},{27:1.0},{30:0.45454545454545453},{31:0.5454545454545454},{35:1.0},{41:1.0},{45:1.0},{47:1.0},{51:1.0},{64:0.5714285714285714},{65:0.42857142857142855},{72:0.625},{73:0.375},{75:1.0},{961:1.0},{1361:1.0},{1441:1.0},{1840:1.0},{1856:1.0},{1860:1.0},{3440:1.0},{3680:1.0},{3700:1.0},]
Boolean Based Blind (t)	Inject Type (t), Encoder (t), Http Parameter Pollution (t)	[{0:0.21052631578947367},{1:0.42105263157894735},{3:0.3684210526315789},{131:0.5714285714285714},{133:0.42857142857142855},{170:1.0},{196:0.7272727272727273},{198:0.2727272727272727},{222:1.0},{261:1.0},{287:1.0},{300:1.0},{326:1.0},{417:0.5714285714285714},{419:0.42857142857142855},{469:0.5},{471:0.5},{482:1.0},{6241:1.0},{8841:1.0},{9361:1.0},{11960:1.0},{12064:1.0},{12090:1.0},{22360:1.0},{23920:1.0},{24050:1.0},]
Time Blind (t)	Inject Type (t), Encoder (t),	[{0:0.21052631578947367},{3:0.7894736842105263},{133:1.0},{172:1.0},{198:1.0},{224:1.0},{263:1.0},{289:1.0},{302:1.0},{328:1.0},{419:1.0},{471

	Http Parameter Pollution (t)	:1.0},{484:1.0},{6243:1.0},{8843:1.0},{9363:1.0},{11960:1.0},{12064:1.0},{12090:1.0},{22360:1.0},{23920:1.0},{24050:1.0},]
Error Blind (t)	Inject Type (t), Encoder (t), Http Parameter Pollution (t)	[{0:0.21052631578947367},{7:0.2631578947368421},{8:0.3684210526315789},{10:0.1578947368421052 5},{187:0.14285714285714285},{188:0.42857142857142855},{190:0.42857142857142855},{244:1.0},{277:0.09090909090909091},{278:0.3636363636363636 5},{280:0.5454545454545454},{316:1.0},{370:1.0},{406:1.0},{424:1.0},{460:1.0},{583:0.14285714285714285},{584:0.4285714285714285 5},{586:0.42857142857142855},{655:0.125},{656:0.5},{658:0.375},{676:1.0},{8650:1.0},{12250:1.0},{12970:1.0},{16560:1.0},{16704:1.0},{16740:1.0},{30960:1.0},{33120:1.0},{33300:1.0},]
Stacked Time (t)	Inject Type (t), Encoder (t), Http Parameter Pollution (t)	[{0:1.0},{120:1.0},{156:1.0},{180:1.0},{204:1.0},{240:1.0},{264:1.0},{276:1.0},{300:1.0},{384:1.0},{432:1.0},{444:1.0},{5760:1.0},{8160:1.0},{8640:1.0},{11040:1.0},{11136:1.0},{11160:1.0},{20640:1.0},{22080:1.0},{22200:1.0},]
Out Of Band (t)	Inject Type (t), Encoder (t), Http Parameter Pollution (t)	[{0:0.21052631578947367},{6:0.7894736842105263},{116:1.0},{149:1.0},{171:1.0},{193:1.0},{226:1.0},{248:1.0},{259:1.0},{281:1.0},{358:1.0},{402:1.0},{413:1.0},{5286:1.0},{7486:1.0},{7926:1.0},{10120:1.0},{10208:1.0},{10230:1.0},{18920:1.0},{20240:1.0},{20350:1.0},]
Code (t)	Inject Type (t), Encoder (t), Http Parameter Pollution (t)	[{0:1.0},{30:1.0},{39:1.0},{45:1.0},{51:1.0},{60:1.0},{66:1.0},{69:1.0},{75:1.0},{96:1.0},{108:1.0},{111:1.0},{1440:1.0},{2040:1.0},{2160:1.0},{2760:1.0},{2784:1.0},{2790:1.0},{5160:1.0},{5520:0.5714285714285714},{5522:0.42857142857142855},{5552:1.0},]
Command (t)	Inject Type (t), Encoder (t), Http Parameter Pollution (t)	[{0:1.0},{20:1.0},{26:1.0},{30:1.0},{34:1.0},{40:1.0},{44:1.0},{46:1.0},{50:1.0},{64:1.0},{72:1.0},{74:1.0},{960:1.0},{1360:1.0},{1440:1.0},{1840:0.3448275862068966},{1841:0.6551724137931034},{1857:1.0},{1861:1.0},{3441:1.0},{3680:1.0},{3700:1.0},]
Network (t)	Dbms Fingerprint (t), Code (t), Command (t)	[{120:1.0},{264:1.0},{269:1.0},{282:1.0},]
Dbms Fingerprint (t)	Union Injection (t), Boolean Based Blind (t), Error Blind (t), Time Blind (t),	[{11:1.0},{540221:1.0},{1260941:1.0},{1281533:1.0},{5399933:1.0},]

	Stacked Time (t), Out Of Band (t)	
Network (t)	Dbms Fingerprint (t), Code (t), Command (t)	[{120:1.0},{264:1.0},{269:1.0},{282:1.0},]
Cmd Execution (t)	Dbms Fingerprint (t), Code (t), Command (t)	[{120:1.0},{264:1.0},{269:1.0},{282:1.0},]
File System (t)	Dbms Fingerprint (t), Code (t), Command (t)	[{123:1.0},{264:1.0},{269:1.0},{282:1.0},]
DbStructure (t)	Dbms Fingerprint (t)	[{10:1.0},{23:1.0}]
TableData (t)	DbStructure (t)	[{0:1.0},{3:1.0}]
Confidentiality (t)	DbStructure (t), Network (t), Cmd Execution (t), File System (t)	[{6:1.0},{129:1.0},{170:1.0},{212:1.0},]
Authentication (t)	File System (t), TableData (t), Cmd Execution (t)	[{9:1.0},{27:1.0},{45:1.0},{48:1.0},]
Authorization (t)	File System (t), TableData (t), Cmd Execution (t)	[{9:1.0},{27:1.0},{45:1.0},{48:1.0},]
Integrity (t)	DbStructure (t), Network (t), TableData (t), Cmd Execution (t), File System (t)	[{6:1.0},{289:1.0},{362:1.0},{436:1.0},]
Availability (t)	Cmd Execution (t), File System (t), TableData (t)	[{3:1.0},{12:1.0},{22:1.0},{42:1.0},]

Таблица 2

Модель перехода $P(X_{t+1}|X_t)$ ДБС фаззинга инъекций веб-приложений

Узел	Родительские вершины	Таблица условных вероятностей
Union Injection (t+1)	Union Injection (t)	[{0:1.0},{3:1.0}]

