ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Токарева Виктория Андреевна

Математические модели и алгоритмы для формирования расписания в распределённых системах обработки данных с агрегированным доступом к информационным ресурсам

Специальность 05.13.18
«Математическое моделирование, численные методы и комплексы программ»

ДИССЕРТАЦИЯ
на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук,
доктор технических наук, доцент
Бондаренко Юлия Валентиновна

Воронеж — 2022

Оглавление

Стр.

Введение 5

Глава 1. Проблемы составления расписаний в многолинейных системах обработки информации с ограничениями на ресурсы 13

1.1 Проблемы теории расписаний: классификация, нотация, основные определения, примеры 13

1.1.1 Подходы к классификации задач ТР 15

1.1.2 Разрешимость задач теории расписаний и их алгоритмическая сложность 17

1.1.3 Постановка задач составления расписаний 19

1.1.4 Нотация, используемая при записи задач ТР 23

1.2 Понятие систем агрегации и представление о них, как об объекте моделирования 26

1.2.1 Понятие системы агрегации 26

1.2.2 Функциональная модель составления адаптивных расписаний в системе агрегации 29

1.2.3 Логическая модель «сущность-связь» для составления адаптивных расписаний в системе агрегации 32

1.2.4 Теоретико-множественное описание системы агрегации . . 34

1.2.5 Характеристики задач составления расписаний в СА 36

1.3 Модели и методы составления расписаний в условиях ограниченных ресурсов 36

1.3.1 Анализ моделей и методов составления расписаний для систем с непрерывными возобновимыми ресурсами 41

1.4 Выводы и результаты по главе 1 49

Глава 2. Математические модели обработки данных в распределённых гибридных архитектурах 51

2.1 Дискретная модель многолинейной системы с ограниченными восполнимыми ресурсами для систем агрегации 51

Стр.

2.2 Дискретно-непрерывная модель многолинейной системы с ограниченными восполнимыми ресурсами для систем агрегации . . 57
2.3 Построение приоритето-порождающего функционала системы с ограниченными восполнимыми ресурсами 68
2.4 Выводы и результаты по главе 2 75

Глава 3. Алгоритмы построения расписаний обслуживания задач в многолинейной системе с ограниченными восполнимыми ресурсами 76
3.1 Построение расписаний в случае выделенных приборов ($m = S$) . . 77
3.1.1 Алгоритмы, основанные на функционалах 1-приоритета . . 77
3.1.2 Эвристический алгоритм диспетчеризации задач в СА в случае выделенных приборов 80
3.2 Построение расписаний в случае индивидуальных независимых приборов ($m < S$) 83
3.2.1 Алгоритмы, основанные на приоритето-порождающих функционалах 83
3.2.2 Эвристический алгоритм диспетчеризации задач в многолинейной системе с ограниченными восполнимыми ресурсами для систем с агрегацией 87
3.3 Выводы и результаты по главе 3 95

Глава 4. Численные эксперименты и комплексы программ 97
4.1 Применение приоритето-порождающих функционалов 1-го порядка для построения расписаний в многолинейной СА 97
4.1.1 Постановка численного эксперимента 97
4.1.2 Описание хода численного эксперимента 97
4.1.3 Анализ результатов вычислительного эксперимента 100
4.1.4 Комплекс программ для реализации численного эксперимента 107
4.2 Создание имитационной модели СА 110
4.2.1 Постановка численного эксперимента 111
4.2.2 Описание реализации численного эксперимента 115
4.2.3 Анализ результатов вычислительного эксперимента 117

Стр.

4.2.4 Программная реализация системы имитационного моделирования 124
4.3 Реализация комплекса программ для агрегированной обработки распределенных смешанных данных в центре анализа и обработки данных инициативы GRADLCI 130
4.4 Результаты и выводы по главе 4 140

Заключение 142

Список литературы 144

Список рисунков 160

Список таблиц 163

Приложение А. Используемые в работе обозначения 165
А.1 Нотация записи задач теории расписаний 165
А.2 Нотация диаграмм Сущность - Связь 168

Приложение Б. Свидетельства о регистрации ПО ЭВМ 170
Б.1 Свидетельство о регистрации ПО ЭВМ №2021680467 170
Б.2 Свидетельство о регистрации ПО ЭВМ №2021681570 171
Б.3 Свидетельство о регистрации ПО ЭВМ №2022610178 172

Приложение В. Акты о внедрении 173

Введение

Актуальность темы. Усиление цифровизации общества в представлении широкого круга исследователей связывается с социально-экономическим феноменом больших данных и вызванного им интенсивного развития прикладных информационных систем различного назначения, таких как системы электронной коммерции, автоматизированные системы управления различными процессами, системы компьютерного моделирования, системы автоматизации (научных исследований, проектирования, производства) и др.

Сопутствующая этому развитию эволюция информационных технологий приводит, с одной стороны, к увеличению количества данных, обрабатываемых такими системами, и, как следствие, к ужесточению требований, предъявляемых к их работе, увеличению вычислительной нагрузки как в плане количества обрабатываемых информационных единиц, так и в плане роста числа выполняемых внутренних задач. Работа таких систем требует эффективной диспетчеризации, которая может быть достигнута за счёт использования моделей и алгоритмов такой области исследования операций, как теории расписаний. С другой стороны, наблюдается тренд на дифференциацию систем обработки больших данных, что позволяет выделять новые классы таких систем и разрабатывать для них более эффективные специализированные подходы к обработке информации.

Таким образом, актуальным становится развитие математического аппарата теории расписаний, создание эффективных алгоритмов диспетчеризации и их реализации в виде комплексов проблемно-ориентированных программ.

Степень разработанности темы. Научный поиск моделей и алгоритмов составления расписаний опирается прежде всего на формальное описание свойств ресурсов, их типологию и ограничений на количество доступных ресурсов в системе, раскрываемых в работах таких отечественных учёных, как В.Н. Бурков, Е.Р. Гафаров, В.С. Гордон, А.А. Лазарев, В.С. Танаев, Я.М. Шафранский и др., в частности, в работах воронежских учёных: Т.В. Азарновой, А.Я. Асниной, С.А. Баркалова, Ю.В. Бондаренко, Т.М. Леденевой, М.Г. Матвеева и др. Среди зарубежных учёных можно выделить таких как: Я. Блажевич, П. Брукер, Г. Валигора, Д.С. Джонсон, Г. Кендалл и др.

Теоретической основой исследования послужили модели и алгоритмы составления расписаний для приборов и сетевого планирования. Выдающиеся ре-

зультаты в области составления расписаний для приборов с дискретными ограничениями на ресурсы (таких как обслуживание на одном приборе, и различных постановок задач цеха для нескольких приборов) получены таким авторами, как М.Р. Гэри, А.Х.Г. Ринной Кан, Э.Г. Кофман-мл, Р.Дж. Деннинг и др. Составление расписаний для приборов в задачах с непрерывными ресурсными ограничениями рассматривалось в работах Я. Юзефовски, Р.М. Карпа, А. Яняка, Е. Новицки и др. Различные модификации модели минимизации времени выполнения проекта с учетом ограничения на ресурсы (обозначаемой в литературе как RCPSP), а также модели выравнивания по ресурсам (RLP), поиска компромисса между временем и затратами и вероятностные модели были разработаны такими авторами как В. де Рейк, С. Барум, А. Шпрехер и др.

В основу разработки алгоритмов в данном исследовании положен анализ генетических алгоритмов, алгоритмов муравьиных колоний, методов динамического перебора, а так же широко применяемых в системных планировщиках ИС робастных эвристик, таких как простые правила диспетчеризации (SPT, STF и др.) или алгоритм Round-robin (А.К. Гупта, Н. Арора, Ф. Алаа, М.М. Зулиха и др.).

Однако, несмотря на несомненную значимость проанализированных работ, изложенные в них результаты, хотя и являются достаточными для решения широкого круга проблем, требуют расширения математического аппарата моделями и алгоритмами, обеспечивающими построение расписаний с учётом специфики информационных ресурсов. Существенным отличием таких моделей и алгоритмов должно стать такое современное требование работы с большими данными, как обеспечение баланса между эффективностью расписания и временем выполнения расчета. В этой связи разработка математических моделей, формирование эффективных алгоритмов и проблемно-ориентированных комплексов программ для построения расписаний в системах с ограничениями доступа к распределенным информационным ресурсам является важной научной и прикладной проблемой.

Обьект исследования. Процессы формирования расписаний в системах с несколькими приборами и дополнительными возобновляемыми ресурсами, ограниченными в терминах качественной доступности применительно к классу распределённых систем хранения и обработки данных, использующих удалённые информационных ресурсы.

Предмет исследования. Дискретные и дискретно-непрерывные математические модели, методы, вычислительные алгоритмы, предназначенные для со-

ставления расписаний в системах обработки данных с несколькими приборами и дополнительными ресурсами.

Целью исследования является разработка и теоретическое обоснование новых математических моделей, быстрых вычислительных алгоритмов для составления расписаний в распределенных системах обработки данных с агрегированным поиском, основанных на учёте ограничений на дополнительные информационные ресурсы, а также создание комплексов параллельных программ для оперативного решения задач. Для достижения поставленной цели необходимо решить следующие **задачи**:

- провести анализ процессов формирования расписания в информационных системах с агрегированным доступом к данным, обзор существующих математических моделей и алгоритмов теории расписаний с целью выявления основным проблем применения аппарата теории расписаний в прикладных информационных системах и обоснование актуальности построения моделей, предусматривающих ограничения на дополнительные информационные ресурсы;
- разработать подход к математическому моделированию распределённых систем обработки данных с несколькими приборами и дополнительными информационными ресурсами;
- сформулировать постановку математической задачи поиска оптимального расписания в исследуемом классе систем;
- построить семейство моделей составления расписаний для нескольких приборов с ограничениями на доступность ресурсов;
- разработать алгоритмы и численные методы решения задач составления расписаний в многоприборных системах с ресурсами, обладающими ограниченной качественной доступностью;
- реализовать разработанные математические модели и алгоритмы в виде комплексов проблемно-ориентированных программ для проведения комплексного исследования проблем составления расписаний в распределённых систем обработки данных с несколькими приборами и дополнительными информационными ресурсами.

Соответствие диссертации Паспорту научной специальности. Исследование соответствует формуле специальности 05.13.18 «Математическое моделирование, численные методы и комплексы программ» по следующим пунктам: п. 4 «Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного экспе-

римента», п. 5 «Комплексные исследования научных и технических проблем с применением современной технологии математического моделирования и вычислительного эксперимента», п. 8 «Разработка систем компьютерного и имитационного моделирования».

Материалы и методы исследования. Для решения задач, поставленных в работе, применялись фундаментальные положения ряда научных направлений: теории расписаний, теории массового обслуживания, математических методов исследования операций, статистики, теории вероятностей, теории алгоритмов.

Комплексы программ для создания имитационных моделей, проведения численных экспериментов и решения задач диспетчеризации в системе доступа и распределённой обработки данных GRADLCI реализованы на языке Python 3 для работы в дистрибутивах операционной системы Linux или MacOS с использованием библиотек plotly и matplotlib для визуализации результатов и библиотек subprocess и multiprocessing для параллельных асинхронных вызовов в многопроцессорной вычислительной системе для параллельной реализации разработанных алгоритмов.

В качестве **методологической основы** использован системно-аналитический подход с фокусом на структурном, функциональном и динамическом аспектах, численные методы, методы математического моделирования и программирования. В качестве вспомогательных методологических средств использованы контент-анализ, методы интеллектуальной схемотехники, сравнение, систематизация. Работа основывается на принципах системности, комплексности, детерминированности.

В диссертации получены следующие основные результаты, характеризующиеся **научной новизной**:

1. Предложено формальное описание класса распределенных систем обработки данных с агрегированным поиском, отличающихся наличием дополнительных информационных ресурсов ограниченных в терминах качественной доступности. Сформулирована математическая задача поиска оптимального расписания в рассмотренном классе систем.
2. Разработаны оптимизационные дискретная и дискретно-непрерывная математические модели составления расписаний для нескольких приборов, отличающиеся учетом дополнительных ограничений на качественную доступность ресурсов и функцией цели, заключающейся в минимизации времени выполнения агрегированной работы. Проведено исследование существования аналитиче-

ского решения для случаев дискретных ограничений на ресурсы и различных видов функции расхода ресурса для модели с непрерывными ограничениями на ресурсы.

3. Сформированы новые алгоритмы решения задач составления расписаний в многоприборных системах с ресурсами, обладающими ограниченной качественной доступностью, основанные на приоритето-порождающих функционалах для различных соотношений числа приборов к числу ресурсов. Для предложенных алгоритмов исследована асимптотическая сходимость для худшего случая выполнения.
4. Разработаны комплексы программ для направленного численного эксперимента по исследованию свойств разработанных алгоритмов, имитационного моделирования поведения систем агрегации при различных заданных параметрах.
5. Разработанные численные методы и алгоритмы реализованы в виде комплекса программ для работы с пользовательскими заявками и диспетчеризации задач в центре сбора и анализа данных экспериментальной астрофизики частиц GRADLCI.

Практическая значимость заключается в построении комплекса дискретных и дискретно-непрерывных математических моделей и алгоритмов, позволяющих сформировать расписание обслуживания информационных запросов в широком классе систем с дополнительными информационными ресурсами за практически приемлемое время.

Разработаны комплексы программ на языке Python 3 для построения имитационных моделей и проведения численных экспериментов, использующие асинхронные распаралеленные вызовы, реализованные с использованием библиотек subprocess и multiprocessing, что определило такие отличительные свойства разработанных программ, как оптимизированное выполнение высоконагруженных вычислений, улучшенная масштабируемость, ускоренный по сравнению с последовательным выполнением отклик пользовательских интерфейсов и более эффективное использование системных ресурсов. Разработанные программные комплексы позволяют: создавать и конфигурировать имитационные модели распределённых систем обработки данных с агрегированных доступом к удалённым информационным ресурсам; осуществлять моделирование различных сценариев составления расписаний в многоприборных распределённых системах с ограниченными в терминах качественной доступности информационными ресурсами; осуществлять за практически приемлемое (полиномиальное либо линейно-

логарифмическое) время составление расписаний обработки пользовательских запросов на среднесрочном временном интервале для пользовательских запросов, распределённых согласно экспоненциальному распределению с периодически изменяющимся параметром интенсивности потока заявок. Получены 3 свидетельства о регистрации ПО ЭВМ [12—14].

Разработанные алгоритмы и модели нашли своё применение при создании автором системы доступа к данным экспериментов астрофизики частиц GRADLCI, используемой профессиональными исследователями в области астрофизики частиц. Полученные результаты включены в программу учебных курсов «Теория игр и исследование операций», «Имитационное моделирование», «Имитационное моделирование в задачах машинного обучения», «Методы оптимизации», преподаваемых на факультете прикладной математики, информатики и механики ФГБОУ ВО «Воронежский государственный университет».

Основные результаты и положения, выносимые на защиту:

1. Формальный аппарат описания класса распределенных систем обработки данных с агрегированным поиском, отличающихся наличием дополнительных информационных ресурсов, позволяет выделить характерные особенности процесса составления расписаний и сформировать базовые составляющие для построения комплекса моделей для построения эффективных расписаний в данном классе систем.
2. Комплекс математических моделей составления расписаний для нескольких приборов, включающих дискретную и дискретно-непрерывную оптимизационные модели составления расписаний в многоприборных системах с ограничениями на дополнительные информационные ресурсы. В моделях учтены ограничения на количество соединений и скорость доступа к удаленным информационным ресурсам. Комплекс предназначен для формирования оптимальных расписаний по одному из следующих критериев: минимизации длины расписания, минимизации суммы времён обслуживания задач и минимизации суммарного времени выполнения агрегированных работ. Математическое исследование существования аналитического решения для случаев дискретных ограничений на ресурсы и различных видов функции расхода ресурса для модели с непрерывными ограничениями на ресурсы, позволяет обосновать использование численных алгоритмов решения.
3. Алгоритмы численного решения задач составления расписаний в многоприборных системах с ресурсами, обладающими ограниченной качественной до-

ступностью, основанные на приоритето-порождающих функционалах для случаев числа приборов равного числу ресурсов и числа приборов меньшего числа ресурсов, делают возможным отыскание эффективного решения за практически приемлемое время.

4. Комплексы программ, включающие в себя: систему имитационного моделирования, предназначенную для создания имитационных моделей процессов формирования расписаний в распределённых системах обработки данных с агрегированным доступом к информационным ресурсам для различных комбинаций параметров; комплекс программ для реализации численных экспериментов, предназначенный для моделирования различных сценариев составления расписаний в многоприборных распределенных системах с агрегированным поиском; для работы с пользовательскими заявками и диспетчеризации задач в центре сбора и анализа данных экспериментальной астрофизики частиц GRADLCI.

Достоверность полученных результатов основывается на строгих формулировках и доказательствах и подтверждается проведёнными вычислительными экспериментами.

Апробация работы. Основные выводы и теоретические положения диссертационного исследования были представлены в докладах на научных мероприятиях международного и всероссийского уровня, в числе которых: The 8th International Conference “Distributed Computing and Grid-technologies in Science and Education” (Дубна, 2018); School for Astroparticle Physics Obertrubach-Bärnfels (Obertrubach, Germany, 2018); Matter and the Universe Days (Hamburg, Germany, 2019); Workshop “Big Data Science in Astroparticle Research” (Aachen, Germany, 2019, 2020); 6th KSETA Plenary workshop, (Durbach, Germany, 2019, 2020); 19th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (Saas Fee, Switzerland, 2019); DPG conference (Aachen-Berlin, Germany, 2019, 2021); International workshop “Data life cycle in physics” (Карлсруэ-Иркутск-Москва, 2018, 2019, 2020); Sparse Digital Radio Arrays online workshop (Москва, 2019); International Conference on Computer Simulation in Physics and beyond (Москва, 2020); The XXIV International Scientific Conference of Young Scientists and Specialists (Дубна, 2020); 13-я Международная конференция «Интеллектуализация обработки информации» (Москва, 2020); 5th International Workshop on Deep Learning in Computational Physics (Москва, 2021); 37th International Cosmic Ray Conference-2021 (Berlin, Germany, 2021); Международная научная

конференция «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2021); семинар лаборатории №68 «Теории расписаний и дискретной оптимизации» ИПУ РАН; а также рабочих совещаниях коллаборации GRADLCI/APPDS и семинарах Института Астрофизики Частиц Технологического Института Карлсруэ.

Работа была выполнена при поддержке грантов РНФ № 18-41-06003 и фонда Гельмгольца № HRSF-0027.

Публикации. Основные результаты по теме диссертации изложены в 15 статьях [1—15], 11 из которых проиндексированы базами Scopus и Web of Science [1—11] и 3 являются свидетельствами о регистрации ПО ЭВМ [12—14].

Личный вклад. Все результаты, изложенные в диссертации, получены автором самостоятельно. Из совместных работ в диссертацию включены только результаты, полученные лично автором.

Объём и структура работы.

Диссертация состоит из введения, четырёх глав, заключения, библиографического списка и приложений. Полный объём диссертации составляет 175 страниц, включая 44 рисунка и 22 таблицы. Список литературы содержит 159 наименований.

Глава 1. Проблемы составления расписаний в многолинейных системах обработки информации с ограничениями на ресурсы

В первой главе обосновывается актуальность диссертационного исследования, введены основные понятия и категории, рассмотрена постановка задачи составления расписания агрегированной обработки данных и её отличие от задач RCPSP и MRCPSP, описаны ее основные свойства и особенности. Произведён анализ существующих моделей и алгоритмов составления расписаний минимальной длины с учётом ограничения на ресурсы, известных из публикаций исследований отечественных и зарубежных специалистов. Сформирована теоретико-методологическая база диссертационной работы.

1.1 Проблемы теории расписаний: классификация, нотация, основные определения, примеры

Задачи упорядочения во времени различных целенаправленных действий с учетом целевой функции и системных ограничений часто возникают в таких областях социально-экономической активности, как: планирование производства (составление календарных планов производства, оптимизация труда и изготовления продукции), составление расписаний для общественного транспорта: поездов, городского транспорта, самолётов, речных судов и т. д.; организации работы сотрудников (планирование дежурств и распределения задач по исполнителям); составление расписаний занятий в учебных заведениях, организация обработки данных и передачи сообщений телекоммуникационных сетях; планирование проведения мероприятий; автоматизация производственно-экономической деятельности. Распространённость таких задач определяет их существенное общественное значение и важность выделения отдельного научного направления, занимающегося исследованием данных задач с использованием аппарата математического моделирования и численного эксперимента и поиском эффективных научно-обоснованных подходов к их решению.

Данный раздел исследования операций (области математической науки, занимающейся разработкой теоретических основ и научно обоснованных методов разработки количественно обоснованных рекомендаций по принятию решений) носит название теория расписаний. Данный термин был впервые предло-

жен в 1956 г. в работе Р. Беллмана «Математические аспекты теории расписаний» (Mathematical aspects of scheduling theory) [16].

В то же время основополагающей работой, с которой принято вести начало данного научного направления, принято считать классическую работу [17] Г.Л. Ганта, опубликованную в 1903 г., где были впервые предложены т. н. диаграммы Ганта для визуализации результатов планирования организационной деятельности. Результаты данной работы были позже расширены Гантом в ряде работ, посвящённых исследованию и оптимизации управления на промышленных предприятиях, таких как заводы, производящие хлопчатобумажные ткани и на промышленных верфях.

Пример диаграммы Ганта приведен на рисунке 1.1. На оси ординат диаграммы перечислены активности, которые необходимо выполнить (работы), на оси абсцисс представлены временные отрезки. Планируемое время выполнения задачи показано фиолетовым цветом: выполнение каждой j-ой задачи начинается в некий момент времени s_j, называемый временем начала работы, происходит в течение временного периода длительностью p_j, называемого временем выполнения, и завершается в момент времени C_j, называемого временем окончания работы. Определение данных обозначений будет приведено нами далее в этой части, а также в кратком виде представлено в Приложении А.

В общем случае задача теории расписаний $\Pi(I, A)$ может быть сформулирована следующим образом: для заданного множества задач, приборов (или исполнителей), необходимых задачам ресурсов и предложенных системных ограничений — т. н. начальных условий, обозначаемых через I, требуется выполнить для каждой задачи назначение A времени начала выполнения, исполнителя или исполнителей (прибора или приборов в случае составления расписаний для приборов) и ресурсов таким образом, чтобы выполнялись наложенные на систему ограничения. Способы формальной записи некоторых задач теории расписаний будут рассмотрены нами в части 1.1.3.

Определение* 1.** *Расписание, которое соответствует данным условиям, называют* ***допустимым *для задачи* $\Pi(I, A)$.