Code (t+1)	Code (t)	[{0:1.0},{8:1.0}]
Time Blind (t+1)	Time Blind (t)	[{0:1.0},{42:1.0}]
Error Blind (t+1)	Error Blind (t)	[{0:1.0},{133:1.0},{152:1.0},{190:1.0}]
Out Of Band (t+1)	Out Of Band (t)	[{0:1.0},{66:0.012048192771084338},{72:0.9879518072289156}]
Inject Type (t+1)	Inject Type (t)	[{0:1.0},{4:1.0},{8:1.0}]
Http Parameter Pollution (t+1)	Http Parameter Pollution (t)	[{0:1.0},{328:1.0},{410:1.0},{533:1.0},{615:1.0},{697:1.0},{820:1.0},{902:1.0},{943:1.0},{1025:1.0},{1312:1.0},{1476:1.0},{1517:1.0}]
Command (t+1)	Command (t)	[{0:1.0},{3:1.0}]
Dbms Fingerprint (t+1)	Dbms Fingerprint (t)	[{5:0.6610009442870632},{11:0.33899905571293676}]
Stacked Time (t)	Stacked Time (t)	[{0:1.0},]
Boolean Based Blind (t+1)	Boolean Based Blind (t)	[{0:1.0},{14:1.0},{42:1.0}]
Encoder (t+1)	Encoder (t)	[{0:1.0},{288:1.0},{408:1.0},{432:1.0},{480:1.0}]

Таблица 3

Модель восприятия $P(E_{t+1}|E_t)$ ДБС фаззинга инъекций веб-приложений

Узел	Родительские вершины	Таблица условных вероятностей
Network (t+1)	Dbms Fingerprint (t+1), Code (t+1), Command (t+1)	[{120:1.0},{264:1.0},{268:0.4936708860759494},{269:0.5063291139240507},{282:1.0}]
Cmd Execution (t+1)	Dbms Fingerprint (t+1), Code (t+1), Command (t+1)	[{120:1.0},{264:1.0},{268:0.4936708860759494},{269:0.5063291139240507},{282:1.0}]
File System (t+1)	Dbms Fingerprint (t+1), Code (t+1), Command (t+1)	[{123:1.0},{264:1.0},{268:0.4936708860759494},{269:0.5063291139240507},{282:1.0}]
Db Structure (t+1)	Dbms Fingerprint (t+1)	[{10:0.9},{11:0.1},{23:1.0}]
Table Data (t+1)	DbStructure (t+1)	[{0:1.0},{3:1.0}]
Confidentiality (t+1)	Db Structure (t+1), Network (t+1), Cmd Execution (t+1), File System (t+1)	[{6:1.0},{128:0.03861003861003861},{129:0.9613899613899614},{135:1.0},{170:1.0},{212:0.16666666666666666},{213:0.8333333333333334}]

Authentication (t+1)	File System (t+1), Table Data (t+1), Cmd Execution (t+1)	[{9:1.0},{27:1.0},{45:1.0},{48:1.0},{57:1.0}]
Authorization (t+1)	File System (t+1), TableData (t+1), Cmd Execution (t+1)	[{9:1.0},{27:1.0},{45:1.0},{48:1.0},{57:1.0}]
Integrity (t+1)	DbStructure (t+1), Network (t+1), Table Data (t+1), Cmd Execution (t+1), File System (t+1)	[{6:1.0},{288:0.03861003861003861},{289:0.9613899613899614},{295:1.0},{362:1.0},{436:0.16666666666666666},{437:0.8333333333333334}]
Availability (t+1)	Cmd Execution (t+1), File System (t+1), TableData (t+1)	[{2:0.03861003861003861},{3:0.9613899613899614},{12:1.0},{15:1.0},{22:1.0},{42:0.16666666666666666},{43:0.8333333333333334}]

Таблица 4

Начальное распределение $P(X_0)$ ДБС процесса фаззинга межсайтового скриптинга

Узел	Родительские вершины	Таблица условных вероятностей
Xss Type (t)		[{1:0.4805194805194805},{2:0.1168831168831169},{3:0.4025974025974026}]
Encoder (t)	Xss Type (t)	[{1:0.8288288288288288},{5:0.0810810810810811},{6:0.0900900900900901},{9:0.8518518518518519},{13:0.0740740740740741},{14:0.0740740740740741},{17:0.8279569892473119},{21:0.0860215053763441},{22:0.0860215053763441}]
Evasion (t)	Xss Type (t)	[{1:0.2882882882882883},{2:0.0810810810810811},{4:0.0720720720720721},{5:0.0810810810810811},{9:0.0810810810810811},{10:0.0810810810810811},{11:0.0720720720720721},{13:0.0810810810810811},{16:0.0810810810810811},{17:0.0810810810810811},{18:0.3333333333333333},{19:0.0740740740740741},{21:0.0740740740740741},{22:0.0740740740740741},{26:0.0740740740740741},{27:0.0740740740740741},{28:0.0740740740740741},{30:0.0740740740740741},{33:0.0740740740740741},{34:0.0740740740740741},{35:0.2580645161290323},{36:0.0860215053763441},{38:0.0860215053763441},{39:0.0860215053763441},{43:0.0860215053763441},{44:0.0860215053763441},{45:0.0752688172043011},{47:0.0860215053763441},{50:0.0752688172043011},{51:0.0752688172043011}]

Xss Payload (t)	Encoder (t), Evasion (t), Xss Type (t)	[{1:0.3846153846153846},{2:0.1538461538461539},{8:0.0769230769230769},{12:0.3846153846153846},{14:0.4000000000000000},{25:0.6000000000000000},{27:0.3750000000000000},{28:0.1250000000000000},{34:0.1250000000000000},{38:0.3750000000000000},{40:0.3333333333333333},{41:0.2222222222222222},{47:0.1111111111111111},{51:0.3333333333333333},{53:0.5000000000000000},{64:0.5000000000000000},{66:0.3750000000000000},{67:0.1250000000000000},{73:0.1250000000000000},{77:0.3750000000000000},{118:0.3750000000000000},{119:0.2500000000000000},{129:0.3750000000000000},{131:0.5000000000000000},{142:0.5000000000000000},{144:0.3750000000000000},{145:0.1250000000000000},{151:0.1250000000000000},{155:0.3750000000000000},{157:0.3333333333333333},{158:0.2222222222222222},{164:0.1111111111111111},{168:0.3333333333333333},{170:0.5000000000000000},{181:0.5000000000000000},{183:0.3750000000000000},{184:0.1250000000000000},{190:0.1250000000000000},{194:0.3750000000000000},{313:0.3333333333333333},{314:0.2222222222222222},{320:0.1111111111111111},{324:0.3333333333333333},{326:0.5000000000000000},{337:0.5000000000000000},{339:0.3750000000000000},{340:0.1250000000000000},{346:0.1250000000000000},{350:0.3750000000000000},{352:0.3333333333333333},{353:0.2222222222222222},{359:0.1111111111111111},{363:0.3333333333333333},{365:0.5000000000000000},{376:0.5000000000000000},{378:0.3750000000000000},{379:0.1250000000000000},{385:0.1250000000000000},{389:0.3750000000000000},{391:0.3750000000000000},{392:0.2500000000000000},{402:0.3750000000000000},{404:0.5000000000000000},{415:0.5000000000000000},{417:0.4285714285714285},{418:0.1428571428571428},{428:0.4285714285714285},{469:0.3333333333333333},{470:0.2222222222222222},{476:0.1111111111111111},{480:0.3333333333333333},{482:0.5000000000000000},{493:0.5000000000000000},{495:0.3750000000000000},{496:0.1250000000000000},{502:0.1250000000000000},{506:0.3750000000000000},{586:0.3333333333333333},{587:0.2222222222222222},{593:0.1111111111111111},{597:0.3333333333333333},{599:0.5000000000000000},{610:0.5000000000000000},{612:0.4285714285714285},{613:0.1428571428571428},{623:0.4285714285714285},{625:0.3333333333333333},{626:0.2222222222222222},{632:0.1111111111111111},{636:0.3333333333333333},{638:0.5000000000000000},{649:0.5000000000000000},{651:0.4285714285714285},{652:0.1428571428571428},{662:0.4285714285714285},{2653:0.3333333333333333},{2654:0.2222222222222222},{2660:0.1111111111111111},{2664:0.3333333333333333},{2666:0.5000000000000000},{2677:0.5000000

		000000000},{2679:0.3750000000000000},{2680:0.1250000000000000},{2686:0.1250000000000000},{2690:0.3750000000000000},{3316:0.3000000000000000},{3317:0.2000000000000000},{3323:0.1000000000000000},{3327:0.4000000000000000},{3329:0.5000000000000000},{3340:0.5000000000000000},{3342:0.3750000000000000},{3343:0.1250000000000000},{3349:0.1250000000000000},{3353:0.3750000000000000}]
Keylogger Module (t)	Xss Payload (t)	[{1:0.7126436781609196},{2:0.2873563218390804},{3:0.8333333333333334},{4:0.1666666666666667},{15:0.7894736842105263},{16:0.2105263157894737},{23:0.7078651685393258},{24:0.2921348314606741}]
Spy Eye Module (t)	Xss Payload (t)	[{1:0.7011494252873564},{2:0.2988505747126437},{3:0.8333333333333334},{4:0.1666666666666667},{15:0.7894736842105263},{16:0.2105263157894737},{23:0.6966292134831461},{24:0.3033707865168540}]
DDos Module (t)	Xss Payload (t)	[{1:0.7126436781609196},{2:0.2873563218390804},{3:0.8333333333333334},{4:0.1666666666666667},{15:0.7894736842105263},{16:0.2105263157894737},{23:0.7078651685393258},{24:0.2921348314606741}]
Port Scanner Module (t)	Xss Payload (t)	[{1:0.6321839080459770},{2:0.3678160919540230},{3:0.7777777777777778},{4:0.2222222222222222},{15:0.7894736842105263},{16:0.2105263157894737},{23:0.6179775280898876},{24:0.3820224719101123},]
Network Scanner Module (t)	Xss Payload (t)	[{1:0.7011494252873564},{2:0.2988505747126437},{3:0.8333333333333334},{4:0.1666666666666667},{15:0.7894736842105263},{16:0.2105263157894737},{23:0.6966292134831461},{24:0.3033707865168540}]
Nat Pinning Module (t)	Xss Payload (t)	[{1:0.7126436781609196},{2:0.2873563218390804},{3:0.8333333333333334},{4:0.1666666666666667},{15:0.7894736842105263},{16:0.2105263157894737},{23:0.7078651685393258},{24:0.2921348314606741},]
Drive By Download Module (t)	Xss Payload (t)	[{1:0.7011494252873564},{2:0.2988505747126437},{3:0.8333333333333334},{4:0.1666666666666667},{15:0.7894736842105263},{16:0.2105263157894737},{23:0.6966292134831461},{24:0.3033707865168540}]
Browser Fingerprint (t)	Xss Payload (t)	[{1:0.7011494252873564},{2:0.2988505747126437},{3:0.8055555555555556},{4:0.1944444444444444},{15:0.7894736842105263},{16:0.2105263157894737},{23:0.6966292134831461},{24:0.3033707865168540}]

Authentication (t)	Nat Pinning Module (t), Spy Eye Module (t), Drive By Download Module (t)	[{1:1.0000000000000000},{7:1.0000000000000000},{16:1.0000000000000000}]
Authorization (t)	Keylogger Module (t), Drive By Download Module (t), Spy Eye Module (t)	[{1:1.0000000000000000},{7:1.0000000000000000},{16:1.0000000000000000}]
Integrity (t)	Port Scanner Module (t), Drive By Download Module (t), Nat Pinning Module (t)	[{1:1.0000000000000000},{9:1.0000000000000000},{13:1.0000000000000000},{16:1.0000000000000000}]
Confidentiality (t)	Browser Fingerprint (t), Network Scanner Module (t), Spy Eye Module (t), Drive By Download Module (t)	[{1:1.0000000000000000},{15:1.0000000000000000},{17:1.0000000000000000},{32:1.0000000000000000}]

Таблица 5

Модель перехода $P(X_{t+1}|X_t)$ ДБС процесса фаззинга межсайтового скриптинга

Узел	Родительские вершины	Таблица условных вероятностей
Xss Type (t+1)	Xss Type (t)	[{1:1.0000000000000000},{5:1.0000000000000000},{9:1.0000000000000000}]
Encoder (t+1)	Encoder (t)	[{1:1.0000000000000000},{37:1.0000000000000000},{46:1.0000000000000000},]
Evasion (t+1)	Evasion (t)	[{1:1.0000000000000000},{19:1.0000000000000000},{55:1.0000000000000000},{73:1.0000000000000000},{145:1.0000000000000000},{163:1.0000000000000000},{181:1.000

		0000000000000},{217:1.0000000000000000},{271:1.0000000000000000},{289:1.0000000000000000}]
Xss Payload (t+1)	Xss Payload (t)	[{1:1.0000000000000000},{15:1.0000000000000000},{99:1.0000000000000000},{155:1.0000000000000000},]

Таблица 6

Модель восприятия $P(E_{t+1}|X_{t+1})$ ДБС процесса фаззинга межсайтового скриптинга