Определение* 2.** *На множестве допустимых расписаний* $\tilde{A}$ *может быть задан функционал цели* F_{aim} *такой, что существует некоторое расписание* A^* *такое, что достигается экстремум функционала* F_{aim}. *Такое решение* A^* *задачи* $\Pi(I, A)$ *называется* ***оптимальным расписанием *для данной задачи.*

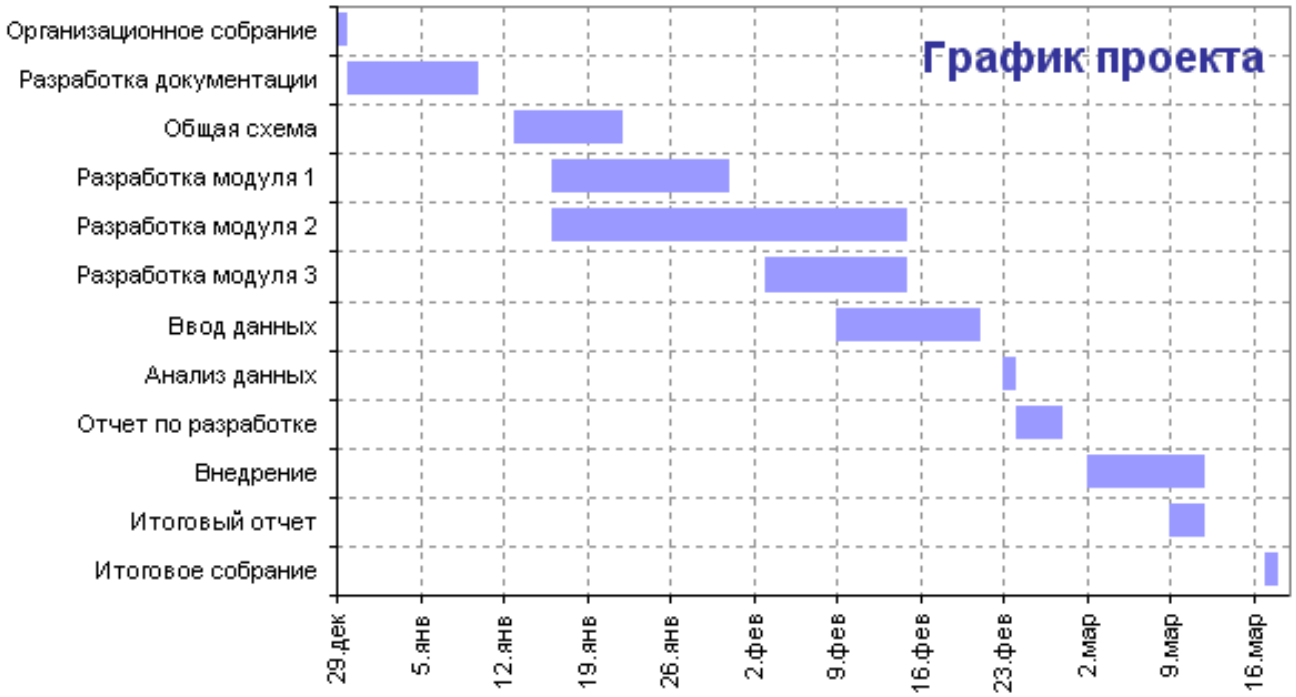


Рисунок 1.1 — Пример диаграммы Ганта для задачи календарного планирования проекта.

1.1.1 Подходы к классификации задач ТР

На сегодняшний день выделяют [18—24] следующие разделы теории расписаний:

– **Построение расписаний для приборов (Machine Scheduling)** — требуется составить расписание обработки заданий одним или более прибором (машиной, процессором), как правило, с одновременным обслуживанием каждого требования не более, чем на одном приборе. Среди задач, изучаемых в этом разделе, выделяют задачи:
  – **Обслуживания одним прибором** [25];
  – **Обслуживания несколькими приборами** [26—28] (существуют различные типы таких приборов, см. часть 1.1.4 «Нотация»);
  – **Задачи цеха (shop scheduling)**, среди которых выделяют: **общую задачу цеха (general shop problem)** [29], **задачу планирования для поточной линии (flow shop problem)** [30], **задачу планирования для рабочего цеха (job shop problem)** [31], **задачу планирования для открытой линии (open shop problem)** [32];
  – **Задачи обработки сгруппированных работ (batching)** [33; 34];

  - **Задачи обработки работ в системах жёсткого и мягкого реального времени (hard real-time system scheduling problem, soft real-time system scheduling problem)** и др. [35];
- **Построение расписаний для проектов (Project scheduling)** [36; 37] — требуется спланировать выполнение совокупности действий (задач), связанных между собой отношениями предшествования, требующих для своего выполнения ресурсов системы, одного или нескольких исполнителей.
- **Составление временны́х таблиц (Time Tabling)** [38] — задачи, изучаемые в этом разделе, связаны с составлением расписаний, учитывающих все ограничения задачи в виде таблиц и возникают при планировании рабочего времени персонала, учащихся, при назначении дат и времен рабочих встреч и т. д. (например, составление расписания занятий в школе или ВУЗе. Задачи составления временных таблиц зачастую могут быть сведены в задачам построения расписаний проектов;
- **Стохастические задачи теории расписаний** [39] — задачи, в которых для некоторых параметров известны не точные значения, но их вероятностные характеристики;
- **Задачи планирования движения транспорта** [40—42], такие как:
  - **Составление расписаний движения транспортных средств (Transport Scheduling)**;
  - **Построение циклических расписаний движения транспортных средств (Vehicle Routing)**;
  - **Организация доставки товаров (Shop-Floor Scheduling)**;
  - **Составление расписания обработки грузового транспорта на станциях отгрузки (Cross dock scheduling)** и др.

Также задачи можно различать по типу поступления информации: имеют место либо непосредственное динамическое поступление требований и их обработка, либо решение статически заданных требований, по типу целевой функции (возможные значения целевых функций будут указаны нами далее), по наличию определенных дополнительных условий, определенных в постановке задачи, таких как: ограничения ресурсов, ограничения предшествования, наличие прерываний обработки требований, наличие директивных сроков выполнения задач и т. д. Возможные виды таких ограничений будут перечислены нами далее при описании нотации записи задач теории расписаний.

По типу искомого решения выделяют [19]:

– *Задачи упорядочивания.* В этих задачах уже задано распределение работ по исполнителям, а также определены все параметры работ (продолжительность выполнения, время поступления т.д.). Необходимо составить расписание (или порядок) выполнения работ каждым исполнителем;
– *Задачи согласования.* Основное внимание в этих задачах уделяется выбору продолжительности выполнения работ, времени поступления и другим параметрам;
– *Задачи распределения* подразумевают поиск оптимального распределения работ по исполнителям.

Также задачи ТР, как входящие в подмножество множества задач комбинаторного поиска, могут разделяться на [18]:
– *Задачи оптимизации*. Решение такой задачи состоит в нахождении экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств [21];
– *Задачи выбора*. Для таких задач может существовать только два возможных решения: «да» или «нет».

При этом для любой задачи оптимизации Ψ_o всегда можно составить аналогичную задачу выбора Ψ, отвечающую на вопрос о существовании решения задачи Ψ_o со значением функции цели, не превышающим (или не меньшим, чем) некоторое дополнительно заданное пороговое значение b.

1.1.2 Разрешимость задач теории расписаний и их алгоритмическая сложность

Начиная с 70-х гг 20-го века, в связи с выходом работ таких авторов как Р.М. Карп [43; 44], Я.К. Ленстра и А.Х.Г. Ринной Кан [45], А. В. Ахо [46], Дж. Д. Ульман [46—48], М. Р. Гэри и Д. С. Джонсон [49; 50] и др., посвящённых теории сложности решения задач исследования операций, увеличивается интерес исследователей к исследованию класса сложности (P, псевдополиномиальные, NP, APX, PTAS, FPTAS) задачи получению более точных оценок времени выполнения алгоритмов и гарантий точности решения. На сегодняшний день существует большое количество обзоров литературы, созданных такими авторами как В.С. Танаев и соавт. [51—53], К. Янсен и соавт. [54], П. Брукер [55] и соавт., С. Кунст [55; 56] и соавт., и др.

Поскольку ТР является областью прикладной математики, крайне важным для решения её задач является получение не только по возможности точного, но

и своевременного решения, что понимается как решение, полученное за время не хуже полиномиального. Однако, большинство прикладных задач составления расписаний являются NP-трудными в строгом смысле, что вынуждает исследователей наряду с применением методов полного перебора, таких как метод ветвей и границ, использовать также релаксацию ограничений задачи либо построение приближенных вычислительных алгоритмов, основанных на определенных эвристических догадках о свойствах искомых решений. Эвристические алгоритмы, для которых возможно оценить погрешность решения задачи, называются приближёнными [57; 58]. Приближенные алгоритмы могут гарантировать как абсолютную [59], так и относительную погрешность [60]. Также выделим алгоритмы, допускающие существование вполне полиномиальной (FPTAS) [61] или полиномиальной (PTAS) [62] аппроксимационных схем, представленные в работах таких авторов как Ковалёв [63], Мастролилли [64], Келлерер и соавт. [65], Вёгингер [66] и др.

Характерной областью применения ТР является поиск моделей и алгоритмов для обеспечения эффективного функционирования систем управления большими данными. Так, активные исследования ведутся в направлении применения ТР в ГРИД-системах такими авторами, как М. Мика [67], Г. Валигора [67], Я. Вегларц [68]. Важными характеристиками моделей и методов построения расписаний для крупных информационных систем является скорость расчетов и робастность. Поэтому актуальным является поиск алгоритмов, обеспечивающих баланс между такими характеристиками решения как оптимальность и время выполнения расчёта, и при этом учитывающих характер ресурсных ограничений системы.

Среди алгоритмов, используемых для решения таких задач, можно выделить такие алгоритмы динамического перебора как динамическое программирование и метод ветвей и границ, а также эвристические и приближенные алгоритмы, среди которых: генетические алгоритмы, алгоритмы муравьиных колоний, поиск с запретами, алгоритмы имитации отжига и др., а так же широко применяемых в системных планировщиках ИС робастных эвристик, таких как простые правила диспетчеризации (SPT, STF, LILO, STPT и др.) или алгоритм Round-robin (А.К. Гупта [69], Н. Арора [70], Ф. Алаа [71] и др.).

Важно отметить, что принадлежность конкретной задачи ТР к классу P или NP-трудных задач не определяется способом записи задачи [18]. В то время как различные схемы кодирования задачи могут влиять на сложность полинома для полиномиальных алгоритмов или сложность экспоненциального алгоритма для

NP-трудных задач, задачи, которые являются NP-трудными в сильном смысле, остаются NP-трудными даже в случае использования унарной схемы кодирования [72].

1.1.3 Постановка задач составления расписаний

В данной части опишем в формальном виде постановку задач теории расписаний с акцентом на задачи построения расписаний для приборов, поскольку именно такие задачи будут изучаться нами далее. При этом постановку будем излагать в следующем порядке: определим необходимые элементы и их свойства, затем обозначим возможные ограничения, и далее рассмотрим возможные целевые функционалы.

Запись условий в задачах ТР

В общем случае задачи теории расписаний, изучаемые в рамках данной работы, могут быть охарактеризованы следующими множествами:

- Множеством работ $J = \{J_1, J_2, \ldots J_N\}$, где N — количество работ, находящихся в системе. Если каждая работа включает в себя выполнение нескольких действий, то множества таких действий могут обозначаться как множество задач T либо как множество операций O (такая терминология характерна для задач цеха), и элементы этих множеств являются составными частями работы. Если работа J_j состоит из одного действия, её всё равно можно формально записывать как работу, которой соответствует множество задач T_j, состоящее из одного элемента;
- Опционально — множеством задач $T_j = \{T_{j1}, T_{j2}, \ldots\}$, если одна работа включает в себя выполнение нескольких действий;
- Множеством приборов $P = \{P_1, P_2, \ldots, _m\}$, где m — это количество доступных приборов. Могут существовать различные типы приборов, и в зависимости от этого будет различаться запись задачи. Подробнее этот момент будет описан в части 1.1.4.
- Множеством ресурсов $R = \{R_1, R_2, \ldots, R_s\}$, где s — различные типы ресурсов R. Классификация видов ресурсов, способы постановки задач с ограничениями на ресурсы и основные известные результаты для таких задач будут проанализированы нами в части 1.3.

Найти решение задачи теории расписаний означает выполнить для каждой работы J_j из (либо для каждой задачи T_{ji} из T_j) назначение прибора из P и (если требуется) одного или нескольких ресурсов из R и времени начала обработки

s_j (s_{ji}), таким образом, чтобы все задачи были выполнены с соблюдением заданных ограничений (см. 1.1.3).

Рассмотрим более подробно свойства, которыми может обладать задача и то, как эти свойства экстраполируются на работы, как на объекты более высокого уровня абстракции.

Необходимые свойства задач T_{ji}:

1. Моменты поступления r_{ji} (release dates). В зависимости от постановки задачи, все r_{ji} могут быть как равны некоторой константе (в т. ч. нулю), так и быть случайными величинами из $\mathbb{Z}$ либо из $\mathbb{R}$;
2. Моменты начала обработки s_{ji}. Являются одной из искомых величин при решении задач теории расписаний. По объективным причинам, всегда должно выполняться $s_{ji} \geq r_{ji}$;
3. Времена обработки p_{ji}. Могут как определяться равными некой константе, так и различаться в зависимости от свойств прибора, на который будет направлена задача, или необходимых для выполнения задачи ресурсов (такие задачи будут исследованы нами в данной работе), а также быть случайными величинами, что характерно для задач стохастической теории расписаний;
4. Момент завершения обработки C_{ji} задачи T_{ji}.

Проиллюстрируем данные свойства с помощью следующей иллюстрации (рисунок 1.2).

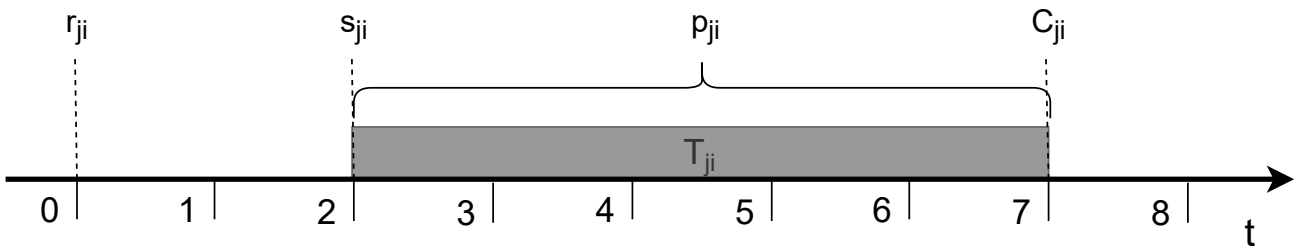


Рисунок 1.2 — Основные свойства задачи T_{ji}.

Для каждой задачи в системе определено время поступления в систему r_{ji}, в данном случае задача T_{ji} поступила в систему в момент времени 0. После того, как задача поступила в систему, ей могут быть выделены прибор и ресурсы для обслуживания и назначен момент начала обслуживания s_{ji}. Обслуживание задачи составляет некоторую положительную величину $p_{ji'}$ значение которой зависит от дополнительных условий задачи. Момент окончания обслуживания обозначается через C_{ji} и равен $s_{ji} + p_{ji}$. Очевидно, что $\forall C_{ji} \geq r_{ji}$.

Для работы J_j, состоящей из нескольких задач T_{ji}, составляющих множество T_j, данные свойства могут быть экстраполированы следующим образом:

1. Для всех задач $T_{ji'}$, составляющих работу J_j, моменты поступления задач в систему равны моменту поступления работы в систему: $r_j = r_{ji} \quad \forall i$;
2. Время начала выполнения работы J_j совпадает с самым ранним из моментов начала обработки задач из множества $T_j : s_j = \min_i \{s_{ji}\}$;
3. Момент окончания обслуживания J_j совпадает с самым поздним из моментов окончания обработки задач из множества $T_j : C_j = \max \{C_{j1}\}$;
4. Таким образом, общее время обслуживания p_j задачи может быть выражено как $p_j = C_j - s_j$. При обслуживании задач по очереди на одном приборе p_j может быть оценено как $p_j = \sum_i p_{ji}$, при параллельном обслуживании на нескольких приборах p_j можно оценить (с возможной коррекцией, связанной с типом прибора), как $p_j \geq \max_i p_{j1}$.Таким образом, время обслуживания p_j работы J_j находится в интервале $\left[\max_i p_{ji}; \sum_i p_{ji}\right]$.

Также задача T_{ji} может обладать дополнительными свойствами, связанными с накладываемыми постановкой ограничениями:

– Директивный срок завершения обслуживания d_{ji}, определяющий желательный момент времени, к которому обслуживание задачи T_{ji} должно быть завершено. В случае превышения данного срока назначается штраф, изменяющий значение целевого функционала задачи;
– Предельный срок завершения обслуживания $\hat{d}_{ji}$, который не может быть нарушен ни при каких условиях. Расписание, в котором нарушен хотя бы один из предельных сроков $\hat{d}_{ji}$, является недопустимым;
– Приоритет выполнения (вес задачи) w_{ji}, определяющий относительную срочность выполнения задачи. В главе 2 нами будет изучена возможность назначения задачам приоритетов с использованием метода построения приоритетопорождающих функционалов;
– Необходимость ресурсов R_s для выполнения задачи. Возможные виды ресурсов и способы записи такого свойства задач систематизированы на основе анализа литературы в 1.3.

Ограничения в задачах теории расписаний

В рамках классической теории расписания существует два широко применяемых ограничительных принципа, которых мы будем придерживаться в ходе данного исследования:

1. Каждая задача должна обрабатываться не более чем одним прибором одновременно;
2. Каждый прибор может обрабатывать единовременно не более одной задачи.

Кроме того, из определения свойства задач очевидно, что:

– Выполнение задачи T_{ji} не может быть начато раньше времени её поступления r_{ji}. Говорят, что в момент времени $t \geq r_{ji}$ задача T_{ji} является доступной. Иначе задача T_{j1} недоступна для включения её в расписание;
– Если для выполнения задачи требуются некие ресурсы, обслуживание этой задачи возможно только тогда, когда нужные ресурсы доступны в течение всего времени выполнения в нужном количестве.

Обслуживание заданий в системе возможно как с разрешенными прерываниями (тогда подразумевается, что при определенных условиях допустимо прервать обработку текущей задачи, и продолжить её позже), так и строго без прерываний. В случае обслуживания без прерываний расписание не будет допустимым, если в нём есть прерывание обслуживания по крайней мере одного задания. При обслуживании с прерываниями число прерываний должно быть счётным [18].

Задачи, входящие в одну работу, могут быть связаны между собой ограничениями предшествования, что значит, что одна из задач должна быть обслужена строго раньше другой. Например, если задача T_{ji} должна быть обслужена раньше задачи $T_{ji'}$, пишут: $T_{ji} < T_{ji'}$.

Если расписания удовлетворяет всем изложенным условиям, оно называется допустимым.

Запись целевых функций в задачах ТР

Функции цели в задачах теории расписаний могут быть заданы следующим образом [19; 22; 23]:

– Через суммарные критерии оптимизации. Минимизируется сумма некоторых величин, возможно с использованием весовых коэффициентов или усреднением. Например: $\sum_{j=1}^{n} C_j$ — сумма времен завершения обслуживания работ; $\sum_{j=1}^{n} w_j C_j$ — взвешенная сумма времен завершения обслуживания работ; $\frac{1}{n}\sum_{j=1}^{n} F_j$ — среднее время, проводимое работой в системе, где $F_i = C_j - r_j$ — это время пребывания в системе конкретной работы;

– Через minmax критерии оптимизации. Минимизируется максимальное значение некоторой величины. Например: $C_{\max} = \max\limits_{j} C_j$ — максимальная длина расписания; $L_{\max} = \max\limits_{j=1}^{n} L_j$ — максимальное запаздывание (используется в системах с директивными сроками), где $L_j = C_j - d_j$;
– Через множество критериев оптимизации (многокритериальная оптимизация). «Некоторые многокритериальные задачи могут быть преобразованы (сведены) к однокритериальным. Другие же могут подразумевать нахождение „Парето-оптимального решения“ или нахождение оптимума иерархической целевой функции» [19].

Критерии оптимизации, используемые в данном исследовании, будут приведены и обоснованы в главе 2. Таблица известных из литературы возможных критериев оптимизации находится в приложении А.1, таблица 20.

1.1.4 Нотация, используемая при записи задач ТР

Будем изучать задачи, связанные с ограничениями ресурсов, в первую очередь, задачи построения расписаний для приборов и планирования расписаний для проектов.

Для записи задач данных типов в их дискретной форме в литературе принято использовать т.н. $\alpha|\beta|\gamma$-нотацию (также называемую «нотацией Грехема»), впервые опубликованную в 1979 году в классической статье [73] Р.Л. Грэхэма, Э. Лоулера, Дж. Ленстры и А.Х.Г. Ринной-Кана, исследующей задачи построения расписаний для приборов. Позднее эта нотация расширялась, в частности, большой вклад в её расширение для задач построения расписаний для приборов с ограничениями на ресурсы был внесён в 1983 году Я. Блажевичем и соавторами в работе [74]. Позднее, при расширений класса задач построения расписаний для проектов данная нотация была адаптирована и частично дополнена [75; 76].

Нотация состоит из полей α, β и γ, с помощью которых описываются занятые в задаче приборы, переменные и ограничения модели и целевая функция задачи, соответственно.

При этом, первое поле α, описывающее используемые в задаче приборы, состоит из двух частей α_1 и α_2, где через параметр α_1 описывается тип прибора, а через параметр α_2 — количество приборов в задаче, если релевантно. Возможные значения параметра α_1 принадлежат множеству $\{\emptyset, P, J, F, O, Q, R\}$, где

– $\emptyset$ обозначает отсутствие информации о типе прибора. В таком случае подразумевается расчёт расписания для одноприборной системы. Если $\alpha_1 = \emptyset$, т. е. не

указан, то параметр α_2 также не указывается. Иначе, имеется в виду обслуживание заданий в системе параллельными приборами. Различают следующие виды параллельных приборов: идентичные, с различной производительностью, несвязанные и выделенные. Определения и обозначения для таких приборов будут даны нами далее.

- P обозначает идентичные приборы, обслуживающие задачи с одинаковой скоростью $v_i = v = \text{Const} \in \mathbb{R}_+$;
- Q используется для обозначения приборов с различной производительностью. Для каждого такого прибора условиями определяется скорость $v_i \in \mathbb{R}_+$, которая понимается как непосредственное свойство такого прибора, не зависящее от обрабатываемых им заданий;
- R используется для обозначения несвязанных (unrelated) приборов. Скорость обслуживания заданий таким прибором зависит от конкретного задания T_{ji};
- В задачах цеха для обслуживания заданий используются т. н. выделенные (dedicated) приборы в зависимости от типа решаемой задачи цеха, задачи с выделенными приборами обозначают следующим образом:
  - J — для задач рабочего цеха;
  - F — для задач планирования поточной линии;
  - O — для задач планирования открытой линии.

Возможные значения параметра $\alpha_2 \in \{\emptyset, m\}$, где $\emptyset$ используется либо если $\alpha_1 = \emptyset$, либо если число приборов в задаче полагается переменным, а $m \in \mathbb{N}$ используется если в задаче используется фиксированное число приборов.

α определяется как $\alpha = \alpha_1 + \alpha_2$. Например, задача планирования поточной линии с m приборами будет иметь $\alpha = 'F' + 'm' = 'Fm'$. Заметим, что существует также расширенная нотация, содержащая поля α_3 и α_4, описывающие топологию вычислительной сети и коммуникационные ограничения между её узлами, соответственно. Данные расширений нотации приведены в таблице 18 приложения А.1.

В поле β задаются ограничения и условия, накладываемые на обслуживание требований. β можно читать как $\beta = \sum_{i=1}^{8} \beta_i$, где:

- Параметр β_1 указывает,допустимо ли в задаче обслуживание с прерываниями. Он имеет значение pmtn, если прерывания допустимы, либо $\emptyset$, если они не допускаются;

– Параметр β_2 указывает на использование в задаче дополнительных ресурсов. Если задача использует ресурсы, то пишут res$\lambda\sigma\varrho$, где λ, σ, ϱ — это количественные характеристики используемых ресурсов. Более подробно эта нотация описана в части 1.3. Если задача не использует дополнительных ресурсов, то поле β_2 остаётся пустым, $\beta_2 = \emptyset$;
– Учёт ограничений предшествования в задаче

$$\beta_3 \in \{\emptyset, \text{prec}, \text{tree}, \text{chain}, \text{uan}, \text{sp} - \text{graph}\},$$

где:
  – $\emptyset$ — отсутствие ограничений предшествования;
  – prec — ограничения предшествования общего вида;
  – tree (также in – tree, out – tree) — ограничения предшествования, заданные деревом. Могут быть даны уточнения о том, является ли дерево «входящим» либо «исходящим»;
  – chain — ограничения предшествования между задачами формирует наборы (цепи) последовательно обслуживаемых заданий;
  – uan (англ. uniconnected activity networks) — ограничения предшествования определены как односвязная сеть активности;
  – sp – graph — ограничения предшествования заданы SP-графом.

  Подробнее об определения упомянутых структур и их свойств заинтересованный читатель может узнать в [77; 78].
– Ограничения на время обслуживания задания вводятся следующим образом. $\beta_5 \in \left\{\emptyset, p_j = \text{const}, p_j \in [\hat{p}, \bar{p}]\right\}$, где $\emptyset$ обозначает отсутствие ограничений, $p_j = \text{const}$ показывает, что для любого задания время обработки ограничено некоторой константой, а $p_j \in [\hat{p}, \bar{p}]$ задает нижнее и верхнее ограничение на время выполнения задания;
– $\beta_6 = \left\{\emptyset, \overline{d}_{ji}\right\}$ описывает предельные сроки завершения обслуживания, которые либо не задаются, либо определены для каждой ji-ой задачи как $\overline{d}_{ji}$;
– $\beta_7 = \left\{\emptyset, n_j \leq k\right\}$ указывает максимальное число задач в работе. Если оно не указано ($\beta_7 = \emptyset$), значит, такое ограничение отсутствует. Иначе для любой -ой работы количество её задач n_j не превышает $k \in \mathbb{K}$;
– Параметр $\beta_8 = \{\emptyset, \text{no} - \text{wait}\}$ описывает ограничение, связанное с наличием буферов в системе и планирование поведения системы в соответствие с этим:

– В случае $\beta_8 = \emptyset$ предполагается, что система имеет буферы неограниченной ёмкости, что позволяет делать паузы при многостадийной обработке требований на нескольких приборах;
– Иначе ($\beta_8 = \text{no} - \text{wait}$) считается, что буферы между приборами имеют ёмкость, равную нулю, и следующий этап обслуживания требования на очередной приборе должен начинаться как только завершилось обслуживание требования на первом приборе.

Поле γ описывает используемый в задаче критерий оптимальности и имеет одно из значений, описанных в п. 1.1.3.

Каждый из представленных пунктов может быть использован как способ классификации задач теории расписаний в дополнение к описанным в ч. 1.1.1. Краткое представление изложенной нотации в табличном виде приведено в приложении А.1.

1.2 Понятие систем агрегации и представление о них, как об объекте моделирования

1.2.1 Понятие системы агрегации

Агрегацией (лат. aggregatio — «присоединение») называют соединение частей в целое [79]. Данное понятие имеет широкое применение в таких областях как информатика, экономика, биология, медицина и инженерия. Системы, компании или компьютерные программы, называемые агрегаторами, осуществляют организацию единого интерфейса доступа к некоторым (как правило, информационным) ресурсам.

***Определение* 3.** *Мы будем называть системой агрегации (или агрегатором) систему, имеющую доступ к ряду изолированных распределенных ресурсов и объединяющую их в логическое целое с единым интерфейсом.*

Обработка запросов пользователей, направленных на агрегатор, может осуществляться как одним прибором, так и несколькими. В общем случае будем считать такую систему многолинейной. Выделение и исследование такого класса систем представляет значимость в связи с развитием обработки больших данных и растущей представленностью таких систем в повседневной жизни.

Помимо приведённых ранее примеров обработки научных данных, можно привести такие СА как:

– Торговые агрегаторы, широко представленные в области электронной коммерции. Это электронные торговые площадки, обеспечивающие одновременное проведение закупок из большого числа доступных удалённых каталогов [80]. Агрегатор рационализирует снабжение благодаря использованию большого числа каталогов, ориентированных по группам покупателей. Модель агрегатора поддерживает процесс закупок вплоть до заключения договоров поставки с различными продавцами и обеспечивает покупателя информацией о состоянии поставки.
– Агрегаторы такси — компания (юридическое лицо или индивидуальный предприниматель), осуществляющая приём и (или) передачу заказов на перевозку пассажиров и багажа легковым такси.
– Агрегаторы геоконтента — компании и программное обеспечение, обеспечивающие сбор, агрегацию и использование геоданных с любыми источниками непространственных данных, а также непрерывную высокоточную их актуальность.
– Агрегаторы новостей, социальных сетей, рецензий и др. информации.

Таким образом агрегаторы составляют крупный класс информационных систем, для которых характерны:

– Работа с большими объемами данных.
– Интенсивный поток задач.
– Подключение агрегатора к большому числу удалённых распределённых ресурсов.
– Удалённый характер взаимодействия: базы удалённых ресурсов слишком большие, чтобы их можно было целиком сохранить в памяти агрегатора, и содержат информацию, представляющую коммерческую или иную секретность. Опрос удалённых источников производится через предоставленные ими самими протоколы взаимодействия.
– Преимущественно «информационный характер» решаемых задач: как правило, целью работы, запущенной на агрегаторе, является выяснение информации, производить дополнительные вычисления с этой информацией не требуется, либо они занимают незначительное пренебрежимо малое время.
– Наличие условного центрального звена (которое физически может быть представлено в виде распределённого суперкомпьютера), принимающего решения и опрашивающего удалённые информационные ресурсы.

– Ограничение на одновременный доступ к удаленным ресурсам - возможно не более фиксированного количество подключений.
– Различные скорости получения информации от удалённых источников, зависящие как от непосредственной технической организации удалённого ресурса, так и от влияния внешних факторов, таких как географическая удалённость источника, ширина и скорость интернет соединения и т. д.
– Зависимость времени удалённого запроса от объема запрашиваемой информации.

С точки зрения программно-аппаратной организации системы агрегации являются распределенными системами. Техническая организация и оснащение таких систем рассматривается в теории распределенных систем [81; 82]. В общем случае выделяют следующие уровни работы распределенных систем:

– Уровень приложений;
– Промежуточный уровень, уровень обработки;
– Уровень ОС.

Взаимодействие между процессами, т. е. независимыми экземплярами программ во время выполнения, которым выделены системные ресурсы, в распределённой системе осуществляется по многоуровневому протоколу согласно модели OSI [83] на нескольких уровнях. Каждый уровень выполнения инкапсулирует предыдущий, поэтому для целей данного исследования достаточно рассматривать работу на уровне прикладного взаимодействия. Алгоритмы составления расписаний, как правило, принадлежат к программному обеспечению промежуточного уровня (middleware).

Для рассматриваемого нами класса систем характерно наличие центрального звена, подключенных к нему удалённых ресурсов и вычислительных мощностей (машин), служащих для обработки заявок. В зависимости от исследовательского подхода, возможно рассмотрение машин и удалённых ресурсов системы либо как субъектов с жёстким подчинением центру, либо в парадигме агентного подхода, где учитывается разнонаправленность интересов субъектов и их возможности к самоорганизации (в производственно-экономическом отношении такие модели были рассмотрены в работах П.О. Скобелева [84; 85], Н.А. Кузнецова [86; 87], В.А. Виттиха [85], В.И. Городецкого [88; 89], И.Д. Зайцева [90], А.Б. Шабунина [91]).

1.2.2 Функциональная модель составления адаптивных расписаний в системе агрегации

Введём в рассмотрение следующую концептуальную функциональную модель составления адаптивных расписаний в системе агрегации (рисунок 1.3), разработанную с помощью категориального метода «Схема функциональной системы».

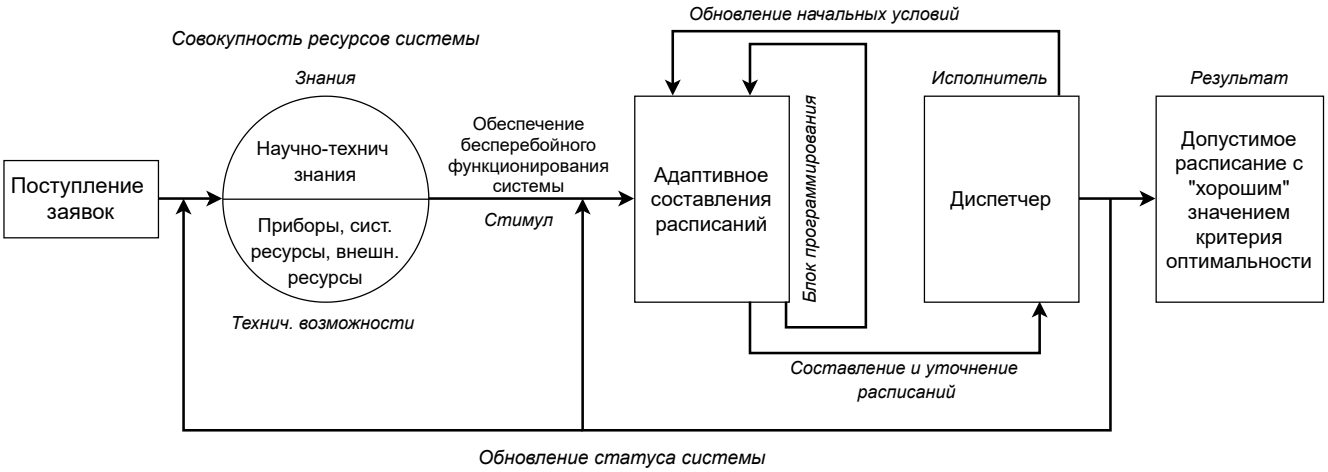


Рисунок 1.3 — Функциональная модель составления адаптивных расписаний в системе агрегации.

На рисунке 1.3 показано функциональное поведение системы-агрегатора, соответствующее следующему кратком описанию. На вход системы поступают пользовательские заявки, для выполнения которых в системе создаются работы. Совокупность ресурсов системы можно разделить на знания - научно-методические ресурсы, методы и технологии и техническое возможности, определяемые приборами, которые имеются в наличие в системе, её внутренними ресурсами таким, как память, связи между компонентами системы и внешними ресурсами, подключенными к агрегатору через удалённое соединение. Обеспечение бесперебойного функционирования системы требует учет поступившей работы, её декомпозицию на задачи и назначение этим задачам стартовых времен обработки. Эти действия производятся в блоке Адаптивного составления расписаний под контролем программы-Диспетчера. Результатом работы системы является допустимое расписание обработки задач, включающее все поступившие в систему требования.

Далее опишем подробно составные элементы системы и характер связей между ними.

В методе «Схема функциональной системы» на входе находится блок «Внешнее воздействие». В этом качестве выступают различные изменения среды, требующие ответа от системы, побуждающие системный объект адаптироваться к изменению условий.

В модели «Функциональная модель составления адаптивных расписаний в системе агрегации» в качестве внешнего воздействия выступает поступление заявок.

Элемент модели «Поступление заявок» отражает тот факт, что система агрегации является системой массового обслуживания, заявки на информационное обслуживание (получение данных) в неё поступают пуассоновским потоком с интенсивностью λ.

Следующий элемент метода — «Память/Ресурс». «Память» представляет собой наличие опыта поведения объекта в сходных условиях. А «Ресурс» — потенциал объекта или элементов среды, который может быть использован для формирования функциональной системы.

В разработанной нами модели элемент «Память» представлен блоком «Научно-технические знания», а элемент «Ресурс» — блоком «Технические возможности».

В системе агрегации циркулируют знания научно-технического характера (конкретные технические решения по организации системы, концепции её организации). Ресурс, представленный техническими возможностями, предполагает наличие приборов (также называемых в рамках данной работы «приборами» — ЭВМ, выполняющие обработку данных в соответствие с запросами пользователей), системных ресурсов (память — разные виды, ширина канала доступа, время отклика) и различных внешних ресурсов (например, удалённые хранилища данных).

Третий элемент метода — «Блок программирования». Это та область системы, где разрабатываются сценарии возможных ответов системы на внешнее воздействие, где с каждым из сценариев проводятся эксперименты (тестирование) для выбора наиболее адекватного из них.

В разработанной нами модели «Блок программирования» воплощен в блоке «Адаптивное составление расписаний». В нём содержатся алгоритмы, основанные на математических моделях обработки данных и происходит составление и адаптация расписаний обработки задач в зависимости от вновь приходящих заявок, параметров заявок и обратной связи, получаемой от элемента «Диспетчер».

Следующий элемент метода — «Орган-исполнитель». Это подсистема, реализующая сценарий, подготовленный в рамках «Блока программирования», область проявления реакций системы на воздействие со стороны внешней среды.

В разработанной нами модели «Орган-исполнитель» представлен блоком «Диспетчер». Он осуществляет управление работой блока составления расписаний, основываясь на вновь поступающей в систему информации, с целью оптимизировать значение целевого функционала системы.

Последний элемент метода — «Результат». Он отражает характер приспособительной реакции системы относительно внешнего воздействия, за счёт обратной связи позволяющей системе совершенствоваться. В разработанной нами модели в качестве «Результата» выступает блок «Допустимое расписание» с „хорошим“ значением критерия оптимальности». Оно показывает ожидаемые выходные данные системы. Для модели «Функциональная модель составления адаптивных расписаний в системе агрегации» такими данными являются допустимые расписания обработки задач, удовлетворяющие ресурсным ограничениям системы, с оптимальным временем получения и достаточно хорошими значениями критериев оптимальности. К сожалению, поскольку большинство задач составления расписаний в сложных системах являются NP-трудными (в частности, NP-трудность в строгом смысле задач составления расписаний в СА доказана нами в части 2.1. данной работы), получение оптимальных решений оказывается не всегда достижимым за разумное (уточнить, какое) время. Подробный комментарий об этом будет дан далее в части 1.3, содержащей анализ методов составления расписаний в сложных системах.

Выходные данные системы составления расписаний далее используются другими составными частями СА как СМО для эффективного обслуживания поступивших в систему заявок. Фактически, в рамках данной системы принимаются решения о преобразовании входящих заявок пользователей в технические задачи и планирование их исполнения - такое, как эффективное назначение им ресурсов и времён начала обработки.

Связи показывают поведение системы, т. е. целенаправленное изменение её состояния во времени, реализуемое системной согласно её целям, посредством направленного влияния элементов модели друг на друга. Инструментом влияния при этом выступает информация в виде потока данных.

В концептуальной модели на рисунке 1.3 прямые связи отражают непосредственное поступление в систему новых данных, а обратные - саморегуляцию эле-

ментов системы в ответ на вновь поступающие воздействия внешней среды. Так, адаптивное изменение плана обслуживания заявок в системе нужно для обеспечения бесперебойного функционирования системы (связь «Стимул»). Поступление новых данных учитывается блоком «Диспетчер» и запускает процесс обновления начальных условий для блока «Адаптивное составление расписаний» (связь «Обновление начальных условий»). Данная информация используется этим блоком для обновления расписания таким образом, чтобы обеспечить стабильное функционирование системы, что показано на схеме 1 связью «Составление и уточнение расписаний». После получения обновления расписания блок «Диспетчер» проверяет его на допустимость и исследует значения критерия оптимальности. В случае неудовлетворительного результата инициируется перерасчёт расписания.

По мере выполнения задач происходит обратное распространение информации о текущем состоянии системы, необходимое элементам системы для оптимизации своего функционирования. Этот процесс обозначен на рисунке 1.3 связью «Обновление статуса системы».

1.2.3 Логическая модель «сущность-связь» для составления адаптивных расписаний в системе агрегации

Используя системный подход, выделим следующие компоненты (сущности) системы-агрегатора: Работы, Задачи, Приборы, Ресурсы и Диспетчер. Взаимодействие между сущностями в рамках логической модели «сущность-связь» (ERD) показано на рисунке 1.4. Описание использованных обозначений приведено в таблице 21 приложения. Соотношения кардинальности и ординарности связей между сущностями приведены в нотации «вороньи лапки», представленной в таблице 22 приложения.

Приведем описание сущностей и их атрибутов, согласно модели (3):

– **Работа.** Каждая работа в системе имеет свой номер j (обозначенный на схеме как Job ID), время поступления в систему r_j, описание потребностей в ресурсах V, может быть декомпозирована на независимые задачи T_{ji}, список которых также является её атрибутом.
– **Задача.** Является зависимой «дочерней» сущностью для Работы. Наследует от родительского элемента время готовности, в случае задачи обозначаемое r_{ji}. Также характеризуется номером (Task ID), статусом выполнения: «поступила», «назначена», «выполняется», «выполнена», потребностью в определенном ресурсе V_{ji} и ожидаемым временем выполнения p_{ji}.

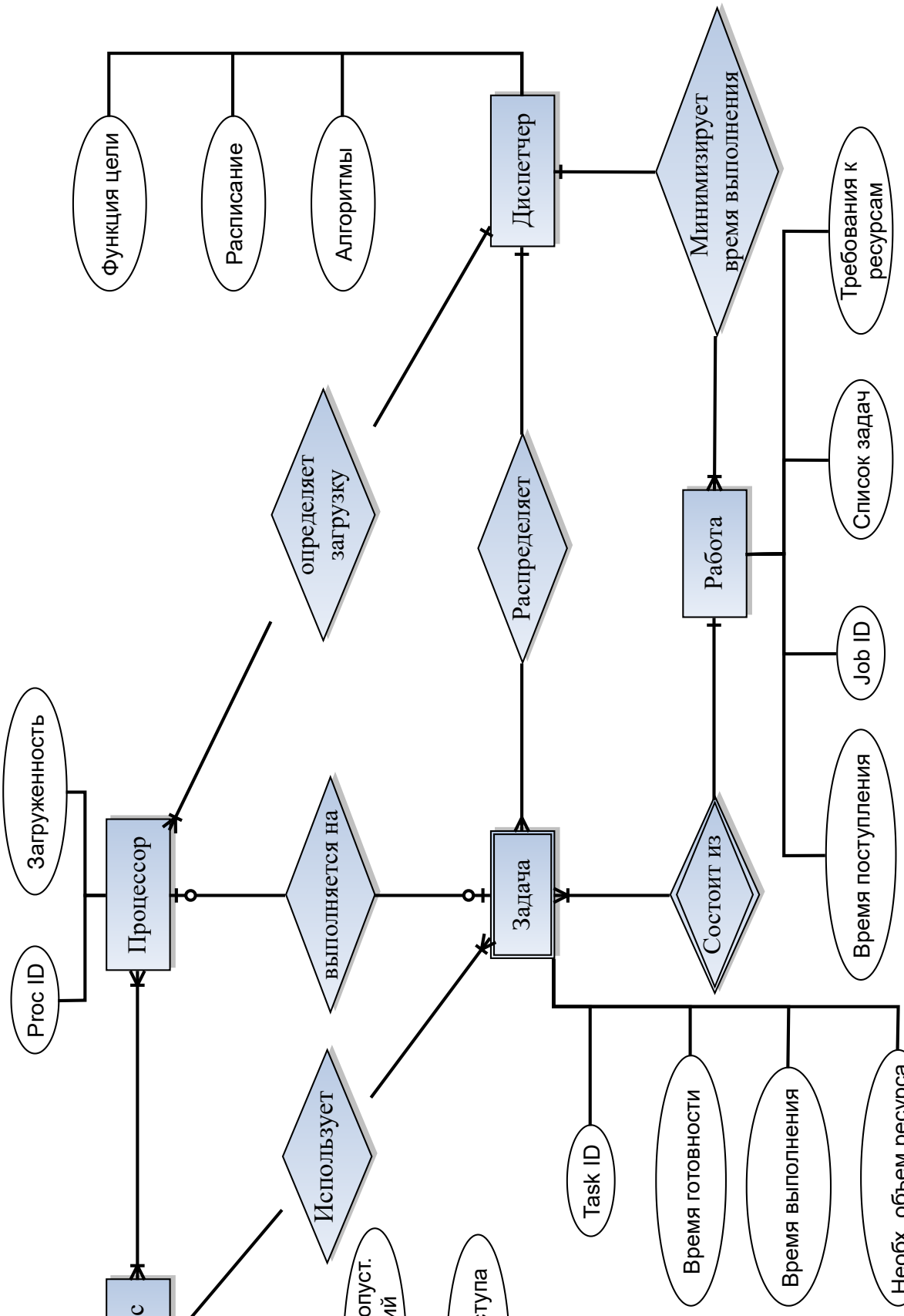


Рисунок 1.4 — Модель сущность-связь для системы агрегации.