Узел	Родительские вершины	Таблица условных вероятностей
Keylogger Module (t+1)	Xss Payload (t+1)	[{1:0.6206896551724138},{2:0.3793103448275862},{3:0.6944444444444444},{4:0.3055555555555556},{15:0.6315789473684210},{16:0.3684210526315789},{23:0.6067415730337079},{24:0.3932584269662922},]
Spy Eye Module (t+1)	Xss Payload (t+1)	[{1:0.6091954022988506},{2:0.3908045977011494},{3:0.6944444444444444},{4:0.3055555555555556},{15:0.6315789473684210},{16:0.3684210526315789},{23:0.5955056179775281},{24:0.4044943820224719},]
DDos Module (t+1)	Xss Payload (t+1)	[{1:0.6206896551724138},{2:0.3793103448275862},{3:0.6944444444444444},{4:0.3055555555555556},{15:0.6315789473684210},{16:0.3684210526315789},{23:0.6067415730337079},{24:0.3932584269662922},]
Port Scanner Module (t+1)	Xss Payload (t+1)	[{1:0.5632183908045977},{2:0.4367816091954023},{3:0.6388888888888888},{4:0.3611111111111111},{15:0.6315789473684210},{16:0.3684210526315789},{23:0.5505617977528090},{24:0.4494382022471910},]
Network Scanner Module (t+1)	Xss Payload (t+1)	[{1:0.6091954022988506},{2:0.3908045977011494},{3:0.6944444444444444},{4:0.3055555555555556},{15:0.6315789473684210},{16:0.3684210526315789},{23:0.5955056179775281},{24:0.4044943820224719},]
Nat Pinning Module (t+1)	Xss Payload (t+1)	[{1:0.6206896551724138},{2:0.3793103448275862},{3:0.6944444444444444},{4:0.3055555555555556},{15:0.6315789473684210},{16:0.3684210526315789},{23:0.6067415730337079},{24:0.3932584269662922},]
Drive By Download Module (t+1)	Xss Payload (t+1)	[{1:0.6091954022988506},{2:0.3908045977011494},{3:0.6944444444444444},{4:0.3055555555555556},{15:0.6315789473684210},{16:0.3684210526315789},{23:0.5955056179775281},{24:0.4044943820224719},]

Browser Fingerprint (t+1)	Xss Payload (t+1)	[{1:0.6091954022988506},{2:0.3908045977011494},{3:0.6666666666666666},{4:0.3333333333333333},{15:0.6315789473684210},{16:0.3684210526315789},{23:0.5955056179775281},{24:0.4044943820224719},]
Authentication (t+1)	Nat Pinning Module (t+1), Spy Eye Module (t+1), Drive By Download Module (t+1), DDos Module (t+1)	[{1:1.0000000000000000},{13:1.0000000000000000},{32:1.0000000000000000}]
Authorization (t+1)	Keylogger Module (t+1), Drive By Download Module (t+1), Spy Eye Module (t+1), DDos Module (t+1)	[{1:1.0000000000000000},{13:1.0000000000000000},{32:1.0000000000000000}]
Integrity (t+1)	Port Scanner Module (t+1), Drive By Download Module (t+1), Nat Pinning Module (t+1)	[{1:1.0000000000000000},{9:1.0000000000000000},{13:1.0000000000000000},{16:1.0000000000000000}]
Confidentiality (t+1)	Browser Fingerprint (t+1), Network Scanner Module (t+1),Spy Eye Module (t+1), Drive By Download Module (t+1)	[{1:1.0000000000000000},{15:1.0000000000000000},{17:1.0000000000000000},{32:1.0000000000000000}]

Приложение Б. Таблицы апостериорных вероятностей динамических байесовский сетей процесса фаззинга

Таблица 1

Полное совместное распределение $P(X_{t+1}|E_{t+1})$ ДБС фаззинга инъекций веб-приложений

Узел	Родительские вершины	Таблица условных вероятностей
Inject Type (t+1)	-	[{0:0.7541800862912437},{1:0.12286461583354008},{2:0.12295529787521621}]
Http Parameter Polution (t+1)	Inject Type (t+1)	[{10:0.10442210774429189},{13:0.051082895748726195},{15:0.04812418445391252},{17:0.10368311673711407},{20:0.09753970505139588},{22:0.08934574819427463},{23:0.09324833707783242},{25:0.09343690014895698},{32:0.11328826333793954},{36:0.11001097812510209},{37:0.09581776338045384},{48:0.48267958220329976},{50:0.5173204177967003},{90:1.0},]
Encoder (t+1)	Inject Type (t+1)	[{12:0.33456312025778046},{17:0.3530048860557451},{18:0.3124319936864744},{43:1.0}]
Union Injection (t+1)	Inject Type (t+1), Encoder (t+1), Http Parameter Polution (t+1)	[{20:0.6017970999552655},{21:0.3982029000447344},{27:1.0},{30:0.4608957053159531},{31:0.5391042946840469},{35:1.0},{41:1.0},{45:1.0},{47:1.0},{51:1.0},{64:0.5969925022774982},{65:0.40300749772250183},{72:0.6264665440236445},{73:0.3735334559763554},{75:1.0},{961:1.0},{1361:1.0},{1441:1.0},{1856:1.0},{1860:1.0},{3440:1.0},{3700:1.0}]
Boolean Based Blind (t+1)	Inject Type (t+1), Encoder (t+1), Http Parameter Polution (t+1)	[{131:0.5925884312450667},{133:0.4074115687549334},{170:1.0},{196:0.7153645070867654},{198:0.2846354929132346 4},{222:1.0},{261:1.0},{287:1.0},{300:1.0},{326:1.0},{417:0.5527273576333999},{419:0.4472726423666001},{469:0.5314825505097532},{471:0.4685174494902468},{482:1.0},{6241:1.0},{8841:1.0},{9361:1.0},{12064:1.0},{12090:1.0},{22360:1.0},{24050:1.0}]
Time Blind (t+1)	Inject Type (t+1), Encoder (t+1), Http Parameter Polution (t+1)	[{133:1.0},{172:1.0},{198:1.0},{224:1.0},{263:1.0},{289:1.0},{302:1.0},{328:1.0},{419:1.0},{471:1.0},{484:1.0},{6243:1.0},{8843:1.0},{9363:1.0},{12064:1.0},{12090:1.0},{22360:1.0},{24050:1.0}]
Error Blind (t+1)	Inject Type (t+1), Encoder (t+1), Http Parameter Polution (t+1)	[{187:0.1634479053909441},{188:0.4077348920711403},{190:0.4288172025379157},{244:1.0},{277:0.1121972034521021},{278:0.37558847546532886},{280:0.5122143210825 69},{316:1.0},{370:1.0},{406:1.0},{424:1.0},{460:1.0},{583:0.13767976573508567},{584:0.44253337966121253},{5