– **Прибор**, он же обслуживающий прибор. Представлен в системе в единственном или множественном числе. Характеризуется своим Proc ID и степенью загруженности.
– **Ресурс.** Атрибутами ресурса являются его номер Res ID, скоростью доступа к ресурсу μ_s и количеством доступных соединений L. При ограничения на доступ к ресурсу мы считаем связанными не с его количеством, а с количеством возможных соединений. В этом отношении мы можем говорить о том, что количество ресурса не ограничено, но ограничено количество «экземпляров» ресурса, что соответствует нотации [74] res$S.L$, где S — количество различных ресурсов, а L — количество доступных соединений.
– **Диспетчером** будем называть сущность, ответственную за назначение задачам приборов и доступных ресурсов. Атрибутами диспетчера будем называть используемые им алгоритмы составления расписаний, текущее расписание и целевую функцию, подсчет которой используется для определения допустимости расписания.

Рассмотрим связи, имеющие место между сущностями модели (таблица 1).

Таким образом, определив главные компоненты исследуемой системы и взаимоотношения между ними, можем перейти к её теоретико-множественному описанию.

1.2.4 Теоретико-множественное описание системы агрегации

CA = {Работы, Задачи, Приборы, Ресурсы, Диспетчер}
Работы = $\{\{J_1, \ldots, J_n\}, j, n, p_j, V_j = (V_{ji_1}, \ldots, V_{ji_k}), k, r_j, s_j, C_j\}$
Задачи = $\{\{T_{ji_1}, \ldots, T_{ji_k}\}, j, i, r_{ji}, s_{ji}, C_{ji}, V_{ji}, p_{ji}, PR_{js}(t)\}$
Ресурсы = $\{\{R_1, \ldots, R_S\}, s, S, \mu_s, B_s(t)\}$
Приборы = $\{\{P_1, \ldots, P_m\}, l, m\}$
Диспетчер = $\{F_{aim} = \{C_{max}, \sum_{j=1}^{n} C_j, C_a\}, Sched\}$
Sched = $(s_{ji}), j = 1, \ldots, n, i = i_1, \ldots, i_k$.

Определение* 4.** ***Допустимым *расписанием для CA будем называть расписание, в котором не нарушаются условия на количество одновременных подключений к ресурсу и на количество задач, одновременно обрабатываемым одним прибором.*

Определение* 5.** *Расписание для CA, для которого выполняются условия допустимости и которое минимизирует целевой функционал F_{aim}, будем называть* ***оптимальным*.*

Таблица 1 — Связи между сущностями логической EDR модели системы-агрегатора

Пара сущностей	**Тип связи**	**Комментарий**
Работа-Задача	Одна к одной или многим	Каждая работа может быть декомпозирована на одну или несколько задач в зависимости от потребностей в ресурсе
Работа-Диспетчер	Многие к одному	Диспетчер при составлении расписания учитывает атрибуты работы для минимизации целевой функции - суммы времен выполнения работ или общей длины расписания
Диспетчер-Задача	Один ко многим	Диспетчер назначает стартовое время выполнения для каждой задачи, прибор, на котором она будут обрабатываться, и гарантирует доступность ресурса на время выполнения
Задача-Ресурс	Одна к одному или нулю	Декомпозицию по задачам будем производить так, чтобы одна задача использовала только один ресурс
Задача-Прибор	Один к одному	Одна задача может быть назначена только на один прибор. Один прибор может одновременно обслуживать только одну задачу
Прибор-Диспетчер	Один ко многим или одному	Диспетчер направляет задачи на выполнение на приборы, которых в системе может быть один или более
Прибор-Ресурс	Один ко многим или одному	В зависимости состояния системы, один прибор может иметь доступ к одному или многим ресурсам

1.2.5 Характеристики задач составления расписаний в СА

Таким образом, особенностями СА, которые следует учитывать в рамках моделирования обработки заявок в СА в контексте теории расписаний, являются:

– Отсутствие прерываний;
– Ограниченная доступность ресурса в любой момент времени;
– Отсутствие ограничений на количество ресурса;
– Разнородность ресурсов и скорости доступа к ним;
– Постоянное поступление в систему новых требований и необходимость пересчёта (адаптации) расписания обработки заявок;
– В рамках данной работы будем рассматривать системы, где возможна такая декомпозиция работ на задачи, что между задачами будут отсутствовать ограничения предшествования;
– Ресурсы являются возобновляемыми и непрерывными;
– Продолжительность выполнения работы зависит от объема и типа запрашиваемых ресурсов;
– Все работы имеют продолжительность больше нуля, поддающуюся оценке (методы оценки продолжительности будут рассмотрены нами в главе 3). Мы не рассматриваем контрольные события (работы с продолжительностью 0) или промежуточные работы, не имеющие фиксированной продолжительности.

1.3 Модели и методы составления расписаний в условиях ограниченных ресурсов

Первые работы, посвященные исследованию составления расписаний в условиях ограничения на дополнительные ресурсы, появились в конце 50-х годов 20-го века. Рост интереса к этому направлению продолжился в 60-х [92—97] и 70-х [98—101]. Первые попытки классификации ресурсных ограничений и категоризации понятия ресурса можно обнаружить в обзорах Э.В. Дэвиса [94; 102], В.С. Херролена [103] и др. Значительного успеха в этом отношении удалось достигнуть Грехему и соавторам в работе [73]. Позже, существенный успех в исследовании задач в ресурсными ограничениями был достигнут в работах Блажевича и соавторов [74; 104; 105], где также впервые была введена нотация записи ограничений на ресурсы, принятая на сегодняшний день. В то время, как одними авторами, такими как Я. Блажевич, А. Яняк, Я. Грабоский, Дж.К. Ленстра и А.Х.Г. Ринной-Кан, исследовались задачи построения расписаний для приборов с наличием ресурсных ограничений, параллельно такими авторами как М.Р. Гэри,

Д.С. Джонсон, П. Брукер, П. Колиш, Я. Веонгларц и др. развивалось исследование ограничений на ресурсы в задачах планирования расписаний для проектов и разрабатывались модификации таких задач и ограничений. При этом нотация, введённая в [74] стала де-факто стандартом записи задач составления расписаний с ограниченными дискретными ресурсами.

Рассмотрим данную нотацию более подробно. Предположим, что для обслуживания заданий системе нужны ограниченные дополнительные ресурсы $R_1, \ldots, R_S$, где R_s используется для обозначения ресурса i-того типа.

***Определение* 6.** *Под типом ресурса будем понимать обладание им определёнными функциональными признаками, однозначно отличающими его от ресурсов других типов.*

Например, у лекарственного средства, как у ресурса, необходимого для обслуживания больных тип может определяться их медикаментозными свойствами или форм-фактором.

Пусть для каждого ресурса R_s задано $\kappa_s \in \mathbf{N}$, определяющее общее количество доступного ресурса R_s в системе в любой момент времени.

Количество ресурса R_s, необходимого для обслуживания работы J_j обозначается за ν_{sj} и выделяется для задачи в течение всего периода её обслуживания. В случае многоэтапного обслуживания работы или других ситуаций, когда работы J_j оказывается декомпозируема на задачи или операции, объем ресурса R_s, выделяемого на каждую задачу T_{ji} обозначается как ν_{sji}.

***Определение* 7.** *Расписание считается допустимым с точки зрения выполнения ресурсных ограничений, если в любой момент времени t выполняется:*

$$\sum_{j \in S_t} \nu_{sji} \leq \kappa_s, \quad s \in [1, \ldots, S],$$

где $S_t = j \in \{1, \ldots, n\}$: J_j выполняется в момент времени t.

Тогда для введённых выше условий справедлива следующая классифицирующая схема (нотация):

$$\mathrm{res}\lambda\sigma\varrho, \tag{1.1}$$

где

– λ показывает количество разных типов ресурсов в задаче и записывается как S, если число типов фиксировано и равно $S \in \mathbf{N}$, и '.', если количество различных типов дополнительных ресурсов в задаче не фиксировано.

– σ показывает количество единиц каждого s-того ресурса в случае, если он ограничен условиями задачи и является одинаковым для каждого типа ресурса. В этом случае пишут $\sigma = \kappa_s =_{\forall s} \kappa$, иначе — $\sigma =$ ’.’.

  Одними из наиболее изученных в литературе по дискретным задачам ТР являются задачи, где $\kappa_s =_{\forall s} \kappa = 1$, для записи которых данная нотация является удовлетворительной.

  Отметим, однако, что для многих практических приложений ситуация, когда все ресурсы в системе доступны в одинаковом количестве, выполняется крайне редко, и для моделирования таких ситуаций данная нотация оказывается недостаточной.

– Величина ϱ показывает максимальный объем ресурса, необходимый для обслуживания задания в системе. Его определяют как $\varrho = max_{s,j} \nu_{sj}$. Иначе пишут $\varrho =$ ’.’.

  Заметим, то такая оценка необходимого для выполнения работы количества ресурса также является достаточно грубой для моделирования прикладных систем, несмотря на то, что является достаточной для теоретических изысканий и выведения общих закономерностей составления расписаний в системах с ограниченными ресурсами.

Большинство известных результатов в области дискретных моделей составления расписаний обслуживания заявок приборами, было получено для λ, σ и ϱ равных либо ’.’, либо 1. Обзор данных результатов приведён в таблице 2.

Таблица 2 — Известные результаты для задач составления расписаний для приборов с ограниченными ресурсами

Задача	Известный результат	Автор(ы)
$P_3 \mid \text{res1}, \ldots, p_{ij} = 1 \mid C_{\max}$	NP-трудная в строгом смысле	М.Р. Гэри, Д.С. Джонсон [99]
$Q_2 \mid \text{res1}, \ldots, p_J = 1 \mid C_{\max}$	Разрешима с временной сложностью $O(n \log n)$	Я. Блажевич, Я.К. Ленстра, А.Х.Г. Ринной Кан [74]

Продолжение на следующей странице

Задача	Известный результат	Автор(ы)
$Q \mid \text{res1.1}, p_J = 1 \mid C_{\max}$	Разрешима с временной сложностью $0\left(n^3\right)$	Я. Блажевич, Я.К. Ленстра, А.Х.Г. Ринной Кан [74]
$P_2 \mid \text{res1} \ldots, \text{tree}, p_{ij} = 1 \mid C_{\max}$	NP-трудная в строгом смысле	М.Р. Гэри, Д.С. Джонсон [99]
$P_2 \mid \text{res111}, \text{prec}, p_{ij} = 1 \mid C_{\max}$	NP-трудная в строгом смысле	Дж.Д. Ульман [106]
$P_2 \mid \text{res1} \ldots, \text{chain}, p_{ij} = 1 \mid C_{\max}$	NP-трудная в строгом смысле	Я. Блажевич, Я.К. Ленстра, А.Х.Г Ринной Кан [74]
$Q \mid \text{res1.1}, p_J = 1 \mid \max_j \left\{f_j\left(C_j\right),,\right.$	Разрешима за полиномиальное время	Я. Блажевич, Я.К. Ленстра, А.Х.Г Ринной Кан [74]
$\left.\sum f_j\left(C_j\right)\right\}$, for arbitrarty non-decreasing cost functions $f_j : Z^n \rightarrow Z$	NP-трудная в строгом смысле	Я. Блажевич, Я.К. Ленстра, А.Х.Г Ринной Кан [74]
$P \mid \text{res1.1}, p_j = 1 \mid L_{\max}$	NP-трудная в строгом смысле	Я. Блажевич, Я.К. Ленстра, А.Х.Г Ринной Кан [74]
$Q \mid pmtn$, res 111 $\mid C_{\max}$	Разрешима за полиномиальное время	Е.К. Хорват, С. Лам, Р. Сети [107]

Продолжение на следующей странице

Задача	Известный результат	Автор(ы)
$F_2 \mid \text{res111}, p_{ij} = 1 \mid C_{\max}$	$O(m \log(n+m))$	Я. Блажевич, Я.К. Ленстра, А.Х.Г Ринной Кан [74]
$F_2 \mid \text{res111} \mid C_{\max}$	Разрешима с временной сложностью O(n)	С.М. Джонсон [108]
$P_2 \mid \text{res...}, p_j = 1 \mid C_{\max}$	Разрешима за полиномиальное время	М.Р. Гэри, Д.С. Джонсон [99; 109]
$P \mid \text{res...}, p_j = 1 \mid C_{\max}$	Существует несколько приближенных алгоритмов для решения задачи, лучшая оценка времени выполнения $R = \left(\frac{m}{2}\right)$	К. Краузе, Э. Г. Кофман, Х. Рёк [110—112]

Заметим, что данные категории ограничений, хотя и являются достаточными для формулирования и решения широкого круга проблем, могут быть недостаточными для моделирования особенностей, возникающих при понимании информации, как ресурса, используемого системой для эффективного функционирования, и в рассмотренных выше моделях с дискретными ограничениями не учитываются случаи, когда ограниченная доступность ресурсов связана не непосредственно с его имеющимся количеством, а с совокупностью внешних факторов, ограничивающих как одновременный доступ к ресурсу несколькими приборами, так и объем ресурса, получаемый системной в единицу времени (далее определяемый как «скорость получения ресурса»). Ресурсы, ограниченные в данном смысле, определим в категорию «качественно ограниченных» (по аналогии с существующей категорией «количественной доступности»). Расширенная таким образом классификация ресурсных ограничений представлена на рисунке 1.5.

Предложена следующая форма записи таких ограничений. Для обозначения количества одновременных соединений определить параметр ϕ:

– либо как $\phi = [L_1, \ldots, L_S]$, в случае, если важно учесть различные ограничения на доступ L_s, $s \in [1, \ldots, S]$ у разных ресурсов,
– либо как $\phi = \max_{s=1}^{S}\{L_s\}$,
– либо как '.', если такого ограничения не задано.

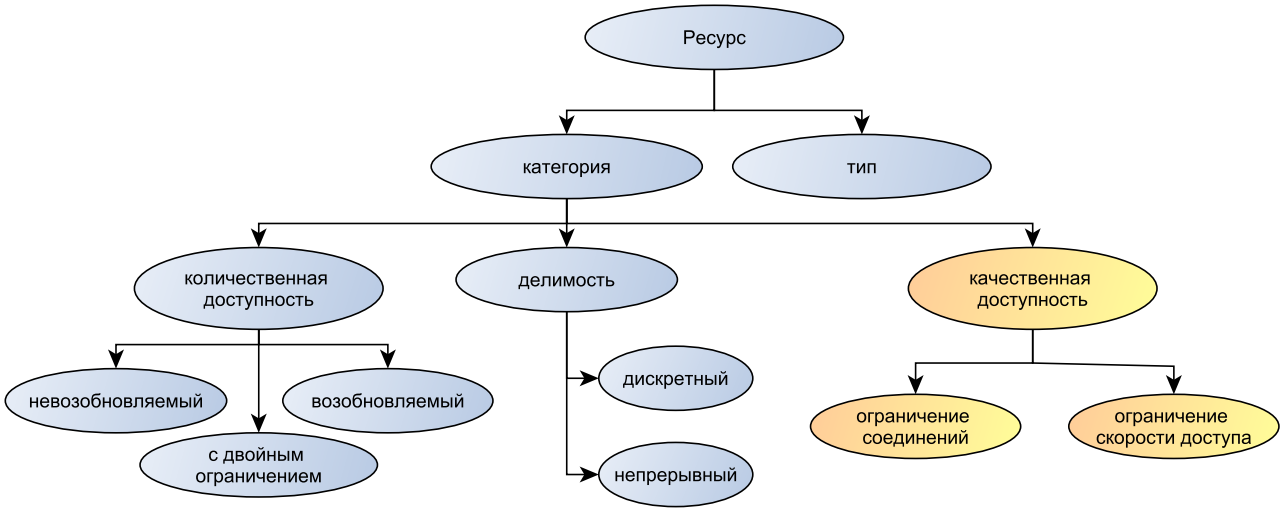


Рисунок 1.5 — Классификация ограничений дополнительных ресурсов в задачах теории расписаний, расширенная за счёт введения категории качественной доступности ресурса.

По аналогии в рассмотрение введён параметр $\hat{\mu}$ для обозначения скорости доступа, задаваемый как:

– либо как $\hat{\mu} = [\mu_1, \ldots, \mu_S]$, в случае, если важно учесть различные ограничения на доступ μ_s, $s \in [1, \ldots, S]$ у разных ресурсов,
– либо как $\hat{\mu} = \max_{s=1}^{S}\{\mu_s\}$,
– либо как '.', если такого ограничения не задано.

Таким образом, ограничения на ресурсы в задачах ТР можно записать как $\mathrm{res}\lambda\sigma\varrho\phi\hat{\mu}$, что является расширением принятой на сегодняшний день нотации $\mathrm{res}\lambda\sigma\varrho$.

1.3.1 Анализ моделей и методов составления расписаний для систем с непрерывными возобновимыми ресурсами

Анализируя литературу, можно выделить следующие направления моделирования задач с непрерывными ресурсами:

– Модели с зависимостью скорости обработки задач от требуемого для выполнения задачи объема ресурса;
– Модели с зависимостью времени обработки задачи от требуемого для выполнения задачи объема ресурса;
– Модели с зависимостью требуемого для выполнения задачи объема ресурса и времен поступления задач $r_j i$.

Выполним краткий обзор работ по данным направлениям.

Модель «Скорость обработки — Объём ресурса»

Базовый случай

Пусть задан непрепывный ресурс в количестве (объёме) U. Задачи могут резервировать долю u_j от объёма, которая может изменяться от времени, т.е. $u_j = u_j(t)$.

Обозначим объём выполненной задачи x_j, скорость её выполнения $v_j = f_j(u_j)$, общий объём задачи X_j, время завершения задачи C_j.

Тогда

$$v_j(t) = \frac{dx_j(t)}{dt} = f_j(u_j(t)), \tag{1.2}$$

а

$$X_j = \int_0^{C_j} f_j(u_j(t))dt.$$

Также обозначим U совокупность резервирований всех ресурсов за всё время выполнения, V — совокупность скоростей выполнения задач; V однозначно определяется через U.

В [68] показано, что если $u_j(t) = const$ и функции $f_j(u_j)$ вогнутые, то объём $V(U)$ выпуклый, и решение, наилучшее по критерию оптимальности C^*_{max}, определяется пересечением границы объёма V с прямой, определяемой параметрическими уравнениями $v_j = \frac{X_j}{C_{max}}$, т.е. решением уравнения (1.3)

$$U = \sum_j u^*_j = \sum_j f_j^{-1}(X_j/C^*_{max}), \tag{1.3}$$

где u^*_j — оптимальное использование ресурса задачами. В некоторых случаях данное уравнение может быть решено аналитически.

Далее рассматривается случай, когда $f_j \le c_j u_j, c_j = \frac{f_j(U)}{U}$, и показывается, что существуют оптимальные решения, когда задачи выполняются последовательно, и когда параллельно с постоянным использованием ресурса $u_j = const$.

Случай для задач с ограничениями предшествования

Рассмотрим случай, когда некоторые задачи могут начать выполняться только после завершения других: например, T_3 и T_4 — только после завершения T_1 (см. раздел 1.1.3 данной главы). Начертим граф зависимостей, где рёбра — задачи, а узлы — зависимости (см. рисунок 1.6). Разобьём задачи T_j на группы нескольких задач $T_k = \{T_{j_1}, T_{j_2}, \}$, которые могут выполняться параллельно. Например, на рисунке 1.6 это будут $T_1 = \{T_1, T_2\}, T_2 = \{T_2, T_3, T_4\}, T_3 = \{T_4, T_5\}$. Если

задача входит одновременно в несколько групп, разобьём её на части T_{jk}: например, на рисунке 1.6 часть задачи T_2 может выполняться параллельно с задачей T_1 (T_{21}), а часть — с задачами $T_3, T_4 (T_{22})$. В зависимости от выбранного расписания, объёмы этих задач будут X_{jk} ($\sum_k X_{jk} = X_j$).

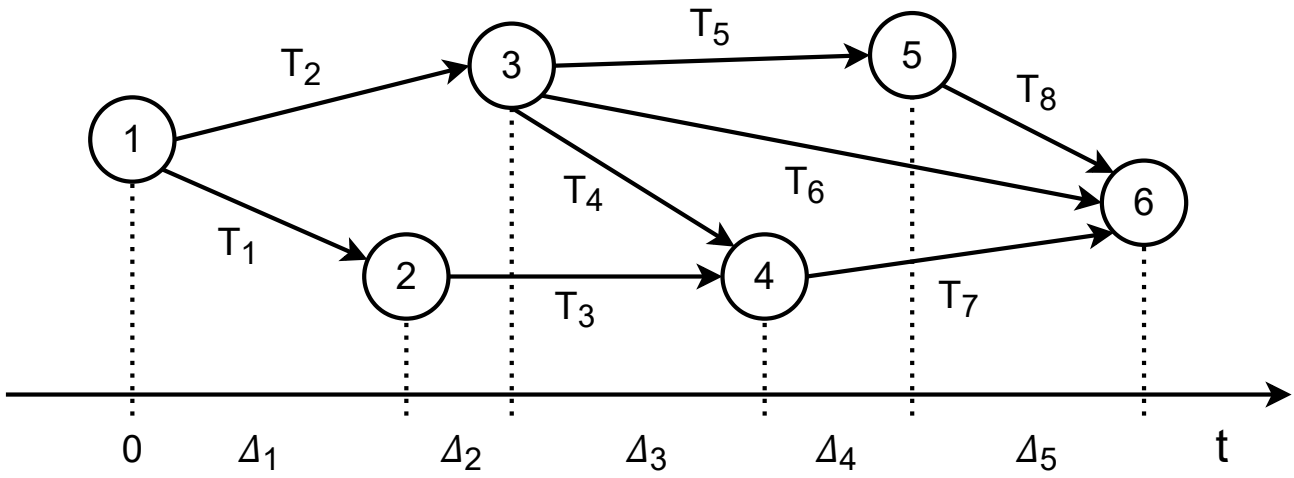


Рисунок 1.6 — Пример графа зависимостей для обслуживания задач с заданными ограничениями предшествования.

Пусть время выполнения группы T_k равно Δ_k. Тогда нахождение оптимального распределения ресурсов в группе сводится к уравнению (1.3) в форме и можно применять аппарат, развитый для предыдущего базового случая, в т.ч. его особых случаев. Основной проблемой является получение оптимального разбиения X_j на X_{jk}, минимизирующего $C_{max} = \sum_k \Delta_k^*$.

В [68] показано, что C_{max} является выпуклой функцией от X_{jk} для произвольных f_j, поэтому мы имеем задачу выпуклого программирования с линейными ограничениями.

Случай с независимыми задачами и количеством m машин меньшим количества задач n

Задачу можно свести к предыдущей, обозначив T_k группу задач, выполняющихся параллельно. Таких разбиений на группы существует C_n^m (число сочетаний из n по m). В сколько-то нетривиальных случаях количество таких сочетаний слишком много, чтобы решить задачу аналитически, и необходимо использовать эвристики.

Общая идея таких эвристик следующая. Пусть структура разбиения на группы — вектор $(K_1, , K_j, , K_n)$, где K_j — сколько раз T_j встречается в группах T_k. Идея — изучать соотношение между K_j и объёмами задач X_j. Так, если f_j выпуклые и

одинаковые — выбирать такие разбиения, где K_j примерно пропорционально X_j. Обоснование подхода и примеры конкретных алгоритмов приведены в [113].

Другой подход состоит из двух шагов:

1. Взять некоторые фиксированные объёмы ресурса u_j и распланировать задачи по приборам исходя из времён выполнения $p_j = \frac{X_j}{f_j(u_j)}$;
2. Распределить ресурс между частями задач в расписании, полученном в п. 1

В большинстве случаев шаг 2 не представляет затруднений — как показано в предыдущем пункте, для некоторых случаев он даже решается аналитически.

Сложность шага 1 радикально отличается для задач с разрещёнными прерываниям и без.

Для задач с прерываниями сложность — $O(n)$, если использовать алгоритм Макнотона, для обработки без прерываний задача NP-сложная [43]. Также возникает вопрос, как определять объёмы ресурсов u_j для шага 1. Некоторые подходы описаны в [104], они дают результаты, отличающиеся на несколько процентов от оптимального случая. Также выбор u_j можно улучшать, выполняя шаги 1 и 2 итеративно.

Следует отметить, что шаг 1 представляет собой задачу дискретного планирования, а шаг 2 — задачу распределения непрерывного ресурса. Таким образом, в них можно использовать различные подходы из соответствующих разделов теории расписаний.

Для задач, рассмотренных в данной части, возможны следующие обобщения:

– Кроме ограничения на объём ресурса, доступный в единицу времени, можно задать ограничение на суммарный объём задач $\sum_j X_j \neq V_{lim}$. Данный случай рассмотрен в [114];
– Задачи могут требовать несколько видов разделяемых ресурсов. Рассмотрено там же;
– В литературе рассматривались также другие критерии оптимальности, такие как: $\int_0^{C_{max}} g(u(t))dt$ [115]; $\sum w_j C_j$ [116], [117]; L_{max} [118];
– Ещё один подход состоит в изучении последовательностей на множестве зависимых задач [119].

Рассмотренная модель также находит свое применение в решении задач составления расписаний обработки заданий в Grid-инфраструктуре[67]. Обширный обзор результатов, касающихся планирования при ограниченных ресурсах, можно найти в обзоре [120].

Модель «Время обработки — Объём ресурса»

В данном разделе рассмотрим задачу составления расписания обслуживания задач без прерыванияна единственном приборе, где их времена выполнения определяются функцией

$$p_j = b_j - a_j u_j, \quad a_j > 0, b_j > 0,$$

$$u_j^{min} \le u_j \le u_j^{max}, \; u_j^{min} \in [0; b_j/a_j], \; u_j^{max} \in [0; b_j/a_j].$$

Обозначим множество задач $T = \{T_1, , T_j, , T_n\}$, вектор использования ресурса $u = \left[u_1, , u_j, , u_n\right]$. Как и в предыдущей модели, $\sum_j u_j = U$ — но в отличие от нёё, ресурс не обязательно возобновляемый. Задачи составления расписаний согласно данной модели были подробно рассмотрены в работах [121—124].

Для дальнейшего рассмотрения примем $u_j^{min} = 0$. Если $u_j^{min} \neq 0$, задача может быть сведена к случаю $u_j^{min} = 0$ очевидной заменой переменных.

Порядок выполнения задач определяется перестановкой индексов задач $z = [z(1), , z(n)]$, где Z — множество всех таких перестановок. Таким образом, расписание определяется парой (z, u). Рассмотрим следующие случаи:

- **Случай** $1|prec, p_j = b_j - a_j u_j, \sum_j u_j \le U|C_{max}$**.** Задача минимизации C_{max} для случая равных времён выполнения и произвольных ограничений на порядок следования, была рассмотрена в [121]. Эта задача может быть записана как $1|prec, p_j = b_j - a_j u_j, \sum_j u_j \le U|C_{max}$. Оптимальное решение не зависит от выбора перестановки $z \in Z$ и достигается распределением ресурса в соответствии с алгоритмом, описанным в [121] для данного случая, временная сложность которого — $O(n \log n)$;
- **Случай** $1|prec, r_j, p_j = b_j - a_j u_j, \sum_j u_j \le U|C_{max}$**.** Рассмотрим вариант модели с произвольными временами поступления заданий в систему $1|prec, r_j, p_j = b_j - a_j u_j, \sum_j u_j \le U|C_{max}$. Легко показать, что оптимальное расписание (z^*, u^*) находится, если порядок выполнения заданий выбирается согласно алгоритму из [125], а выделение ресурсов согласно алгоритму, описанному в [121] для этой задачи. Сложность второго алгоритма и, соответственно, сложность задачи составляет $O(n^2)$;
- **Задача минимизации максимальной задержки** L_{max}**.** Пусть максимальная задержка L_{max} определена, как в п. 1.1.3. Тогда задача $1|prec, p_j = b_j - a_j u_j, \sum_j u_j \le U|L_{max}$ эквивалентна задаче $1|prec, r_j, p_j = b_j - a_j u_j, \sum_j u_j \le U|C_{max}$ из предыдущего рассмотренного случая данной модели. Таким образом, она может быть решена в соответствии с упомянутыми так алгоритмами.

Задача $1|r_j, p_j = b_j - a_j u_j, u_j \leq U|L_{max}$ является NP-трудной в строгом смысле, так как NP-трудным в строгом смысле является более общий случай $1|r_j|L_{max}$, как показано в [126]. Для задачи $1|prec, r_j, p_j = b_j - a_j u_j, u_j \leq U|L_{max}$, с дополнительными ограничениями на порядок следования, точный алгоритм решения опубликован в [122].

Модель «Время поступления — Объём ресурса»

В данной модели предполагается, что время обработки задачи является константой, но время готовности непрерывно зависит от количества выделенного непрерывного ресурса, т. е.

$$r_j = f_j(u_j), u_j^{\min} \leq u_j \leq u_j^{\max}, j = 1, 2, \ldots, n,$$

где все нижние и верхние границы выделения ресурса, $u_j^{\min}$ и $u_j^{\max}$, являются известными константами. Рассматриваются задачи без прерывания на единственном приборе. Задачи этого типа появляются, например, в процессе предварительного нагрева слитка на сталелитейных заводах [127].

Задача $1|r_j = f_j(u_j), \sum u_j \leq U|C_{\max}$

Эта задача является NP-трудной в строгом смысле даже в частном случае линейных функций $f_j = b_j - a_j u_j$ и $u_j^{\min} = 0, \forall j$, и NP-трудной в случае $a_j = a, \forall j$ [127]. Однако при одинаковых r_j, т. е. при $f_j = f, u_j^{\min} = u^{\min}$ и $u_j^{\min} = u^{\min}$ для всех j, задача может быть решена за полиномиальное время. В этом случае оптимальное решение (z^*, u^*) получается путем планирования задач в соответствии с невозрастающим временем обработки p_j (таким образом определяя перестановку z^*), также известны формулы для выделения непрерывного ресурса для z^* [122].

Также в работе [122] определяется вычислительная сложность поиска оптимального решения (z^*, u^*). В частности, если f_j линейно, $b_j = b$ для $\forall j$ и либо 1) $u_j^{\max} = u^{\max}$, $p_j = p$, $\forall j$, либо 2) $a_j = a$, $p_j = p$, $\forall j$ — существуют алгоритмы временной сложности $O(n \log n)$. Оптимальное решение (z^*, u^*) получается путем планирования задач в соответствии с невозрастающими значениями a_j в случае 1), невозрастающими u_j в случае 2) и выделением непрерывного ресурса с использованием соответствующих модификаций формул для общего случая [128]. Для произвольных линейных функций f_j Яняк [128] смог привести алгоритм, с использованием которого для заданного z оптимальное распределение ресурса u_z^* может быть вычислено за время $O(n^2)$.

В той же статье показано, что в случае $a_j = a$, $u_j = u$, $p_j = p$ для $\forall j$ оптимальное решение (z^*, u^*) получается, когда задачи ставятся в очередь в порядке неубывания b_j, а ресурс распределяется в соответствии с алгоритмом из статьи. То же самое верно и для задач, в которых приведенная выше перестановка соответствует невозрастающим порядкам a_j, u_j и p_j. Конечно, алгоритм из [128] также можно использовать для поиска распределения ресурсов для перестановок z, определённых эвристически. В [128] 25 таких эвристик сравнивались экспериментально. Наилучшие результаты для «низкого» уровня ресурсов ($U = 0.2 \sum_j u_j^{\max}$) были получены при упорядочении задач по неубыванию b_j, тогда как для «высокого» уровня ресурсов ($U = 0.9 \sum_j u_j^{\max}$) наиболее эффективными оказались сортировки задач по неубыванию значений $b_j - a_j u_j^{\max}$.

Задача $1|r_j = f_j(u_j), C_{\max} \le \hat{C}|\sum u_j$

Аналогично $1|r_j = f_j(u_j), \sum u_j \le U|C_{\max}$ можно доказать, что рассматриваемая задача уже является NP-трудной в строгом смысле для $f_j = b_j - a_j u_j$, $\forall j$, и NP-трудной при $a_j = a$, $\forall j$ (см. [127]). Также аналогично решению первой задачи, если $f_j = f$, $u_j^{\min} = u^{\min}$ для всех j, задача решается оптимально путём планирования задач в соответствии с невозрастающим p_j (таким образом определяя перестановку z^*) и путём распределения ресурса согласно условию, описанному в [122]. Вычислительная сложность тоже опредяется так же, как для предыдущей задачи.

В [127] для случая, если функции f_j не идентичны и линейны, из формулировки задачи в терминах линейного программирования получена формула

$$u^*_{z(j)} = \max\left\{0, \left(b_{z(j)} + \sum_{i=j}^{n} p_{z(i)} - \hat{C}\right)/a_{z(j)}\right\}, \; j = 1, \ldots, n,$$

согласно которой для заданного z получается оптимальное значение u_z^* за время $O(n)$. На том же основании было показано, что случаи:

$$b_j = b, u_j^{\max} = u^{\max}, p_j = p, \; j = 1, 2, \ldots, n, \tag{1.4}$$

$$b_j = b, a_j = a, p_j = p, \; j = 1, 2, \ldots, n, \tag{1.5}$$

$$a_j = a, u_j^{\max} = u^{\max}, p_j = p, \; j = 1, 2, \ldots, n, \tag{1.6}$$

решаемы за время $O(n \log n)$ путем планирования задач в соответствии с невозрастающим a_j в случае (1.4), невозрастающим u_j в случае (1.5) и невозрастающим b_j в случае (1.6), а также распределением ресурсов в соответствии с формулой (1.3.1). Для каждого из этих случаев z^* не зависит от^.

Эвристики, в которых z определяется эвристически, а u_z^* вычисляется согласно (1.3.1), изучались в [127]. Наилучшие результаты были получены при планировании задач по неубыванию b_j. К сожалению, производительность этих эвристик в наихудшем случае неизвестна.

На основе представленных результатов можно построить множество всех Парето-оптимальных решений для некоторых бикритериальных задач типа $1|r_j = f_j(u_j)|C_{\max} \wedge \sum u_j$, используя идеи, описанные в [129]. Для линейных моделей этот набор был построен в [127].

Задачи, рассмотренные в этом разделе, были обобщены в [130] для случая с произвольными ограничениями предшествования, где была доказана их NP-трудность даже для идентичных линейных моделей r_j. Когда дополнительно все времена обработки идентичны, оптимальное решение (z^*, u^*) может быть построено за время $O(n^2)$.

В [124; 131] рассматривались задачи планирования для одиночных и параллельных машин с нелинейной функцией: время выпуска в зависимости от потребления ресурса, общего для всех задач, с разными скоростями потребления ресурсов задачами. Были минимизированы следующие критерии: общее взвешенное время выполнения задач при ограниченном максимальном количестве ресурса [131], общее использование ресурса при ограниченном общем взвешенном времени выполнения и двухкритериальный подход [124]. Найдены границы между NP-трудными и полиномиально разрешимыми случаями.

Дальнейшее обобщение модели времени выпуска было сделано в [124], где рассматривалась задача планирования одной машины с применением модели (1.2) ко времени выпуска. Благодаря некоторым свойствам задачи сложная задача динамического распределения ресурсов была сведена к простой задаче выпуклого программирования. Также были представлены некоторые алгоритмы аппроксимации с анализом наихудшего случая.

Таким образом, рассмотренные три модели соответствуют следующим вариантам использования ресурса:

1. Без выделения ресурса задача не выполняется, увеличение объёма выделенного ресурса приводит к возрастанию скорости выполнения;
2. Без выделения ресурса задача выполняется за определённое время, определяемое её характеристиками, увеличение объёма выделенного ресурса приводит к уменьшению времени выполнения;

3. Время выполнения задачи не зависит от объёма ресурса, увеличение объёма выделенного ресурса приводит уменьшению времени поступления.

Диссертационная работа фокусируется на задачах составления расписаний в системах хранения и обработки больших данных, где под ресурсами обычно понимаются либо базы данных и иные способы их хранения, либо вычислительные мощности, необходимые для их обработки. В этом случае задачи не могут быть выполнены без выделения необходимых ресурсов, при этом скорость выполнения задачи напрямую зависит от доступного ей объёма ресурса: скорости канала передачи данных, количества и быстродействия вычислительных узлов и т. д. Таким образом, для рассматриваемой в диссертации области применения наиболее характерна первая модель, «Скорость обработки — Объём ресурса». Последние же два типа моделей в целом не типичны для описания задач в системах хранения и обработки больших данных. Таким образом, в дальнейшем в диссертационной работе будет исследоваться развитие модели «Скорость обработки — Объём ресурса» применительно к системам аггрегации.

1.4 Выводы и результаты по главе 1

В рамках главы 1:

1. Произведён анализ направлений, классификации, нотации и основных проблем теории расписаний;
2. Выделена как объект исследования СА и описаны её основные свойства. Построены концептуальная функциональная модель СА, логическая модель «Сущность-связь» для СА, выявленные закономерности формализованы в виде теоретико-множественного описателя системы;
3. Произведён анализ литературы по задачам теории расписаний с ограничениями на ресурсы и выявлена недостаточная исследованность моделей с ограниченными доступом и скоростью получения ресурсов, что составляет затруднение при исследовании СА с помощью известных моделей. Таким образом, обоснованным является создание как дискретных, таки дискретно-непрерывных математических моделей, учитывающих указанные ресурсные ограничения;
4. Определена категория качественных ограничений доступа к ресурсам и предложено расширение классификации задач с ресурсными ограничениями, а так же нотация записи качественных ограничений ресурсов в задачах теории расписаний;

5. Выделены исследовательские задачи, решению которых посвящены последующие главы данной диссертации.

Таким образом, глава 1:

– Определяет объект исследования и обозначает актуальынй статус исследуемого вопроса;
– Определяет цели и задачи для дальнейшего исследования;
– Вносит вклад в развитие теории расписаний и исследование операций.

Глава 2. Математические модели обработки данных в распределённых гибридных архитектурах

Данная глава посвящена построению и исследованию математических моделей составления расписаний в многолинейных системах агрегации с ограниченными информационными ресурсами. Выводятся и исследуются дискретная и дискретно-непрерывные математические модели многолинейной системы с ограниченными восполнимыми ресурсами для систем агрегации. Исследуются возможности построения приоритетно-порождающих функционалов для решения задач составления раписаний для предлагаемых моделей. Исследуются области поиска допустимых решений, описаны комбинации параметров, приводящие к перспективным частным случаям, требующим специальных решений. Исследовано влияние вида функции скорости доступа к непрерывному ресурсу для дискретно-непрерывной модели СА.

2.1 Дискретная модель многолинейной системы с ограниченными восполнимыми ресурсами для систем агрегации

В данной главе исследуем влияние факторов качественной доступности ресурсов при составлении расписаний в многолинейных системах с ограниченными ресурсами. Будем полагать количество типов ресурсов равным S, а их характеристики количественной доступности, а именно — максимально доступный объем σ и максимально возможное количество запрошенного ресурса ϱ — неограниченными.

Пусть дано n работ, и $J = \{J_1, \ldots, J_n\}$ — множество работ, $P = \{P_1, \ldots, P_m\}$ — набор m идентичных приборов.

Для выполнения работ требуются восполнимые ресурсы $R = \{R_l, \ldots, R_S\}$, которые присутствуют в системе в единственном числе, S — общее число доступных ресурсов.

Будем считать, что для каждого типа ресурса s известно характерное время обработки единицы информации μ_s.

Каждая работа может требовать некоего подмножества из k_j ресурсов.

При этом для выполнения работы J_j требуется определённое количество ресурсов, заданное вектором $V_j = \{V_{js} \in [0; +\infty)\}$, $j \in [1, \ldots, n]$, $s \in [1, \ldots, S]$. Вектор V_j будем называть вектором ресурсоёмкости работы J_j. Матрицу $V = \{V_{js}\}$,

$j \in 1, \ldots, n$, $s \in 1, \ldots, S$, будем называть матрицей ресурсоёмкости для множества задач J.

Каждый ресурс в системе будем считать доступным в произвольном количестве, но при этом одновременно доступным только для одной работы.

Будем считать возможной декомпозицию работ J_j на задачи T_j вида $T_j = \{T_{ji_l}, \ldots, T_{ji_k}\}_n$ где индексы $i_1, \ldots, i_{k_j} \in K \subseteq (1, \ldots, S)$, где каждая задача может быть выполнена независимо от остальных задач работы с использованием только одного из ресурсов V_{ji_k}.

Пример 1. Пусть заданы четыре работы $\{J_1, J_2, J_3, J_4\}$ и три ресурса $\{R_1, R_2, R_3\}$. Пусть работа J_1 имеет вектор ресурсоёмкости $V_1 = (\eta_{11}, \eta_{12}, 0)$, работе J_2 соответствует вектор $V_2 = (\eta_{21}, 0,0)$, а векторы ресурсоёмкости задач J_3, J_4 имеют вид $V_3 = (\eta_{31}, \eta_{32}, \eta_{33})$ и $V_4 = (0,0, \eta_{43})$, соответственно.

Тогда допустима декомпозиция работ J_j на независимые (в рамках одной работы) задачи, показанная на рисунке 2.1:

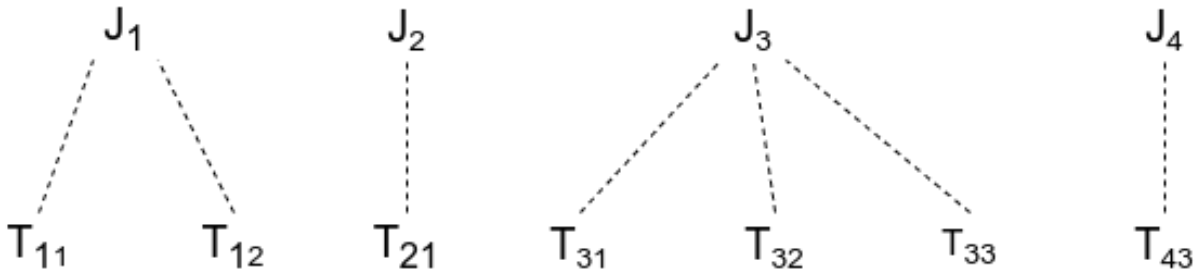


Рисунок 2.1 — Декомпозиция работ на задачи.

Будем считать, что время выполнения задачи T_{ji_k} зависит от коэффициента скорости доступа к ресурсу μ_{i_k} и ресурсоёмкости задачи V_{ji_k}:

$$p_{ji} = g_{ji}\left(\mu_{i_k}, V_{ji_k}\right) = \mu_{i_k} V_{ji_k}$$

где g_{ji} — линейная непрерывная функция, определённая на $[0; +\infty) \rightarrow [0; +\infty)$ (здесь можно понимать как характеристику ресурса, а V_{ji_k} как характеристику задачи T_{ji_k} и работы J_j). Прерывания при обработке задач запрещены.

Выполнение работы J_j будем считать завершённым, когда завершены все её задачи:

$$C_j = \max_{k=1,S}\left(C_{jk}\right) \tag{2.1}$$

Работы J_j, определённые таким образом, будем называть агрегированными. Используя выражение (2.1), запишем критерий C_a, который будем называть

суммарное время завериения агрегированных работ, следующим образом:

$$C_a = \sum_{j=1}^{n} \max_{k=\underline{I,S}} \left(C_{jk}\right) \tag{2.2}$$

Наряду с C_a будем рассматривать такие критерии как суммарное время завершения задач $\sum_{j=1}^{n} \sum_{k=l}^{S} C_{jk}$ и минимальная длина расписания $C_{\max}$. В главе 3 будет предложен алгоритм, минимизирующий длину расписания по каждому из этих критериев.

Используя $\alpha|\beta|\gamma$ нотацию, введенную в работах Грехема и др. [73] и Блажевича и др. [74], можем записать задачу в одной из следующих форм в зависимости от выбранного критерия оптимальности:

$$\left\langle P_m \mid p_{ij} = \mu_{i_k} V_{ji_k}, \text{res}S..L\mu_s \mid C_a \right\rangle, \tag{2.3}$$

$$\left\langle P_m \mid p_{ij} = \mu_{i_k} V_{ji_k}, \text{res}S..L\mu_s \mid \sum_{j=1}^{n} \sum_{k=1}^{S} C_{jk} \right\rangle, \tag{2.4}$$

$$\left\langle P_m \mid p_{ij} = \mu_{i_k} V_{ji_k}, \text{res}S..L\mu_s \mid C_{\max} \right\rangle. \tag{2.5}$$

где $\alpha = \{P_m\}, \beta = \left\{p_{ij} = \mu_{i_k} V_{ji_k}, \text{res}S..L\mu_s\right\}$, $\gamma = \{C_a\}$ либо $\gamma = \left\{\sum_{j=1}^{n} \sum_{k=1}^{S} C_{jk}\right\}$, либо $\gamma = \{C_{\max}\}$

Заметим, что задача (2.5) является расширением задачи $\langle P_2||C_{\max}\rangle$, NP-трудность которой была показана в [43] через сведение к задаче о разделении.

Задача (2.4) является расширением задачи

$$\left\langle P_2 \left\| \sum_{j=1}^{n} w_j C_j \right\rangle \right. \tag{2.6}$$

для случая $w_j = 1, \forall j$ и $S = 1$. NP-полнота (2.6) была показана в [132] через сведение к задаче о ранце. Следовательно, задачи (2.4), (2.5) также являются NP-трудными. Можно сделать предположение о NP-трудности задачи (2.3).

На рисунке 2.2 изображена схема работы СА и поступление в неё новых задач.