		86:0.41978685460370185},{655:0.13078913179211527},{656:0.5067673175181318},{658:0.362443550689753},{676:1.0},{8650:1.0},{12250:1.0},{12970:1.0},{16704:1.0},{16740:1.0},{30960:1.0},{33300:1.0}]
Stacked Time (t+1)	Inject Type (t+1), Encoder (t+1), Http Parameter Polution (t+1)	[{120:1.0},{156:1.0},{180:1.0},{204:1.0},{240:1.0},{264:1.0},{276:1.0},{300:1.0},{384:1.0},{432:1.0},{444:1.0},{5760:1.0},{8160:1.0},{8640:1.0},{11136:1.0},{11160:1.0},{20640:1.0},{22200:1.0}]
Out Of Band (t+1)	Inject Type (t+1), Encoder (t+1), Http Parameter Polution (t+1)	[{116:1.0},{149:1.0},{171:1.0},{193:1.0},{226:1.0},{248:1.0},{259:1.0},{281:1.0},{358:1.0},{402:1.0},{413:1.0},{5286:1.0},{7486:1.0},{7926:1.0},{10208:1.0},{10230:1.0},{18920:1.0},{20350:1.0}]
Code (t+1)	Inject Type (t+1), Encoder (t+1), Http Parameter Polution (t+1)	[{30:1.0},{39:1.0},{45:1.0},{51:1.0},{60:1.0},{66:1.0},{69:1.0},{75:1.0},{96:1.0},{108:1.0},{111:1.0},{1440:1.0},{2040:1.0},{2160:1.0},{2784:1.0},{2790:1.0},{5160:1.0},{5552:1.0}]
Command (t+1)	Inject Type (t+1), Encoder (t+1), Http Parameter Polution (t+1)	[{20:1.0},{26:1.0},{30:1.0},{34:1.0},{40:1.0},{44:1.0},{46:1.0},{50:1.0},{64:1.0},{72:1.0},{74:1.0},{960:1.0},{1360:1.0},{1440:1.0},{1857:1.0},{1861:1.0},{3441:1.0},{3700:1.0}]
Dbms Fingerprint (t+1)	Union Injection (t+1), Boolean Based Blind (t+1), Error Blind (t+1), Time Blind (t+1), Stacked Time (t+1), Out Of Band (t+1)	[{540221:1.0}, {1260941:1.0}, {1281533:1.0}, {5399933:1.0}]
Network (t+1)	Dbms Fingerprint (t+1), Code (t+1), Command (t+1)	[{120:1.0}]
Cmd Execution (t+1)	Dbms Fingerprint (t+1), Code (t+1), Command (t+1)]	[{120:1.0}]

File System (t+1)	Dbms Fingerprint (t+1), Code (t+1), Command (t+1)	[{123:1.0}]
DbStructure (t+1)	Dbms Fingerprint (t+1)	[{10:1.0}]

Таблица 2

Полное совместное распределение $P(X_{t+1}|E_{t+1})$ ДБС фаззинга межсайтового скриптинга веб-приложений

Узел	Родительские вершины	Таблица условных вероятностей
Xss Type (t+1)	-	[{0:0.54}, {1:0.05}, {2:0.41}]
Encoder (t+1)	Xss Type (t+1)	[{0:1.0}, {8:1.0}, {16:0.9512195121951219}, {20:0.04878048780487805}]
Evasion (t+1)	Xss Type (t+1)	[{0:0.37037037037037035},{1:0.07407407407407407},{3:0.1111111111111111},{4:0.05555555555555555},{8:0.018518518518518517},{9:0.12962962962962962},{10:0.07407407407407407},{12:0.018518518518518517},{15:0.07407407407407407},{16:0.07407407407407407},{17:0.6},{21:0.2},{26:0.2},{34:0.34146341463414637},{37:0.14634146341463414},{38:0.07317073170731707},{42:0.04878048780487805},{43:0.0975609756097561},{44:0.07317073170731707},{46:0.14634146341463414},{50:0.07317073170731707}]
Xss Payload (t+1)	Xss Type (t+1), Encoder (t+1), Evasion (t+1)	[{0:0.5},{1:0.1},{7:0.3},{11:0.1},{14:0.5},{24:0.5},{39:0.16666666666666666},{40:0.5},{50:0.3333333333333333},{52:0.6666666666666666},{59:0.3333333333333333},{115:1.0},{117:0.14285714285714285},{118:0.14285714285714285},{128:0.7142857142857143},{131:1.0},{156:1.0},{195:1.0},{219:1.0},{1768:0.3333333333333333},{1779:0.6666666666666666},{1831:1.0},{1885:1.0},{3536:0.6666666666666666},{3547:0.3333333333333333},{3575:0.3333333333333333},{3576:0.6666666666666666},{3588:0.6666666666666666},{3599:0.3333333333333333},{3651:1.0},{3653:0.25},{3660:0.25},{3664:0.5},{3667:0.3333333333333333},{3677:0.6666666666666666},{3692:0.3333333333333333},{3693:0.16666666666666666},{3699:0.3333333333333333},{3703:0.16666666666666666},{3744:1.0},{4427:1.0}]

Приложение В. Акты внедрения

УТВЕРЖДАЮ
Директор «Центра системных исследований и разработок» – филиал АО «Научно-технических центр РЭБ»
кандидат технических наук, доцент

А.А. Болкунов
«__» марта 2024 г.

АКТ
об использовании результатов
докторской диссертационной работы Полухина П.В.

Комиссия в составе: председателя комиссия – главного инженера, к.т.н, доцента Булычева О.А., членов комиссии – ведущего научного сотрудника, д.т.н., профессора Лихачева В.П. и начальника отдела, к.т.н., с.н.с. Юзвенко С.В. установила, что практические результаты диссертационного исследования Полухина Павла Валерьевича «Методы организации процесса фаззинг-тестирования и анализа веб-приложений на основе моделей динамических байесовских сетей» обладают научной новизной и могут быть внедрены в технологический процесс центра при выполнении опытно-конструкторских работ, шифр «Поле-21Э» и «Матрица», а также использованы в рамках создания современных комплексов РЭБ для защиты от высокоточного оружия, в том числе с применением беспилотных воздушны судов, различных модификаций.

Полученные в работе результаты позволили оптимизировать процедуру тестирования разрабатываемых комплексов, снизить временные и ресурсные затраты по оценке надежности перспективных программно-аппаратных комплексов РЭБ.

Председатель комиссии:
Главный инженер, к.т.н., доцент О. А. Булычев

Члены комиссии:
Ведущий научный сотрудник, д.т.н., профессор В.П. Лихачев

Начальник отдела, к.т.н., с.н.с. С.В. Юзвенко

УТВЕРЖДАЮ
Врио заместителя начальника ВУНЦ ВВС
«ВВА им. профессора Н.Е. Жуковского и
Ю.А. Гагарина» (г. Воронеж)
по учебной и научной работе
доктор технических наук, доцент
полковник
Е. Кравцов

АКТ

реализации в образовательном процессе результатов диссертационной работы на соискание ученой степени доктора наук Полухина Павла Валерьевича на тему «Методы организации процесса фаззинг-тестирования и анализа веб-приложений на основе моделей динамических байесовских сетей»

Комиссия в составе:
председателя комиссии:
врио начальника учебно-методического центра ВУНЦ ВВС «ВВА им. проф. Н.Е. Жуковского и Ю.А. Гагарина» (г. Воронеж) к.т.н. полковника Первезенцева Р.Е.
членов комиссии:
врио начальника 54 кафедры к.т.н. полковника Волкова Е.А.
(должность, в/зв., Ф.И.О.)
профессора 54 кафедры д.т.н., профессора полковника Самойлина Е.А.
(должность, в/зв., Ф.И.О.)
доцента 54 кафедры к.т.н., доцента Серебрякова М.А.
(должность, в/зв., Ф.И.О.)