Также введём следующие обозначения. Если задача T_{js} выполняется в данный момент времени t, то

$$\text{PR}_{js}(t) = \begin{cases} 1, & \text{если задача } T_{js} \text{ выполняется в момент времени } t; \\ 0, & \text{иначе.} \end{cases}$$

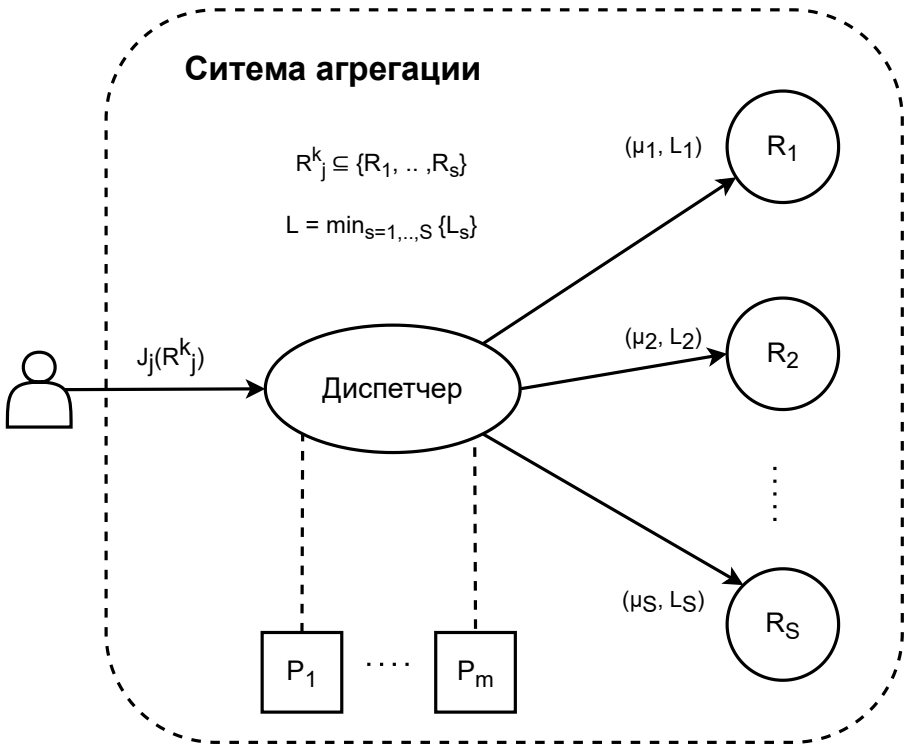


Рисунок 2.2 — Схема работы СА.

Тогда $B_s(t) = \sum_{j=1}^{n} \mathrm{PR}_{js}$ будет показывать количество задач и приборов, одновременно обращающихся к ресурсу R_s. Введём ограничение $B_s(t) \leq L, L \in \mathbb{N}$ — квота на число одновременных соединений.

Очевидно, что $\sum_{s=1}^{S} B_s(t) \leq m$, числу доступных приборов в системе, $B_s(t) \in 0, \ldots, m$.

Таблица 3 — Список переменных математической модели

Обозначение	Название
$J_j,\ j = 1, \ldots, n$	работы
$T_{ji},\ j = 1, \ldots, n,\ i \in i_1, \ldots i_k$	задачи
$R_s,\ s = 1, \ldots, S$	ресурсы
s_j, s_{ji}	времена начала работ и задач
r_j, r_{ji}	времена готовности задачи
p_{ji}	оценка времени выполнения задачи
V_{ji}	ресурсоемкость задачи
μ_s	скорость доступа к ресурсу
$\mathrm{PR}_{js}(t)$	выполнение задачи T_{js} в момент времени t
$B_s(t)$	количество приборов, одновременно

Продолжение на следующей странице

Обозначение	Название
L	ограничение одновременного доступа к ресурсу
$P_l, l = 1, \ldots, m$	приборы

Тогда задачу (2.3) можно записать в виде задачи нелинейного программирования следующим образом:

$$\begin{cases} C_a = \sum\limits_{j=1}^{n} \max\limits_{k=1,\ldots,s} C_{jk} \to \min \\ C_{ji} = s_{ji} + p_{ji} \\ p_{ji} = \mu_i V_{ji} \\ s_{ji} \geq r_{ji} \\ s_j = \min\limits_{i=1,\ldots i_k} s_{ji} \\ r_{ji} = r_j, \forall i \\ \mathrm{PR}_{js}(t) = \begin{cases} 1, \text{если } T_{js} \text{ выполняется в момент времени } t \\ 0, \text{иначе} \end{cases} \\ B_s(t) = \sum\limits_{j=1}^{n} \mathrm{PR}_{js} \leq L \\ \sum\limits_{s=1}^{S} B_s(t) \leq m,\ B_s(t) \in 0, \ldots, m \\ L \in \mathbb{N}, m \in \mathbb{N}, \mu_s \in \mathbb{N}^s, V \in \mathbb{N}^{n \times s}, r_j \in \mathbb{N}, t \in \mathbb{N} \\ j \in \overline{1, n}, k \in \overline{1, S}, i \in \overline{1, k} \end{cases} \tag{2.7}$$

Перечень обозначений модели приведён в таблице 3.

Пусть имеет место случай $L = 1$, т. е. одновременно одному прибору может быть доступен только один ресурс. Рассмотрим случай, когда $m = S$, т. е. мощность множества ресурсов совпадает с мощностью множества приборов. Покажем, что в этом случае оптимальной будет обработка запросов на выделенных приборах.

***Теорема* 2.1.** *Пусть для модели* (2.7) $L = 1$, $m = S$ *и выполнено*

$$\nexists x \in \mathbb{N} : \left(\frac{\mu_s}{\mu_{s'}}\right)^{\kappa},\ s' \neq s'',\ s', s'' \in 1, \ldots, S \tag{2.8}$$

где $\kappa \in \{-1, 1\}$, *времена готовности* $r_{ij} = 0, \forall i, j$, *а ресурсоёмкость задач задана бинарной матрицей* V *размерности* $n \times s$. *Тогда допустимыми будут являться только те расписания, в которых на* $\forall$ *прибор* P_l *могут быть назначены только задачи из множества* $T_l = \left\{T_{js} : s = l, \forall j = \overline{1, n}\right\}$, $l = 1, \ldots, m$.

Доказательство. Произведём доказательно по индукции.

1. Докажем, что утверждение теоремы выполнено для числа приборов $m = 2$. Пусть даны приборы P_1 и P_2, ресурсы R_1 и R_2 и работы $J_1, \ldots, J_n$, декомпозируемые на задачи T_{ji} с $r_{ji} = 0, \forall\, i, j$. Условие (2.8) будет иметь вид:

$$\nexists x \in \aleph : \left(\frac{\mu_1}{\mu_2}\right)^{\kappa} \tag{2.9}$$

Пусть в момент времени $t = 0$ были выбраны в обработку задачи $T_{j_1 1}$ и $T_{j_2 2}, \forall j_1, j_2$ для обработки на приборах P_1 и P_2, соответственно. Время обработки задач составит $p_{j_1 1} = \mu_1$ и $p_{j_2 2} = \mu_2$. Из условия (2.9) понятно, что одна из величин μ_s больше другой.

Примем, что $\mu_1 < \mu_2$. Тогда обработка задачи $T_{j_1 1}$ на приборе P_1 завершится раньше, чем обработка задачи $T_{j,2}$ на приборе P_2. Пока есть необработанные задачи, разумным с точки зрения минимизации времени выполнения работ является назначение одной из необработанных задач на освободившийся прибор без простоя. При этом, из-за того, что выполнение задачи $T_{j,2}$ блокирует доступ к ресурсу R_2, выбор задач для назначения на прибор P_1 может осуществляться только среди задач T_{j1}, использующих ресурс R_1.

Поскольку выполнено (2.9), то не будет такого момента времени, когда времена завершения задач на приборах P_1 и P_2 совпадут. А значит, приборы окажутся выделенными под обработку задач, требующих определённого типа ресурсов. Тем самым доказана верность утверждения для $m = S = 2$.

2. Пусть теперь утверждение теоремы выполнено для $m = s = l - 1$ приборов и ресурсов. Покажем, что утверждение также будет верным для $m = s = l$ приборов и ресурсов.

Если наиболее оптимальным режимом обслуживания задач для $l - 1$ прибора является обслуживание задач, требующих одного определённого ресурса, то с учетом условия (2.8), не будет такого момента времени, когда на прибор P_l можно будет назначить задачу, требующую одного из ресурсов $R_1, \ldots, R_{l-1}$. Соответственно, задачи на этот прибор могут быть назначены только из множества

$$T_l = \left\{T_{js} : s = l, \forall j = \overline{1, n}\right\},$$

и утверждение теоремы верно для $\forall l \in 1, \ldots, m,\ m \in \mathbb{N}$. □

***Следствие* 2.1.** *Пусть выполнены все условия теоремы, но $m > S$. Тогда при $L = 1$ будут простаивать $m - S$ приборов.*

Доказательство. Пусть приборы перенумерованы от 1 до m, и задачи на каждый l-й прибор назначаются из множества $T_l = \{T_{js} : S = l, j = 1, \ldots, n\}$, где $l = 1,..., S$. Тогда для $l > S$ данное множество T_l окажется пустым. □

Дадим краткий комментарий на предмет сводимости рассматриваемой задачи к задачам цеха (shop problems) в в случае $m = S$ и $L = 1$. С одной стороны, в данном случае можно рассматривать задачи, составляющие работу, как операции, выполняемые на определённых приборах, причем связанные по ресурсам операции T_{js} и T_{is} не могут быть выполнены одновременно $\forall i, j$. Отсутствование отношений предшаствования между задачами говорит о том, что данная задача не может быть классифицировна как job shop или flow shop задача. Тем не менее, задача (2.3) нарушает общее требование к задачам цеха о том, что две операции одной и той же работы не могут выполняться одновременно. Более того, если мы говорим о минимизации критерия C_a, то для нас может быть желательно, чтобы некоторые их требований, принадлежащих к одной работе, выполнялись параллельно на разных приборах. В связи с этим задача (2.3) не сводится к задаче цеха в общем виде, но является её расширением.

2.2 Дискретно-непрерывная модель многолинейной системы с ограниченными восполнимыми ресурсами для систем агрегации

В информационных системах, помимо ресурсов с ограничением одновременного доступа встречаются также ресурсы с практически неограниченным параллельным доступом и возможностью разделения ресурса между несколькими приборами (т. е. клиентами) в произвольной пропорции. Ограничение на объём ресурса выразим в виде $\sum_i u_i(t) \le U$, т. е. в каждый момент времени невозможно зарезервировать суммарно под все задачи объём ресурса, больший, чем его масимальный доступный объём. Таким образом, в дополнение к дискретной модели (2.7) сформулируем дискретно-непрерывную модель многолинейной системы

с ограниченными восполнимыми ресурсами для СА:

$$\begin{cases} C_a = \sum\limits_{j=1}^{n} \max\limits_{k=1,\ldots,s} C_{jk} \to \min \\ C_{ji} = s_{ji} + p_{ji} \\ s_{ji} \geq r_{ji} \\ s_j = \min\limits_{i=1,\ldots i_k} s_{ji} \\ r_{ji} = r_j, \forall i \\ v_{ji}(t) = \dfrac{dx_{ji}(t)}{dt} = \tilde{v}_{ji}\big(u_{ji}(t)\big) \\ X_{ji} = \int\limits_{s_{ji}}^{C_{ji}} \tilde{v}_{ji}\big(u_{ji}(t)\big)dt \\ \sum\limits_{j=1}^{n}\sum\limits_{i=1}^{S} u_{ji}(t) \leq U \\ m \in \mathbb{N}, j \in \overline{1,n}, k \in \overline{1,S}, i \in \overline{1,k} \end{cases} \tag{2.10}$$

Перечень обозначений модели приведён в таблице 4.

Таблица 4 — Список переменных математической модели (2.10)

Обозначение	Название
t	непрерывное время модели, $t \in [0; +\inf)$
J_j	работы, $j = 1, \ldots, n$
T_{ji}	задачи, $j = 1, \ldots, n, i \in i_1 \ldots i_k$
R_s	ресурсы, $s = 1, \ldots, S$
P_l	приборы, $l = 1, \ldots, m$
r_j, r_{ji}	времена готовности задачи
s_j, s_{ji}	времена начала работ и задач
C_j, C_{ji}	времена завершения работ и задач
$p_{ji} = C_{ji} - s_{ji}$	времена выполнения задач
X_{ji}	объёмы задач
$x_{ji}(t)$	объёмы задач, выполненные к моменту времени t
$v_{ji}(t)$	скорость выполнения задачи

Продолжение на следующей странице

Обозначение	Название
$u_{ji}(t)$	объём ресурса, используемый для выполнения задачи в момент времени t
U	доступный объём ресурса (полностью возобновляемый ресурс)

Предполагаем, что $\tilde{v}_i(u_i(t))$ — строго возрастающая функция, т. е. увеличение объёма используемого ресурса всегда ускоряет выполнение заданий. Также считаем, что $\forall i \ : \ \tilde{v}_i(0) \ = \ 0$ — т. е. без нужного ресурса обслуживание задачи невозможно.

В дальнейшем применительно к сравнению функций отношение $f(x) \leq g(x)$ будет обозначать $\forall x : \neg\big(f(x) > g(x)\big), \exists x_0 \in D(f) \cup D(g) : f(x_0) > g(x_0)$, в противоположность $f(x) \leqq g(x)$, не предполагающему условия $\exists x_0 : f(x_0) > g(x_0)$.

Анализ модели потребует сравнивать время выполнения задач при различных вариантах выделения ресурса с течением времени. Для проведения такого сравнения докажем:

***Лемма* 2.1.** *Если объёмы задач равны:*

$$\int_{C_0}^{C} \tilde{v}\big(u(t)\big)\,dt = \int_{C_0}^{C} \tilde{v}\big(u'(t)\big)\,dt, \ \ u \in [0, +\infty),$$

а зависимость скорости выполнения задачи от объёма выделенного ресурса $\tilde{v}(u)$ неотрицательна и монотонно возрастает, то для выделения ресурса $u(t) \leq u'(t)$ верно соотношение времён завершения задач $C > C'$.

Доказательство. По определению возрастающей функции

$$u(t_0) = u'(t_0) \Rightarrow \tilde{v}\big(u(t_0)\big) = \tilde{v}\big(u'(t_0)\big) \Rightarrow v(t_0) = v'(t_0),$$
$$u(t_1) < u'(t_1) \Rightarrow \tilde{v}\big(u(t_1)\big) < \tilde{v}\big(u'(t_1)\big) \Rightarrow v(t_1) < v'(t_1).$$

Следовательно,

$$u(t) \leq u'(t) \Rightarrow v(t) \leq v'(t).$$

Пусть $v'(t) = v(t) + \Delta v(t)$. Предположим, что $C \leq C'$. Тогда

$$\int_{C_0}^{C'} v'(t)dt = \int_{C_0}^{C'} v(t)dt + \int_{C_0}^{C'} \Delta v(t)dt =$$
$$= \int_{C_0}^{C} v(t)dt + \int_{C}^{C'} v(t)dt + \int_{C_0}^{C'} \Delta v(t)dt.$$

Т. к.

$$C \le C' \wedge v(t) \ge 0 \quad \Rightarrow \int_{C}^{C'} v(t)dt \ge 0,$$

$$C_0 < C' \wedge \Delta v(t) \ge 0 \Rightarrow \int_{C_0}^{C'} \Delta v(t)dt > 0,$$

то

$$\int_{C_0}^{C'} v'(t)dt > \int_{C_0}^{C} v(t)dt,$$

что противоречит условию.

Следовательно, $C > C'$. □

Выполненная часть задачи зависит от времени как

$$x_k(t) = \int_0^t \tilde{v}_k\big(u_k(t)\big)dt.$$

Таким образом, общий объём задачи можно выразить как

$$X_k = \int_0^{C_k} \tilde{v}_k\big(u_k(t)\big)dt.$$

Пусть времена прихода задач в систему $r_1 = r_2 = \ldots = r_n = 0$. Для двух задач, случай $r_1 < r_2$ можно свести к $r_1 = r_2 = 0$, разбив задачу T_1 на две подзадачи: T_1^1 с $r_1^1 = r_1, C_1^1 = r_2$, и T_1^2 с $r_1^2 = r_2$, и приняв $r_2' = 0$, $r_1' = r_1 - r_2$ (см. рисунок 2.3). Аналогично по индукции можно сделать для произвольного числа задач.

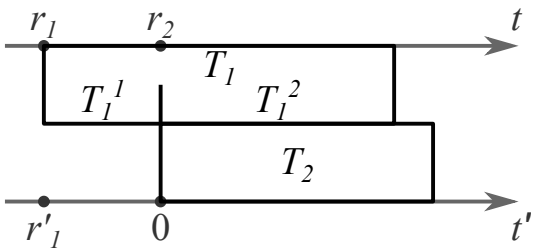


Рисунок 2.3 — Сведение случая $r_1 < r_2$ к $r_1' = r_2' = 0$.

Начнём с рассмотрения случая параллельной многозадачной системы с неограниченным числом приборов, единственным ресурсом и неограниченным параллельным доступом к нему. Далее, изучив данный базовый случай, расширим полученные результаты на случаи ограниченного числа приборов, более чем одного ресурса и декомпозиции работ на задачи.

В качестве критериев оптимальности будем рассматривать критерии $C_{\max} = \max(C_1, \ldots, C_n)$ и $\sum C_i = C_1 + \ldots + C_n$. Докажем лемму:

***Лемма* 2.2.** *Для расписания, минимизирующего значения критериев* $\sum C_i$ *и* $C_{\max}$*, в любой момент времени* t *выполняется* $\sum_i u_i(t) = U$.

Доказательство. Начнём рассмотрение с детерминированной системы, на вход которой поступило $n = 2$ задания.

Пусть ресурсы распределены так, что $C_1 \le C_2$. Если $C_1 > C_2$, то случай сводится к рассматриваемому заменой индексов.

Тогда

$$\begin{aligned} X_1 + X_2 &= \int_0^{C_1} \tilde{v}_1\big(u_1(t)\big)dt + \int_0^{C_2} \tilde{v}_2\big(u_2(t)\big)dt = \\ &= \int_0^{C_1} \tilde{v}_1\big(u_1(t)\big)dt + \int_0^{C_1} \tilde{v}_2\big(u_2(t)\big)dt + \int_{C_1}^{C_2} \tilde{v}_2\big(u_2(t)\big)dt = \\ &= \int_0^{C_1} \Big[\tilde{v}_1\big(u_1(t)\big) + \tilde{v}_2\big(u_2(t)\big)\Big]dt + \int_{C_1}^{C_2} \tilde{v}_2\big(u_2(t)\big)dt. \end{aligned}$$

Из Леммы 2.1 очевидным образом следует, что для минимизации обоих критериев оптимальности $C_{\max}$ и $\sum C_i$ выгодно выбирать

$$\begin{cases} u_1(t) + u_2(t) = U, & 0 \le t \le C_1, \\ u_2(t) = U, & C_1 < t \le C_2. \end{cases}$$

По индукции полученные результаты обобщаются на случай $n > 2$ заданий.

□

Рассмотрим три случая для зависимости скорости выполнения задачи от объёма выделенного ресурса: когда $\tilde{v}_i\big(u_i(t)\big)$ — линейная функция, т. е. $v_i(t) = a_i i(t)$, когда $\tilde{v}_i\big(u_i(t)\big)$ — выпуклая, и когда $\tilde{v}_i\big(u_i(t)\big)$ — вогнутая функция.

Линейный случай.

Рассмотрим случай линейной зависимости $v_i(t) = a_i u_i(t)$. Для него докажем теорему:

***Теорема* 2.2.** *Минимальные значения критериев оптимальности*

$$\begin{aligned} C_{\max} &= X_i/U, \\ \sum_i C_i &= (n + 1 - i)X_i/U \end{aligned}$$

достигаются при строго последовательном выполнении задач в порядке $X_1 < X_2 < \ldots < X_n$*, т. е. при использовании правила диспетчеризации SPT (кратчайшее время выполнения).*

Доказательство. Переобозначив объём задачи X_i (и, следовательно, x_i и v_i) в других единицах измерения, можно свести задачу к $v_i(t) = u_i(t)$, каковой случай и будем рассматривать для упрощения расчётов.

Начнём рассмотрение со случая $n = 2$ задач. Имеем

$$\begin{aligned} X_1 &= \int_0^{C_1} u_1(t)dt = x_1(t)\Big|_0^{C_1} = x_1(C_1), \\ X_2 &= \int_0^{C_1} \Big[U - u_1(t)\Big]dt + \int_{C_1}^{C_2} U dt = \\ &= \Big[Ut - x_1(t)\Big]\Big|_0^{C_1} + Ut\Big|_{C_1}^{C_2} = UC_2 - x_1(C_1). \end{aligned}$$

Таким образом,

$$\begin{cases} X_1 = x_1(C_1), \\ X_2 + X_2 = UC_2. \end{cases}$$

Получаем, что $C_2 = (X_1 + X_2)/U$ не зависит от выбора (u_1, u_2) при $t < C_1$, пока $u_1 + u_2 = U$. Тогда из Леммы 2.1 следует, что для минимизации C_1 оптимальным выбором будет $u_1(t) = U, u_2 = 0$, т. е. задачи выполняются строго последовательно. Следовательно, минимальное значение $C_1 = X_1/U$.

При таком выборе (u_1, u_2) значения критериев оптимальности равны

$$\begin{cases} C_{\max} = \dfrac{X_1 + X_2}{U}, \\ \sum C_i = \dfrac{2X_1 + X_2}{U}. \end{cases}$$

При этом если $X_1 < X_2$, то $(2X_1 + X_2)/U < (2X_2 + X_1)/U$, следовательно, первой для выполнения следует выбирать задачу с меньшим объёмом.

Расширим доказательство на случай произвольного количества задач с линейной зависимостью $v_i(t) = u_i(t)$.