составила настоящий акт о том, что материалы диссертационной работы Полухина П.В. «Методы организации процесса фаззинг-тестирования и анализа веб-приложений на основе моделей динамических байесовских сетей» реализованы в образовательном процессе на 54 кафедре (противодействия техническим средствам разведки) при проведении группового занятия № 2 «Принципы построения аппаратуры комплексов и средств защиты информации и контроля» (тема № 1 «Задачи и классификация комплексов и средств защиты информации и контроля») по дисциплине «Комплексы и средства защиты информации и контроля».

Председатель комиссии: полковник Р. Первезенцев
(в/зв., подпись, Ф.И.О.)

Члены комиссии: полковник А. Волоков
(в/зв., подпись, Ф.И.О.)
полковник Е. Самойлин
(в/зв., подпись, Ф.И.О.)
г.п. М. Серебряков
(в/зв., подпись, Ф.И.О.)

«18» марта 2024 года

УТВЕРЖДАЮ
проректор по науке, инновациям и цифровизации ФГБОУ ВО «ВГУ»
_____Д.В. Костин
«26» февраля 2024 г.

АКТ

о внедрении результатов научно-исследовательской работы П.В. Полухина «Методы организации процесса фаззинг-тестирования и анализа веб-приложений на основе моделей динамических байесовских сетей» в образовательный процесс ФГБОУ ВО «Воронежский государственный университет»

Результаты научно-исследовательской работы Полухина П.В. по теме диссертации «Методы организации процесса фаззинг-тестирования и анализа веб-приложений на основе моделей динамических байесовских сетей», включающие математические модели, методы, алгоритмическое и программное обеспечение, направленные на оптимизацию информационных процессов тестирования и способствующие повышению качества обнаружения ошибок функционирования веб-приложений использовались в учебных курсах «Технология тестирования программ», «Вычислительные системы, сети, телекоммуникации», «Администрирование информационных систем» и «Безопасность интернет-приложений», читаемых магистрам направления 01.04.02 Прикладная математика и информатика, в учебных курсах «Современные Internet-технологии», «Защита программ и данных», «Интеллектуальный анализ данных», читаемых магистрам по направлению 02.04.02 «Фундаментальная информатика и информационные технологии», в учебных курсах «Разработка интернет-приложений», «Управление ИТ-сервисами и контентом», читаемых магистрам направления 38.04.05 «Бизнес-информатика», в учебном курсе «Основы построения защищенных компьютерных сетей», читаемом студентам по программе специалитета 0.05.01 «Компьютерная безопасность», а также в качестве методического обеспечения курсовых и выпускных квалификационных работ по направлению «Фундаментальная информатика и информационные технологии», «Прикладная математика и информатика».

Декан факультета
прикладной математики и механики ВГУ,
к.ф.-м.н., доцент С.Н. Медведев

Заведующий кафедрой
Математических методов исследования операций,
д.т.н., профессор Т.В. Азарнова

АКЦИОНЕРНОЕ ОБЩЕСТВО
«НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ РЕЛЭКС»
394006, Воронеж, ул. 20-летия Октября, 119
Тел./факс: /473/2-711-711; 2-778-333

АКТ о внедрении результатов исследования

Настоящим актом подтверждается, что АО НПП «РЕЛЭКС» – участником Фонда «Сколково» – был проведен анализ моделей, методов, алгоритмов и программного обеспечения по организации процесса фаззинг-тестирования программных продуктов, предложенных в диссертации Полухина П.В. «Методы организации процесса фаззинг-тестирования и анализа веб-приложений на основе моделей динамических байесовских сетей». Анализ качественных и количественных характеристик предложенных в работе Полухина П.В. решений показал их высокую эффективностью для решения задач обнаружения программных ошибок, а также снижение временных и ресурсных затрат на проведение комплексного фаззинг-тестирования. Комплекс проблемно-ориентированных программ, базирующихся на динамических байесовских моделях и алгоритмах организации процедуры тестирования методом фаззинга успешно внедрен в производственную деятельность компании.

Полученные в процессе апробации внедренных решений результаты демонстрируют существенное повышение производительности процессов тестирования и высокие прогностические возможности предложенных в работе алгоритмов вероятностного вывода.

Генеральный директор
АО НПП «РЕЛЭКС» И.А. Бойченко

05.03.2024 г.

Приложение Г. Свидетельства о регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU **2021666156**

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2021666156

Дата регистрации: **08.10.2021**

Номер и дата поступления заявки:
2021665292 01.10.2021

Дата публикации: **08.10.2021**

Контактные реквизиты:
394018, г. Воронеж,
Университетская пл., 1, УНИИП

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (ФГБОУ ВО «ВГУ») (RU)

Название программы для ЭВМ:
«Гибридные механизмы обучения и вероятностного вывода для статических и динамических байесовских сетей»

Реферат:
Программа предназначена для реализации оптимальных подходов к разработке структуры, обучению и реализации вероятностного вывода для байесовских сетей. Программой используется ряд гибридных алгоритмов вероятностного вывода, обучения структуры и параметров байесовских сетей. Гибридные методы на основе аппарата математической статистики, численных методов и эвристических подходов позволяют повысить эффективность классических алгоритмов и расширить функциональные возможности математического аппарата байесовских сетей. Предложены специальные гибридные инструменты оптимизации процедур обучения байесовских сетей с неполными данными. Для решения проблем, связанных с производительностью алгоритмов, снижением временных затрат на моделирование с использованием байесовских сетей используется набор программных компонентов системы Spark, реализующей облачный подход к обработке и распараллеливанию блоков данных. ОС: Windows, Linux, MacOS X.

Язык программирования: Python

Объем программы для ЭВМ: 850 КБ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016619513

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016619513

Дата регистрации: **22.08.2016**

Номер и дата поступления заявки:
2016613485 12.04.2016

Дата публикации: **20.09.2016**

Контактные реквизиты:
394006, г. Воронеж, Университетская пл., 1, ФГБОУ ВО "ВГУ", Центр коммерциализации технологий

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Оценка эффективности тестирования уязвимостей, связанных с нарушением управления доступом интернет-приложений»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний, предоставляющих услуги по аудиту и анализу защищённости приложений. Программа позволяет качественно и эффективно анализировать существующие уязвимости для тестируемых приложений, а также осуществлять прогнозирование их возникновения в будущем. В программе предусмотрено несколько основных методов обнаружения уязвимостей нарушения управления доступом веб-приложений. Первый подход подразумевает анализ веб-приложений на предмет наличия ресурсов, которые должны быть защищены в рамках реализации аутентификации, однако к ним существует доступ. Это позволяет выявить проблемы, связанные с неправильной реализацией подхода в аутентификации.

Тип реализующей ЭВМ: IBM PC-совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, Mac OS X

Объем программы для ЭВМ: 530,44 Кб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016619512

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016619512

Дата регистрации: **22.08.2016**

Номер и дата поступления заявки:
2016613484 12.04.2016

Дата публикации: **20.09.2016**

Контактные реквизиты:
394006, г. Воронеж, Университетская пл., 1, ФГБОУ ВО "ВГУ", Центр коммерциализации технологий

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Особенности выявления использования компонент с уязвимостями на основе технологии фаззинга»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний, предоставляемых услуги по аудиту и анализу защищённости приложений. Программа позволяет в значительной мере дать комплексную оценку защищенности веб-приложений на предмет наличия уязвимых компонентов. Программа использует ресурсы двух баз данных уязвимостей - National Vulnerability Database (NIST) и Exploit Database (Offensive Security). Исходя из особенностей данного типа уязвимостей, программа состоит из нескольких алгоритмов.

Тип реализующей ЭВМ: IBM PC-совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, Mac OS X

Объем программы для ЭВМ: 891 Кб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016616720

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016616720

Дата регистрации: **17.06.2016**

Номер и дата поступления заявки:
2016612433 21.03.2016

Дата публикации: **20.07.2016**

Контактные реквизиты:
394006, г. Воронеж, Университетская площадь, 1. ФГБОУ ВПО "ВГУ", Центр коммерциализации технологий

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Программа повышения качества оценки защищенности веб-приложений от небезопасных ссылок на объекты с помощью фаззинга»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний предоставляемых услуги по аудиту и анализу защищённости приложений. В качестве результата выполнения программы выступает оценка вероятности появления уязвимостей небезопасных ссылок на объекты в рамках тестирования группы приложений и прогнозирования их появления на следующих временных интервалах. Это позволяет дать оценку защищенности приложения в некотором будущем состояние, в частности после выпуска обновлений. В рамках разработки программы представлено разделение объектов данных уязвимостей на файлы, папки и записи базы данных. Стоит отметить, что данная уязвимость применительно к файловой системе сервера, то есть к файлам и папкам приложений несет в себе дополнительную уязвимость раскрытие путей (Path Traversal). Данная уязвимость позволяет получить доступ к файлам, директориям и командам, находящимся вне основной директории веб-сервера. Программа изменяет параметры URL с целью получения доступа к файлам или выполнить команды, располагаемые в файловой системе веб-сервера.

Тип реализующей ЭВМ: IBM PC - совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, MacOS X

Объем программы для ЭВМ: 822,44 Кб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016616719

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016616719

Дата регистрации: **17.06.2016**

Номер и дата поступления заявки:
2016612407 21.03.2016

Дата публикации: **20.07.2016**

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Механизмы анализа защищенности систем аутентификации и управления сессиями веб-приложений на основе технологии фаззинга»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний, предоставляемых услуги по аудиту и анализу защищённости приложений. Результатом работы программы является формирование количественной оценки защищённости приложений, а также вероятностного прогноза присутствия уязвимостей в новых приложениях на основе статистической информации, полученной ранее. Для прогнозирования появления данного рода уязвимостей в приложениях со сходным набором программных модулей и компонентов, используется математический аппарат динамический сетей Байеса, а для решения задач фильтрации, прогнозирования и сглаживания используются рандомизированные алгоритмы выборки, в частности алгоритм фильтрации частиц.

Тип реализующей ЭВМ: IBM PC - совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, MacOS X

Объем программы для ЭВМ: 879,93 Кб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016616336

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016616336

Дата регистрации: **09.06.2016**

Номер и дата поступления заявки:
2016613611 13.04.2016

Дата публикации: 20.07.2016

Контактные реквизиты:
394006, г. Воронеж, Университетская площадь, 1, ФГБОУ ВО «ВГУ», Центр коммерциализации технологий

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Математические модели, методы и алгоритмы тестирования раскрытия персональных данных с помощью фаззинга»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний, предоставляющих услуги по аудиту и анализу защищённости приложений. Тестирование раскрытия персональных данных, реализованное в программе, инкапсулирует ряд важных подходов к процессу тестирования данного типа уязвимостей. В рамках первого подхода производится анализ защищённости протокола взаимодействия клиента и веб-приложений, в особенности детально проверяются наиболее критичные функциональные части тестовой программы. К таким функциональным частям относятся система аутентификации, система управления ролями. Для приложений, использующих инструментарий денежных и финансовых операции, все действия должны быть проворены на передачу параметров только по защищенному протоколу SSL/TLS. Во втором подходе рассматривается анализ так называемых кросс-запросных хранилищ: cookie, которые ассоциируются с переменной сеанса сервера, и переменные состояние (ViewState). Тестирование ViewState особенно важно, если анализируемое приложение написано на Фреймворке ASP.NET или использует контейнер сервлетов Java. Данные контейнеры могут использоваться для передачи дополнительных параметров, в частности раскрывающих, как технические особенности функционирования приложений, так и параметры аутентификации, авторизации, специальные сеансовые идентификаторы и т.д. Третий подход предусматривает выявление элементов разметки браузера, позволяющих сохранять историю, в частности формы с автозаполнением, показывающие все комбинации данных, которые когда-то вводились, что приводит к возможности кражи аутентификационных и иных критичных данных, например, с помощью межсайтового скриптинга. Результатом работы программы является комплексная многогранная оценка и прогнозирование появления уязвимостей на основе имеющихся накопленных тестовых данных. Для прогнозирования уязвимостей раскрытия персональных

данных в программе используется математический аппарат динамических сетей Байеса (ДБС), при этом прогнозирование и фильтрация происходят с помощью вероятностного вывода. Построение ДБС происходит из вероятностных данных, полученных в результате тестирования целевых приложений. Программа использует многокритериальный параметр, позволяющий объединять тестируемые приложений со сходным набором программных компонентов и зависимостей в группы, что даёт возможность тестировать как единичные несвязанные приложения, так и приложения, имеющие сходные признаки, к примеру, одинаковые фреймворки, системы управления сайтами, программные библиотеки или среду исполнения. Под построением ДБС в рамках работы алгоритма программы подразумеваются вычисления апостериорных вероятностей тестовой подсистемы, в общем случае, для момента времени t-1 за счет тестирования методом черного ящика, а также получения модели перехода и восприятия. Для получения прогнозируемого выбора в программе используется алгоритм фильтрации частиц, который позволяет получить наиболее правдоподобные выборки, в полной мере согласующиеся со свидетельством. Под свидетельством в контексте тестирования понимается некоторая эпизодическая переменная, связанная в той или иной мере с некоторым событием. В ходе тестирования свидетельствами будут являться некоторые исходы этапов процесса тестирования, а именно, целевые угрозы безопасности, которые могут возникнуть при наличии данного типа уязвимостей.

Тип реализующей ЭВМ: IBM PC-совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, MacOS X

Объем программы для ЭВМ: 378 Кб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016616243

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016616243

Дата регистрации: **08.06.2016**

Номер и дата поступления заявки:
2016613482 12.04.2016

Дата публикации: **20.07.2016**

Контактные реквизиты:
394006, г. Воронеж, Университетская пл., 1, ФГБОУ ВО "ВГУ", Центр коммерциализации технологии

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Механизмы повышения эффективности и качества тестирования уязвимостей межсайтового подделки запросов (CSRF) с помощью фаззинга»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний, предоставляемых услуги по аудиту и анализу защищённости приложений. Методика тестирования CSRF, предложенная в рамках программы, позволяет дать комплексную оценку защищенности приложения на предмет наличия данного типа уязвимостей. Обобщенный алгоритм тестирования CSRF в программе разделен на два функциональных подхода, исходя из природы возникновения CSRF. В качестве механизма прогнозирования появления данного типа уязвимостей в программе реализован подход к построению Динамической сети Байеса, а также использования алгоритма вероятностного вывода на основе алгоритма фильтрации частиц для получения наиболее правдоподобной выборки для прогнозируемого момента времени. Результатом работы программы является формирование количественной оценки защищённости приложений, а также вероятностного прогноза присутствия уязвимостей в новых приложениях на основе статистической информации, полученной ранее.