Рассмотрим выражение

$$\sum_{i=1}^{n} X_i = \int_0^{C_1} \sum_{i=1}^{n} u_i(t)dt + \int_{C_1}^{C_2} \sum_{i=2}^{n} u_i(t)dt + \ldots + \int_{C_{n-1}}^{C_n} u_n(t)dt.$$

Согласно Лемме 2.1, для минимизации C_n оптимальным выбором будет

$$U = \sum_{i=1}^{n} u_i(t)\Big|_{0 \le t \le C_1} = \sum_{i=2}^{n} u_i(t)\Big|_{C_1 < t \le C_2} = \ldots = u_n(t)\Big|_{C_{n-1} < t \le C_n},$$

откуда следует

$$\sum_{i=1}^{n} X_i = UC_n.$$

Далее, из

$$X_n = \int_0^{C_{n-1}} u_n(t)dt + \int_{C_{n-1}}^{C_n} Udt = \int_0^{C_{n-1}} u_n(t)dt + UC_n - UC_{n-1},$$

откуда следует, что

$$C_{n-1} = U^{-1}\left(\sum_{i=1}^{n-1} X_i + \int_0^{C_{n-1}} u_n(t)dt\right),$$

т. е. для минимизации C_{n-1} следует выбрать

$$\begin{cases} u_n = U, & C_{n-1} < t \le C_n, \\ u_n = 0, & t \le C_{n-1}, \end{cases}$$

и тогда

$$C_{n-1} = \sum_{i=1}^{n-1} X_i/U.$$

В этом случае

$$\sum_{i=1}^{n-1} X_i = \int_0^{C_1} \sum_{i=1}^{n-1} u_i(t)dt + \int_{C_1}^{C_2} \sum_{i=2}^{n-1} u_i(t)dt + \ldots + \int_{C_{n-2}}^{C_{n-1}} u_{n-1}(t)dt.$$

Повторяя по индукции предыдущие рассуждения, получаем, что оптимальным выбором для u_i является

$$\begin{cases} u_i = U, & C_{i-1} < t \le C_i, \\ u_i = 0, & t \le C_{i-1}, \end{cases}$$

и в этом случае достигается минимальное значение

$$C_i = \sum_{j=1}^{i} X_j/U.$$

Таким образом, для произвольного количества задач минимальные значения критериев оптимальности

$$\begin{cases} C_{\max} = \sum_{i=1}^{n} X_i/U, \\ \sum C_i = \sum_{i=1}^{n} (n+1-i)X_i/U. \end{cases} \tag{2.11}$$

Они достигаются при строго последовательном выполнении задач в порядке $X_1 < X_2 < \ldots < X_n$, т. е. использовании правила диспетчеризации SPT. □

Вогнутый случай.

Рассмотрим случай вогнутой функции $\tilde{v}_i(u_i)$. В реальных задачах такая ситуация встречается, когда одновременный доступ к ресурсу предполагает дополнительные накладные расходы; в частности, когда внутреннее устройство ресурса использует принцип вытесняющей многозадачности. Примеры задач с таким поведением — передача данных по сетевому протоколу IP, чтение с HDD-диска и т. д.

Докажем теорему:

***Теорема* 2.3.** *Минимальные значения критериев оптимальности*

$$C_{\max} = X_i/U,$$
$$\sum_i C_i = (n+1-i)X_i/U$$

достигаются при строго последовательном выполнении задач в порядке $X_1 < X_2 < \ldots < X_n$*, т. е. при использовании правила диспетчеризации SPT (кратчайшее время выполнения).*

Доказательство. Выбором единиц измерения для объёма задачи можно преобразовать функцию $\tilde{v}_i(u_i)$ к

$$\begin{cases} v(t) \le u(t), & 0 \le u(t) \le U, \\ v(t) = 0, & u(t) = 0, \\ v(t) = u(t), & u(t) = U. \end{cases}$$

Сравним случаи

$$\begin{cases} 0 \le v_1(t) \le U, & 0 \le t \le C_1, \\ v_1(t) = 0, & C_1 < t \le C_2, \\ 0 \le v_2(t) \le U, & 0 \le t \le C_1, \\ v_2(t) = U, & C_1 < t \le C_2. \end{cases}$$

и

$$\begin{cases} v_1'(t) = U, & 0 \le t \le C_1', \\ v_1'(t) = 0, & C_1' < t \le C_2', \\ v_2'(t) = 0, & 0 \le t \le C_1', \\ v_2'(t) = U, & C_1' < t \le C_2'. \end{cases}$$

Отсюда имеем $v_1(t) \le U = v_1'(t)$ для $[0, C_1] \cap [0, C_1']$ и $v_1(t) + v_2(t) \le u_1(t) + u_2(t) = U = v_1'(t) + v_2'(t)$ для $[0, C_2] \cap [0, C_2']$. Принимая $\tilde{v} = \mathbb{1}$ в Лемме 2.1, из

$$X_1 = \int_0^{C_1} v_1(t)dt = \int_0^{C_1'} v_1'(t)dt,$$

$$X_1 + X_2 = \int_0^{C_2} \Big[v_1(t) + v_2(t)\Big]dt = \int_0^{C_2'} \Big[v_1'(t) + v_2'(t)\Big]dt$$

получаем $C_1 > C_1'$, $C_2 > C_2'$.

Таким образом, минимальные значения критериев $C_{\max} = C_2'$ и $\sum C_i = C_1' + C_2'$ достигаются при строго последовательном выполнении задач. Т. к. в этом случае величины критериев равны значениям для линейного случая, выгода использования критерия SPT доказывается аналогично.

Расширим доказательство на случай произвольного количества задач, сравнив случаи

$$\begin{cases} 0 \le v_i(t) \le U, & 0 \le t \le C_i, \\ v_i(t) = 0, & t > C_i, \end{cases}$$

и

$$\begin{cases} v_i'(t) = 0, & 0 \le t \le C_{i-1}', \\ v_i'(t) = U, & C_{i-1}' < t \le C_i', \\ v_i'(t) = 0, & t > C_i'. \end{cases}$$

Для $[0, C_i] \cap [0, C_i']$ выполняется

$$\sum_{j=1}^{i} v_j(t) \le \sum_{j=1}^{i} u_j(t) \le U = \sum_{j=1}^{i} v_j'(t).$$

Так как

$$\sum_{j=1}^{i} X_i = \int_0^{C_i} \sum_{j=1}^{i} v_j(t)dt = \int_0^{C_i'} \sum_{j=1}^{i} v_j'(t)dt,$$

согласно Лемме 2.1 $C_i > C_i'$, и наилучшие значения критериев оптимальности $C_{\max} = C_n'$ и $\sum C_i = \sum_{i=1}^{n} C_i'$ равны значениям для линейного случая.

Итого, с точки зрения рассматриваемых критериев оптимальности вогнутый случай аналогичен линейному: значения критериев (2.11) достигаются при строго последовательном выполнении задач с использованием правила диспетчеризации SPT. □

Выпуклый случай.

Рассмотрим случай выпуклой функции $\tilde{v}_i(u_i)$. В реальных задачах такая ситуация встречается, когда использование ресурса демонстрирует насыщение; в частности, когда внутреннее устройство ресурса использует параллельную обработку на нескольких приборах. Пример задачи с таким поведением — выгрузка вычислений на сопроцессор.

Докажем теорему:

***Теорема* 2.4.** *В случае выпуклых функций наилучшие значения критериев оптимальности $C_{\max}$ и $\sum_i C_i$ достигаются при резервировании ресурсов, зависящем от формы функций $\tilde{v}_i(u_i)$ для конкретной задачи.*

Доказательство. Выбором единиц измерения для объёма задачи можно преобразовать функцию $\tilde{v}_i(u_i)$ к

$$\begin{cases} v(t) \geq u(t), & 0 \leq u(t) \leq U, \\ v(t) = 0, & u(t) = 0, \\ v(t) = u(t), & u(t) = U. \end{cases}$$

Начнём с рассмотрения случая, когда $\tilde{v}_i\big(u_i(t)\big) = v_i(u_i)$ не зависит от времени t. Тогда из соображений симметрии оптимальный случай должен достигаться при $u_i = \text{const}$ на каждом из интервалов $(C_{i-1}, C_i]$.

Рассмотрим детерминированную систему, на вход которой поступило $n = 2$ задания. Тогда

$$\begin{cases} X_1 = \displaystyle\int_0^{C_1} \tilde{v}_1(u_1)dt = \tilde{v}_1 C_1, \\ X_1 + X_2 = \displaystyle\int_0^{C_1} \Big[\tilde{v}_1(u_1) + \tilde{v}_2(u_2)\Big]dt + \int_{C_1}^{C_2} U dt = \\ \qquad = (\tilde{v}_1 + \tilde{v}_2)\, C_1 + U\,(C_2 - C_1)\,; \end{cases}$$

$$\begin{cases} C_1 = \dfrac{X_1}{\tilde{v}_1}, \\ C_{\max} = C_2 = \dfrac{X_2}{U} + \dfrac{X_1}{U}\left[\dfrac{U - \tilde{v}_2}{\tilde{v}_1}\right], \\ \sum C_i = C_1 + C_2 = \dfrac{X_2}{U} + \dfrac{X_1}{U}\left[\dfrac{2U - \tilde{v}_2}{\tilde{v}_1}\right]. \end{cases}$$

При последовательном выполнении задач $C_{\max} = \dfrac{X_1 + X_2}{U}$. Так как $\tilde{v}_1(u_1) + \tilde{v}_2(u_2) \geq u_1(t) + u_2(t) = U$, то $\dfrac{U - \tilde{v}_2}{\tilde{v}_1} \leq 1$. Следовательно, оптимальное значение $C_{\max}$ достигается при выборе $u_2 > 0$ для $t \leq C_1$, т. е. при параллельном выполении первой и второй задачи. Для нахождения конкретного резервирования ресурсов, соответствующего оптимальному $C_{\max}$, необходимо найти минимум функции $\dfrac{U - \tilde{v}_2(U - u_1)}{\tilde{v}_1(u_1)}$, который зависит от формы функций $\tilde{v}_1$ и $\tilde{v}_2$ для конкретной задачи.

При последовательном выполнении задач $\sum C_i = \dfrac{2X_1 + X_2}{U}$. Чтобы $\sum C_i$ при параллельном выполнении задач был лучше, чем при последовательном, необходимо выполнение условия $\dfrac{2U - \tilde{v}_2}{\tilde{v}_1} < 2$, т. е. $2U < 2\tilde{v}_1 + \tilde{v}_2$. Т. к. $\tilde{v}_1 + \tilde{v}_2 > U$, но $\tilde{v}_1 < U$, выполнение этого условия зависит от выбора функций $\tilde{v}_1$ и $\tilde{v}_2$.

Таким образом, показано, что в случае выпуклых функций для минимизации критериев оптимальности, наилучшие значения $C_{\max}$ и $\sum C_i$ достигаются при резервировании ресурсов, зависящих от формы функций $\tilde{v}_i(u_i)$ для конкретной задачи. Тем более это верно для $\tilde{v}_i\big(u_i(t)\big)$, зависящих от времени.

Аналогично находятся функционалы, минимизация которых даст наилучшие значения $C_{\max}$ и $\sum C_i$ для числа задач, большего 2. □

Рассмотрение конкретных задач с выпуклыми $\tilde{v}_i$ выходит за рамки данной дисертационной работы.

Обобщения модели на случаи ограниченного числа приборов, более чем одного ресурса и декомпозиции работ на задачи.

Рассмотрим обобщение модели на случай с ограниченным числом приборов. Так как для линейного и вогнутого случая доказано, что наилучшие значения критериев оптимальности достигаются при последовательном выполнении задач, возможно задействовать для этого один прибор — таким образом, ограничение на количество приборов не имеет значения. Для выпуклого случая при поиске минимумов функционалов необходимо принимать во вниманиемаксимальное количество параллельно выполняемых задач.

Рассмотрим обобщение модели на случай более чем одного ресурса, если каждая задача использует только один ресурс. Если число приборов больше или равно числу ресурсов, можно выделить каждому ресурсу свой прибор — таким образом, модель разбивается на несколько дочерних моделей по числу ресурсов. Для каждой из дочерних моделей верны рассуждения, изложенные для случая с

единственным ресурсом. Если число приборов меньше числа ресурсов, лимитирующем фактором модели становятся не ресурсы, а приборы. В этом случае приборы можно рассматривать как новый вид дискретного ресурса, таким образом сведя модель к рассмотренному ранее случаю с дискретными ресурсами.

Рассмотрим обобщение модели на случай более чем одного ресурса, если для работы может требоваться более чем один ресурс, с декомпозицией работ на задачи, использующие только один ресурс. Расчёт критериев оптимальности $C_{\max}$ и $\sum C_i$ в этом случае не отличается от предыдущих, но также становится возможным определить новый критерий оптимальности C_a.

Будем рассматривать случаи с линейными и вогнутыми $\tilde{v}_i\left(u_i(t)\right)$ — как показано ранее, алгоритм нахождения критериев оптимальности для выпуклого случая зависит от конкретной функции $\tilde{v}_i\left(u_i(t)\right)$, а рассмотрение частных случаев выходит за рамки данной работы.

Так как

$$C_a = \sum_{j=1}^{n} \max_{k=1,\ldots,s} C_{jk},$$

для минимизации C_a необходимо минимизировать $\max\limits_{k=1,\ldots,s} C_{jk}$ для каждого j — и, следовательно, наибольшее из времён C_{jk} для каждого j. Пусть для $j = j_0$ достигается $\max\limits_{k=1,\ldots,s} C_{jk}$ для задачи, использующей ресурс $k = k_0$. Рассмотрим множество задач $\left\{C_{jk_0}\right\}$, сведя таким ситуацию для $k = k_0$ к рассмотренным ранее моделям с одним ресурсом. Как было показано ранее, для линейного и вогнутого случая значения C_i минимизируются при последовательном использовании ресурса. Соответственно, минимальное значение C_a также достигается при последовательном использовании ресурсов.

Таким образом, модель сводится к рассмотренной ранее модели с дискретными ресурсами, и оптимальный порядок выполнения задач и обращения к ресурсам определяется на основе данной модели.

2.3 Построение приоритето-порождающего функционала системы с ограниченными восполнимыми ресурсами

Будем искать решение задач, сформулированных в ч. 2.1-2.2 в терминах нахождения экстремума приоритето-порождающего функционала (ППФ), заданного на множестве перестановок конечного множества заявок на обслуживание.

Понятие ППФ было впервые введено в [133] и активно изучалось в работах Минской школы Теории расписаний такими её представителями, как Я.М. Ша-

франский, В.С. Танаев, В.С. Гордон. В отечественной литературе использования ППФ можно найти в работах Л.Б. Нисневича, Г.А. Котюжанского, Г.Г. Стецюры и соавторов. Использование приоритето-порождающих функционалов 1-го порядка приводит к правилам диспетчеризации, активно исследуемых в современных работах отечетсвенных [19; 134] и зарубежных [135—137] специалистов.

Введем в рассмотрение определения, необходимые для построения такого функционала.

Необходимые понятия и определения

***Определение* 8.** *Пусть дано* $A = \{a_1, a_2, \ldots, a_N\}$, $|A| = N$. *Будем называть упорядоченную последовательность (т. е. выборку)* $k \leq N$ *элементов множества A, не содержащую повторений, перестановкой длины k из элементов A. Перестановка будет называться частичной, если* $k < N$, *либо полной, если* $k = N$.

Будем символически записывать перестановку как $\pi = \left(a_{i_1}, a_{i_2}, \ldots, a_{i_x}\right)$, где a_{i_r} — это элемент множества A, расположенный в перестановке π на r-ом месте слева.

Также, в случае, если природа элементов не существенна для задачи, возможно работать не с самими элементами множества , а с их индексами, и использовать запись $\pi_x = (i_1, i_2, \ldots, i_x)$, где под i_r понимается индекс (номер) элемента множества , находящийся в перестановке π_x на r-ом месте слева.

Перестановка π_x в данном случае будет являться перестановкой длиной x, определённой на множестве $\{1,2, \ldots, N\}$ индексов элементов множества A. Пусть даны перестановки элементов множества : $\pi^{(1)} = \left(a_{i_1}, a_{i_2}, \ldots, a_{i_y}\right)$, а $\pi^{(2)} = \left(a_{j_1}, a_{j_2}, \ldots, a_{j_z}\right)$, для которых выполнено $\pi^{(1)} \cap \pi^{(2)} = \emptyset$. Тогда под перестановкой $\pi = \left(\pi^{(1)}, \pi^{(2)}\right)$ будем понимать $\pi = \left(a_{i_1}, a_{i_2}, \ldots, a_{i_y}, a_{j_1}, a_{j_2}, \ldots, a_{j_z}\right)$.

Пусть $\widehat{\Pi}$ — множество всех перестановок π_k длины $k = 0, \ldots, N$ элементов множества индексов $\{1,2, \ldots, \mathrm{N}\}$, заданных на множестве . Определим $\{\pi_0\} = \emptyset$.

Пусть $\Pi, \bar{\Pi} \subseteq \widehat{\Pi}$. Через $Q(\Pi)$ обозначим множество перестановок $\pi^{(q)} \in \widehat{\Pi}$, таких что существуют перестановки $\pi \in \Pi$ и $\pi', \pi'' \in \widehat{\Pi}$ такие, что $\pi = \left(\pi', \pi^{(q)}, \pi''\right)$. Определим на множестве $Q(\Pi)$ функционал $\omega(\pi)$.

***Определение* 9.** *Если для функционала* $\Phi(\pi)$, *определенного на множестве* $\bar{\Pi}$, *выполняется условие, что для любых перестановок из* N *требований* $\pi^{(ab)} = \left(\pi^{(1)}, \pi^{(a)}, \pi^{(b)}, \pi^{(2)}\right)$ *и* $\pi^{(ba)} = \left(\pi^{(1)}, \pi^{(b)}, \pi^{(a)}, \pi^{(2)}\right)$, *отличающихся только порядком следования последовательностей элементов* $\pi^{(a)}, \pi^{(b)}$, *в т. ч. и таких, что*

$\pi^{(1)} = \pi_0$ или $\pi^{(2)} = \pi_0$, из условия $\omega\left(\pi^{(a)}\right) > \omega\left(\pi^{(b)}\right)$ следует $\Phi\left(\pi^{(ab)}\right) \leq \Phi\left(\pi^{(ba)}\right)$, а из $\omega\left(\pi^{(a)}\right) = \omega\left(\pi^{(b)}\right)$ следует $\Phi\left(\pi^{(ab)}\right) = \Phi\left(\pi^{(ba)}\right)$, то такой функционал будем называть приоритето-порождающим, а соответствующий функционал $\omega(\pi)$ — функционалом приоритета.

Если $\Phi(\pi)$ является приоритето-порождающим на некотором множестве, то он также будет являться приоритето-порождающим на всех его подмножествах [20].

При решении задач исследования операций вещественное значение функционала $\Phi(\pi)$ может интерпретироваться как величина некоего целевого критерия, а $\omega(\pi)$ как некое правило (или алгоритм) упорядочения требований, направленное на минимизацию целевого функционала.

Далее будем решать задачу построения ППФ для одно и многолинейных СА.

Постановка задачи

Пусть дано множество задач $T = \{T_{ji}\}$, $j = 1, \ldots, n$, $i \in I_r \subseteq \{1, \ldots, S\}$, а Ind множество всех индексов $\{ji_r\}$ для множества T. Известно, что T — конечное счётное множество мощности $|T| \leq n \times S$.

- Каждая работа j состоит из задач: $T_{js} : \{T_{j1}, T_{j2}, \ldots, T_{jS}\}$, по одной для каждого ресурса (возможны пустые задачи)
- Для каждой задачи T_{js} работы J_j, использующей ресурс R_s, определено время выполнения $p_{js} \in \mathbb{R}^+$ (может быть нулевым для пустых задач)
- T — множество всех задач (без учёта работы и ресурса) с соответствующими временами выполнения p
- C_{js} — время выполнения задачи T_{js}
- C_j — время выполнения работы J_j
- C_π — время выполнения расписания π

Кроме того, будем считать верными следующие положения:

- Ресурс может одним только одним прибором одновременно;
- Вытеснение/прерывания запрещены;
- Процессоры не простаивают без необходимости при выполнении расписания;
- Все времена прибытия считаются равными 0.

Рассмотрим построение приоритето-порождающего функционала для одного прибора и предложим стратегию применения разработанных функционалов для случая m приборов.

Построение приоритето-порождающего функционала для случая обслуживания одним прибором

В случае обслуживания одним прибором автоматически оказывается выполнено условие одновременного обслуживания каждой задачи не более, чем одним прибором. Также будем считать выполненным условие, что один прибор обслуживает одновременно не более, чем одну задачу. Таким образом, $L = m = 1$ для модели (2.7).

Пусть

$$\Phi(\pi) = \sum_{j=1}^{n} \max_{i_r=1}^{r_{\max}} C_{ji_r}, \forall \pi = (i_1, i_2, \ldots, i_k), \tag{2.12}$$
$$\Phi(\pi_0) = \emptyset.$$

Покажем, что функционал $\Phi(\pi)$, заданный таким образом, является приоритето-порождающим, и найдем соответствующий ему функционал приоритета $\omega(\pi)$.

Рассмотрим оптимизацию по совокупному времени выполнения задачи $\sum_{i \in T} C_i$ для случая единственного прибора.

$$F(\pi) = \sum_{i \in \pi} C_i. \tag{2.13}$$

Очевидно, что:

$$F(\pi', \pi'') \equiv F(\pi') + (C_{\pi'} \cdot |\pi''|) + F(\pi''). \tag{2.14}$$

Используя тождество (2.14), получаем:

$$F(\pi^1, \pi^a, \pi^b, \pi^2) = F(\pi^1) + (C_{\pi^1} \cdot (|\pi^a| + |\pi^b| + |\pi^2|)) + F(\pi^a, \pi^b, \pi^2), \tag{2.15}$$

$$F(\pi^1, \pi^b, \pi^a, \pi^2) = F(\pi^1) + (C_{\pi^1} \cdot (|\pi^b| + |\pi^a| + |\pi^2|)) + F(\pi^b, \pi^a, \pi^2). \tag{2.16}$$

Из (2.15) и (2.16) следует, что:

$$F(\pi^1, \pi^a, \pi^b, \pi^2) - F(\pi^1, \pi^b, \pi^a, \pi^2) \equiv F(\pi^a, \pi^b, \pi^2) - F(\pi^b, \pi^a, \pi^2).$$

Применяя тождество (2.14) рекурсивно дважды, получаем:

$$F(\pi^a, \pi^b, \pi^2) = F(\pi^a) + C_{\pi^a} \cdot (|\pi^b| + |\pi^2|) + \left(F(\pi^b) + (C_{\pi^b} \cdot |\pi^2|) + F(\pi^2)\right),$$

$$F(\pi^b, \pi^a, \pi^2) = F(\pi^b) + C_{\pi^b} \cdot (|\pi^a| + |\pi^2|) + \left(F(\pi^a) + (C_{\pi^a} \cdot |\pi^2|) + F(\pi^2)\right).$$

Таким образом:

$$F(\pi^a, \pi^b, \pi^2) - F(\pi^b, \pi^a, \pi^2) = C_{\pi^a} \cdot |\pi^b| - C_{\pi^b} \cdot |\pi^a|.$$

Так как $|\pi^a| \geq 0$ и $|\pi^b| \geq 0$, следовательно:

$$F(\pi^1, \pi^a, \pi^b, \pi^2) \leq F(\pi^1, \pi^b, \pi^a, \pi^2) \Leftrightarrow \frac{C_{\pi^a}}{|\pi^a|} \leq \frac{C_{\pi^b}}{|\pi^b|}.$$

Заключаем, что F — приоритето-порождающий функционал с функцией приоритета:

$$\omega(\pi) = -\frac{C_\pi}{|\pi|} = -\frac{1}{|\pi|} \sum_{i \in \pi} p_i.$$

Рассмотрим оптимизацию по максимальному времени завершения $\max\limits_i C_i$. Этот случай тривиален, $C_{\max} = \max\limits_i C_i$ — это просто время завершения последней обработанной задачи, и оно постоянно, независимо от последовательности обработки π:

$$C_{\max} = \sum_{i \in T} p_i.$$

Рассмотрим оптимизацию по совокупному времени завершения работ $\sum_j C_j$. Этот случай можно упростить до 2.13 со следующими соображениями:

– Все задачи T_{js}, относящиеся к конкретной работе, должны обрабатываться последовательно, порядок обработки внутри такой работы очевидно не имеет значения, так как не влияет на C_j;
– Мы рассматриваем все задачи T_{js} работы J_j как одну задачу. Поскольку они обрабатываются последовательно, время выполнения работы не изменяется. Таким образом, производится преобразование случая в 2.13.