Тип реализующей ЭВМ: IBM PC - совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, MacOS X

Объем программы для ЭВМ: 1.02 Мб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016616241

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016616241

Дата регистрации: **08.06.2016**

Номер и дата поступления заявки:
2016613483 12.04.2016

Дата публикации: **20.07.2016**

Контактные реквизиты:
394006, г. Воронеж, Университетская пл., 1, ФГБОУ ВО "ВГУ", Центр коммерциализации технологии

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Комплексный подход к анализу уязвимостей небезопасных направлений веб-приложений»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний, предоставляемых услуги по аудиту и анализу защищённости приложений. Для решения задачи комплексного тестирования небезопасных перенаправлений в разработанной программе предусмотрен ряд подходов, позволяющих дать многогранную оценку защищённости интернет-приложений. В рамках разработки программ нами предложены оригинальные методы, направленные на повышение эффективности обнаружения данного типа уязвимостей путем применения фаззинга.

Тип реализующей ЭВМ: IBM PC - совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, MacOS X

Объем программы для ЭВМ: 445,77 Кб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016615101

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016615101

Дата регистрации: **16.05.2016**

Номер и дата поступления заявки:
2016612406 21.03.2016

Дата публикации: **20.06.2016**

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Методология тестирования инъекции веб-приложений с помощью фаззинга»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний предоставляемых услуги по аудиту и анализу защищённости приложений. Программа позволяет: моделировать работу специалиста по аудиту безопасности; проанализировать степень критичности уязвимости типа инъекция и оценить последствия ее раскрытия; тестировать SQL инъекции, инъекции кода и инъекции команд. Результатом работы программы является формирование количественной оценки защищённости приложений, а также вероятностного прогноза присутствия "уязвимостей" в новых приложениях на основе статистической информации, полученной ранее.

Тип реализующей ЭВМ: IBM PC - совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, MacOS

Объем программы для ЭВМ: 4,6 Мб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016615100

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016615100

Дата регистрации: **16.05.2016**

Номер и дата поступления заявки:
2016612434 21.03.2016

Дата публикации: 20.06.2016

Контактные реквизиты:
394006, г. Воронеж, Университетская площадь, 1, ФГБОУ ВО "ВГУ", Центр коммерциализации технологий

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Комплексная методика тестирования уязвимостей нарушения конфигурации системы»

Реферат:
Программа предназначена для организаций, специализирующихся на разработке программных продуктов на этапах тестирования, а также компаний предоставляемых услуги по аудиту и анализу защищённости приложений. Программа представляет собой комплексное решение по тестированию нарушений конфигурации системы, при этом позволяет осуществлять прогноз появления данных уязвимостей. Программа позволяет выявить уязвимости, связанные с неправильной установкой параметров для всех компонентов веб-приложений: фреймворков, веб-серверов, сервера баз данных, исполняемой платформы и дополнительных конфигураций. При этом комплексность тестирования особенно важна в тех случаях, когда если среда исполнения и функционирования для нескольких приложений одна и также, например, различного рода хостинги и облачные платформы, при этом ошибки конфигурации могут быть удвоены потенциальными рисками. Программа позволяет анализировать раскрытия данного типа уязвимостей за счет попытки использовать пароли и учетные записи, в частности паролей, которые, как правило, устанавливаются по умолчанию, просматривая ошибки и стек вызовов , систем управления базами данных и веб-фреймворков, а также исполняемой среды: (веб-сервера и сервера приложений), что позволяет получить полную и детальную информацию о внутренних механизмах функционирования приложения, анализировать обрабатываемые параметры и программную логику. Для прогнозирования возникновения уязвимостей в программе реализован математический аппарат динамических Байесовских сетей (ДБС). В рамках создания используются два временных интервала (среза). Первый интервал характеризует состояние тестовой системы без параметров безопасности, а второй с примененными обновлениями безопасности. Это позволяет дать оценку эффективности и корректности написания и применения механизмов защиты, так как они должны покрывать все обнаруженные уязвимости. Для построения ДБС производится вычисление апостериорных вероятностей на основе фаззинга методом черного ящика, при этом, считается, что программа не знает, какое

именно приложение мы тестируем, а именно, какие параметры конфигурации она имеет. Для построения модели перехода и восприятие происходит адаптирование и применение механизмов защиты к тестируемым приложениям и цикл фаззинга повторяется еще раз. После построения ДБС для формирования выборок, согласующихся со свидетельствами, используется алгоритм фильтрации частиц. Данный алгоритм позволяет решить задачи прогнозирования фильтрации и сглаживания. Результатом работы программы является вероятностная оценка появления уязвимостей нарушения конфигурации системы, при наличии некоторых свидетельств. В качестве свидетельств в контексте тестирования выступают некоторые переменные - следствия, которые возникают при обнаружении уязвимостей нарушения конфигурации системы.

Тип реализующей ЭВМ: IBM PC - совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, MacOS X

Объем программы для ЭВМ: 677.07 Кб

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2016614976

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2016614976

Дата регистрации: **12.05.2016**

Номер и дата поступления заявки:
2016612410 21.03.2016

Дата публикации: **20.06.2016**

Контактные реквизиты:
394006, г. Воронеж, Университетская пл., 1, ФГБОУ ВО "ВГУ", центр коммерциализации технологий

Авторы:
Азарнова Татьяна Васильевна (RU),
Полухин Павел Валерьевич (RU)

Правообладатель:
федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (RU)

Название программы для ЭВМ:
«Обнаружение и анализ уязвимостей межсайтового скриптинга (XSS) современных интернет приложений»

Реферат:
Программа предназначена как для организаций и компаний, специализирующихся на разработке веб-приложений, так и занимающихся исключительно аудитом безопасности. Функциональные возможности программы базируются на основных видах XSS уязвимостей, а также алгоритмах их обнаружения. Для решения задачи прогнозирования появления уязвимостей, а также снижения временных затрат на тестирования, в рамках разработки программы предложено использовать математический аппарат Динамических байесовских сетей (ДБС). Для реализации алгоритма построения ДБС используются механизмы получения начального распределения, модели перехода и восприятия, необходимые для установления вероятностных характеристик двух временных срезов. В качестве алгоритма вероятностного вывода используется алгоритм фильтрации частиц.

Тип реализующей ЭВМ: IBM PC - совмест. ПК

Язык программирования: Java

Вид и версия операционной системы: Windows, Linux, MacOS X

Объем программы для ЭВМ: 2,5 Мб