Чтобы обосновать это, достаточно доказать, что любое расписание π, в котором T_{js} не являются последовательными, гарантированно будет менее оптимальным, чем соответствующее расписание с последовательными T_{js}. Доказательство:

– Время завершения работы — это время завершения последней задачи, относящейся к этой работе в расписании (очевидно).
– Для конкретного расписания π последовательность последних задач уникальна (очевидно).
– Определим сюръективную (но не биективную) функцию перестановки SEQ : $\pi \to \pi_{\text{seq}}$, которая отображает любое конкретное расписание без ограничений $\hat{\pi}$ в другое расписание $\hat{\pi}_{\text{seq}}$, в котором последовательность последних задач остается неизменной, но все задачи для конкретной работы являются последовательными.

– Докажем, что такое преобразование всегда определено (тривиально).
– Докажем, что оптимизационная функция всегда улучшается, то есть $F(\hat{\pi}) \geq F(\hat{\pi}_{\text{seq}})$.

Подходящая сюръективная функция перестановки SEQ вводится таким образом:

1. Отметим n последних задач $T_{j\text{last}}$, соответствующих каждой работе J_j, это определяет порядок завершения работ.
2. Начиная с последней работы, для каждой работы: выполнить итерацию по расписанию, перемещая каждую задачу T_{ji} на один слот раньше соответствующей $T_{j\text{last}}$.

Доказательство, что $F(\hat{\pi}_{\text{seq}}) \leq F(\hat{\pi})$, с $\hat{\pi}_{\text{seq}} = \text{SEQ}(\hat{\pi})$:

– Время завершения C_j работы J_j — это время завершения соответствующей последней задачи, принадлежащей этой работе $T_{j\text{last}}$ в расписании (очевидно).
– Если задача $T_{j\text{last}}$ занимает в расписании слот $i_{j\text{last}}$, то

$$C_j = \sum_{i \leq i_j} p_i. \tag{2.17}$$

– Обратим внимание, что каждая перестановка задач в SEQ всегда состоит из перемещения задачи слева направо (то есть из прошлого в будущее).
– Согласно уравнению (2.17), каждая перестановка задач улучшит время выполнения всех предыдущих работ, не затрагивая при этом время завершения задачи соответствующей работы (а также время выполнения последующих работ).
– Поскольку SEQ состоит только из таких перестановок задач, применение SEQ улучшает только C_j, поэтому $F(\hat{\pi}_{\text{seq}}) \leq F(\hat{\pi})$.

Построение приоритето-порождающего функционала для случая обслуживания m выделенных приборами

Рассмотрим оптимизацию по совокупному времени завершения задач $\sum_{i \in T} C_i$ для случая S выделенных приборов. Количество ресурсов полагаем равным количеству приборов, $S = m$. Исходя из приведённых допущений, следует, что расписание для каждого прибора/ресурса можно оптимизировать отдельно. Каждая проблема математически эквивалентна случаю с одним прибором, если учитывать только подмножество задач, использующих этот ресурс.

Рассмотрим оптимизацию по максимальному времени завершения (или полному времени) $\max_i C_i$. Этот случай также тривиален. Мы рассматриваем последовательность обработки для каждого прибора/ресурса как отдельные случаи, а затем просто берем максимальное время обработки, которое является ресурсом,

требующим наибольшего совокупного времени обработки.

$$C_{\max} = \max_{s=1,\ldots,S}\{\max_{j=1,\ldots,n} C_{j1}, \max_{j=1,\ldots,n} C_{j2}, \ldots, \max_{j=1,\ldots,n} C_{jS}\}.$$

При этом аналитическое построение ППФ для таких целеых фукнций как совокупное время завершения работ и совокупное время завершения агрегированных работ нетривиально и не выполнялось в рамках данного исследования.

Построение приоритето-порождающего функционала для случая обслуживания индивидуальными независимыми приборами, $1 < m < S$

Нетривиальным является аналитическое постоение ППФ первого порядка для $\sum_j C_j$ и C_a и в случае обслуживания индивидуальынми независимыми приборами, при $m \in [1, S]$.

Случай оптимизации по максимальному времени завершения $C_{\max}$ сводится к задаче об упаковке в контейнеры с дополнительными ограничениями, которая является NP-трудной [138].

Один из возможных подходов к приближенному решению задач составления расписаний может состоять в понимании множества требований, получаемого в задачах составления расписаний для СА после декомпозиции работ на задачи, как неупорядоченного и состоящего из независимых элементов. В таком случае для получения решения задачи минимизации приоритето-порождающего функционала достаточно [53] упорядочить требования по невозрастанию соответствующих им значений функционалов приоритета (в этом случае функционал $\omega(js)$ называется функционалом 1-приоритета для задачи T_{js}).

Так, в [139—143] и др. были получены функционалы 1-приоритета, соответствующие известным в теории расписаний правилам диспетчеризации, таким как SPT (shortest processing time) — функционал 1-приоритета $\omega(js) = \frac{1}{p_{js}}$, LPT (longest processing time) — функционал $\omega(js) = p_{js}$ и др. Близкие подходы были также предложены в работах [144; 145].

Таким образом, возможным оказывается наряду с исследованием возможностей построения эвристических алгоритмов для СА, выполнять поиск алгоритмов составления расписаний в СА, основанных на приоритето-порождающих функционалах 1-го порядка.

2.4 Выводы и результаты по главе 2

В данной главе

1. Разработан комплекс математических моделей составления расписаний для нескольких приборов для случаев дискретных и дискретно-непрерывных ограничений качественного доступа к удалённым информационных ресурсам;
2. Рассмотрены различные постановки оптимизационных задач, соответствующих предложенным моделям;
3. Выполнено математическое исследование существования аналитического решения для случаев дискретных ограничений на ресурсы и различных видов функции расхода ресурса для модели с непрерывными ограничениями на ресурсы;
4. Выполнено исследование по построению приоритето-порождающих функционалов 1-го порядка для частных формулировок задач поиска оптимальных расписаний в СА;
5. Результаты исследования сформулированы в виде ряда теорем (ссылки) и следствия, промежуточные этапы доказательств сформилурованы в леммах 2.1–2.2;
6. Полученные результаты позволили обосновать необходимость разработки численных алгоритмов решения.

Глава 3. Алгоритмы построения расписаний обслуживания задач в многолинейной системе с ограниченными восполнимыми ресурсами

В главе 2 нами было показано, что задача построения расписаний для сформулированных моделей является NP-сложной в строгом смысле, что делает использование точных перечислительных алгоритмов для составления расписаний неприемлемым для практических целей. В связи в этим, в данной главе исследуем эвристические алгоритмы решения таких задач, основанные на методе перестановок (необходимые выкладки были выполнены нами в ч. 2.3.

При этом, будем рассматривать два следующих случая:

– Число приборов $m = S$. В данном случае оказывается возможным осуществить привязку задач, требующих определённого s-того ресурса, к конкретному $l = s$-тому прибору, что существенно упрощает задачу составления расписания. Такие приборы будем называть спецаилизированными, либо выделенными (dedicated);
– Число приборов $m \leq S$, при этом назначание соответствия между отдельными приборами и ресурсами невозможно. Для этого случая будем считать приборы индивидумальными, обладающими одинакотовой скоростью обработки задач и независимыми. Данная задача является обобщенной и более сложной модицикацией предыдущей.

Приведём обоснование изучения таких случаев с точки зрения практического применения. Рассмотрим типичные сценарии распределённой системы обработки агрегированных запросов для области хранения и обработки данных, когда ресурсы — базы данных, предоставляющие информацию, а приборы — это процессы, запущенные в операционной системе на сервере/серверах с соответствующим программным обеспечением для обработки данной информации.

«Узкими местами» такой системы могут являться приборы (приборы) либо ресурсы. Проанализируем данные ситуации:

1. Пусть узким местом системы являются ресурсы: т.е., имеют место базы данных с относительно медленным доступом. При этом как скорость передачи, так и скорость обработки данных велика по сравнению со скоростью доступа к ресурсам. В этом случае мы без ущерба для производительности можем параллельно запустить требуемое количество процессов, обрабатывающих данные, не менее одного на каждое хранилище данных — т.е. $m \geq S$. Согласно

следствию 2.1, случай $m > S$ является неоптимальным с точки зрения составления эффективных расписаний, таким образом, для данного случая разумно полагать $m = S$. Данное предположение помогает снизить трудность задачи и разработать алгоритмы с линейно-логарифмической временной сложности доя её решения;

2. Пусть узкое место — приборы: т.е. обработка получаемых данных требует больших вычислительных ресурсов, а скорость доступа к данным велика по сравнению со скоростью обработки. В этом случае также потенциально возможно параллельно запустить любое количество процессов для обработки информации, но это не будет приводить к ускорению работы из-за ограниченности вычислительных ресурсов. Более того, согласно лемме 2.2, это будет приводить к ухудшению таких критериев оптимальности, как суммарное время обработки задач и суммарное полное время обработки (включающее как время обработки, так и время ожидания задач в очереди). Таким образом, для этого варианта выгодно ограничить количество приборов, так что $m < S$. Возможны следующие случаи данного варианта:
   a) Программное обеспечение, необходимое для обработки данных из каждого источника, доступно на любом из серверов. В этом случае, хотя $m < S$, отсутствует привязка ресурсов к приборам, и любая задача может выполняться на любом приборе.
   b) Программное обеспечение, необходимое для обработки данных из каждого источника, доступно не на всех серверах, например из-за аппаратных или лицензионных ограничений. В этом случае, помимо того, что $m < S$, присутствует привязка ресурсов к приборам, и задачи использующие определённые ресурсы, могут выполняться только на соответствующих серверах и запущенных на них приборах.

Таким образом, данная глава будет разделена на две части: алгоритмы решения задачи (2.7) для выделенных приборов ($m = S$) и алгоритмы решения задачи (2.7) для произвольных индивидуальных независимых приборов ($m < S$).

3.1 Построение расписаний в случае выделенных приборов ($m = S$)

3.1.1 Алгоритмы, основанные на функционалах 1-приоритета

Рассмотрим семейство алгоритмов для случая выделенных приборов, напрямую использующих функционалы 1-приоритета для сортировки задач. Далее для краткости будем называть их алгоритмами, основанными на функционалах 1-

приоритета, или алгоритмами, основанными на приоритето-порождающих функционалах, для выделенных приборов (АППФ1-ВП).

Шаг 0. Если ресурсы не требуют обработки на выделенных приборах и число приборов m больше или равно числа ресурсов S ($m \geq S$), то для каждого ресурса R_s выделить собственный прибор P_{l_s} случайным образом без повторений.

Шаг 1. Выполнить разбиение работ J_j на независимые задачи T_{js}, использующие различные ресурсы.

Шаг 2. Составить список задач Q_T.

Шаг 3. Исключить из него все фиктивные задачи со временем обработки 0.

Шаг 4. Присвоить каждой задаче приоритет согласно используемому приоритето-порождающему функционалу.

Шаг 5. Отсортировать задачи согласно приоритету; в итоге получим отсортированный список задач Q_T.

Шаг 6. Взять задачу T_{js} из списка Q_T. Если список пуст — **завершить** составление расписания.

Шаг 7. Определить ресурс R_s, используемый задачей.

Шаг 8. Определить прибор P_{l_s}, выделенный для данного ресурса.

Шаг 9. Поставить задачу T_{js} в очередь найденного прибора P_{l_s}. Перейти к шагу 6.

Анализ наихудшего случая

Пусть дано $m = S$ приборов P_l, $l = 1, \ldots, m$; S ресурсов R_s, $s = 1, \ldots, S$; и n работ J_j, $j = 1, \ldots, n$, а k — допустимое число задач в каждой работе, $k = 1, \ldots, S$. Худшим случаем для выполнения алгоритма будет ситуация $k = S = m$. Проанализируем количество элементарных операций, выполняемых алгоритмом в данном случае.

***Теорема* 3.1.** *Предел худшего времени исполнения алгоритмов, основанных на приоритето-порождающих функционалах, в случае выделенных приборов (АППФ1-ВП) равен* $T(n, S) = O(nS \log nS)$.

Доказательство. Шаг 0. На данном шаге вначале производятся операции составления списка приборов и ресурсов ($m + S = 2S$ элементарных операций). Для выделения собственного прибора для ресурса удобно использовать операцию перемешивания списка (вычислительная сложность $O(S \log S)$ для наилучших алгоритмов) с последующим попарным сопоставлением (вычислительная сложность S. Таким образом, вычислительная сложность этого шага составляет $O(S \log S)$.

Шаг 1. На данном шаге производятся операции чтения массива работ J_j (n элементарных операций) и инициализации задач, количество которых мы оцениваем как $n \times S$. Итого, вычислительная сложность этого шага составляет $O(nS)$.

Для шагов 2–4 операции производятся для каждой задачи независимо от других, таким образом, вычислительные затраты линейно зависят от числа задач. Вычислительная сложность $O(nS)$.

Шаг 5. Вычислительная сложность данного шага соответствует сложности наиболее эффективных алгоритмов сортировки, $O(N \log N)$, где N — число элементов в списке, $N = nS$. Таким образом, вычислительная сложность этого шага составляет $O(nS \log nS)$.

Проход по шагам 6-9 повторяется, пока список задач не пуст, т.е. $n \times S$ раз для худшего случая. При этом вычислительные затраты на каждом шаге не зависят от числа задач, приборов или ресурсов. Таким образом, до окончания работы цикла будет выполнено число элементарных операций, пропорциональное $n \times S$. Вычислительная сложность $O(nS)$.

Очевидно, что наиболее вычислительно сложным шагом алгоритма является Шаг 5. Таким образом, вычислительная сложность алгоритма для худшего случая составляет $O(nS \log nS)$. □

Сводная информация по анализу худшего случая приведена в таблице 5

Таблица 5 — Асимптотический анализ худшего времени выполнения для алгоритмов, основанных на приоритето-порождающих функционалах, в случае выделенных приборов (АППФ1-ВП)

Шаг	**Элементарных операций**	**Вычислительная сложность**
0	• Составление списков приборов и ресурсов, $2S$; • Перемешивание списка, $O(S \log S)$; • Попарное сопоставление, S	$O(S \log S)$
1	nS	$O(nS)$
2	nS	$O(nS)$
3	nS	$O(nS)$
4	nS	$O(nS)$

Продолжение на следующей странице

Шаг		Элементарных операций	Вычислительная сложность
5		Сортировка массива из nS элементов, $O(nS \log nS)$	$O(nS \log nS)$
Цикл $n \times S$	6	1	$O(1)$
	7	1	$O(1)$
	8	1	$O(1)$
	9	1	$O(1)$
	Всего по циклу	$4nS$	$O(nS)$
Итого			$O(nS \log nS)$

3.1.2 Эвристический алгоритм диспетчеризации задач в СА в случае выделенных приборов

Предложим следующий алгоритм распределения задач по приборам:

Шаг 0. Выполнить разбиение работ J_j на независимые задачи T_{js}, использующие различные ресурсы.

Шаг 1. Для каждой работы J_j инициализировать наихудший приоритет w_j значением 0: $w_j = 0$.

Шаг 2. Для каждого ресурса R_s составить список Q_s обращающихся к нему задач.

Шаг 3. Рассчитать время выполнения для каждой задачи по формуле $p_{js} = \mu_s V_{js}$.

Шаг 4. Для каждого ресурса рассчитать суммарное время выполнения обращающихся к ним задач $p_s = \sum_j p_{js}$.

Шаг 5. Отсортировать ресурсы по убыванию p_s, получив последовательность Q_R.

Шаг 6. Взять очередной ресурс R_s из последовательности Q_R; `если` последовательность пуста, `то` **завершить** составление расписания.

Шаг 7. Присвоить времени ожидания в очереди W значение 0: $W = 0$, присвоить оптимальному приоритету $w_{\min}$ максимально возможное значение `MAX_FLOAT`.

Шаг 8. Взять очередную задачу T_{js} из списка Q_s; `если` расчёт выполнен для всех задач, `то перейти` к шагу 13.

Шаг 9. Для текущей работы J_j обновить наихудший приоритет $w_j = \max(W + p_{js}, w_j)$.

Шаг 10. Присвоить приоритет рассматриваемой задаче T_{js} в соответствии с текущим значением w_j: $w_{js} = w_j$.

Шаг 11. Присвоить оптимальному приоритету $w_{\min}$ значение $\min(w_{\min}, w_{js})$.

Шаг 12. Присвоить времени ожидания W значение $w_{\min}$. `Перейти` к шагу 8.

Шаг 13. Отсортировать задачи списка Q_s по возрастанию w_{js}. `Если` у двух задач одинаковое значение w, `то` взять задачу с меньшим значением $\sum_{k=s+1}^{S} p_{jk}$.

Шаг 14. Поочерёдно поставить задачи списка Q_s в очередь прибора P_s. `Перейти` к шагу 6.

Назовём его алгоритмом двухуровневой диспетчеризации для случая выделенных приборов (АДД-ВП).

Анализ наихудшего случая

Пусть дано $m = S$ приборов P_l, $l = 1, \ldots, m$; S ресурсов R_s, $s = 1, \ldots, S$; и n работ J_j, $j = 1, \ldots, n$, а k — допустимое число задач в каждой работе, $k = 1, \ldots, S$. Худшим случаем для выполнения алгоритма будет ситуация $k = S = m$. Проанализируем количество элементарных операций, выполняемых алгоритмом в данном случае.

***Теорема* 3.2.** *Предел худшего времени исполнения алгоритма двухуровневой диспетчеризации для случая выделенных приборов (АДД-ВП) равен* $T(n, S) = O(nS \log nS)$.

Доказательство. Шаг 0. На данном шаге производятся операции чтения массива работ J_j (n элементарных операций) и инициализации задач, количество которых мы оцениваем как $n \times S$. Итого, вычислительная сложность этого шага составляет $O(nS)$.

Шаг 1. Содержит n элементарных действий, по одному на каждую работу; вычислительная сложность — $O(n)$.

Шаг 2. Выполняется в два этапа: инициализация пустых списков Q_s (сложность $O(S)$), и проход по задачам с добавлением каждой в соответствующий список (сложность $O(nS)$).

Шаг 3. Производится поэлементное умножение каждой из n строк матрицы V на вектор μ_s, S элементарных операций на каждую из n строк. Вычислительная сложность $O(n \times S)$.

Шаг 4. Выполняется суммирование величин p_{js} по n для $s = 1, \ldots, S$. Минимальная вычислительная сложность суммирования в строке может достигать $O(\log n)$ (при суммировании сдваиванием [??]). Общая вычислительная сложность этого шага составляет $O(S \log n)$.

Шаг 5. Вычислительная сложность данного шага соответствует сложности наиболее эффективных алгоритмов сортировки, $O(S \log S)$.

Проход по шагам 6–14 повторяется S раз. При этом:

– Шаг 7 содержит два элементарных действия, сложность — $O(1)$.
– Цикл из шагов 8–12 для задач $T_j s$, обращающихся к ресурсу R_s, выполняется максимум за n проходов. Он состоит из четырёх шагов, на каждом из которых выполняется одно элементарное действие. Таким образом, его сложность — $O(n)$.
– Вычислительная сложность шага 13 соответствует сложности наиболее эффективных алгоритмов сортировки для массива максимум n элементов, $O(n \log n)$.
– Сложность шага 14 очевидно равна $O(n)$.

Итого, сложность каждого шага цикла 6–14 составляет $O(n \log n)$, а сложность всего цикла — $O(nS \log n)$.

Таким образом, АДД-ВП содержит два потенциально вычислительно сложных этапа: шаг 5 сложностью $O(S \log S)$ и цикл 6–14 сложностью $O(nS \log n)$. Итого, вычислительная сложность алгоритма для худшего случая составляет $O(nS \log nS)$. □

Сводная информация по анализу худшего случая приведена в таблице 6.

Таблица 6 — Асимптотический анализ худшего времени выполнения для АДД-ВП

Шаг			**Элементарных операций**	**Вычислительная сложность**
0			nS	$O(nS)$
1			n	$O(n)$
2			• Инициализация пустых списков, const $\times S$; • Добавление задач в списки, nS	$O(nS)$
3			nS	$O(nS)$
4			S суммирований n величин	$O(S \log n)$
5			Сортировка массива из S элементов	$O(S \log S)$
Цикл $n \times S$	6		1	$O(1)$
	7		2	$O(1)$
	Цикл	8	1	$O(1)$

Продолжение на следующей странице

Шаг			Элементарных операций	Вычислительная сложность
	n	9	1	$O(1)$
		10	1	$O(1)$
		11	1	$O(1)$
		12	1	$O(1)$
		Всего по циклу	$5n$	$O(n)$
	13		Сортировка массива из n элементов	$O(n \log n)$
	14		Однократный проход по списку n элементов	$O(n)$
	Всего по циклу		Определяется сложностью шага 13	$O(n \log n)$
Итого				$O(nS \log nS)$

3.2 Построение расписаний в случае индивидуальных независимых приборов ($m < S$)

3.2.1 Алгоритмы, основанные на приоритето-порождающих функционалах

Рассмотрим семейство алгоритмов для случая индивидуальных независимых приборов, количество которых меньше числа ресурсов, напрямую использующих функционалы 1-приоритета для сортировки задач. Далее для краткости будем называть их алгоритмами, основанными на функционалах 1-приоритета, или алгоритмами, основанными на приоритето-порождающих функционалах, для независимых приборов (АППФ1-НП).

Шаг 0. Для каждого ресурса R_s определить пустой список задач Q_s для задач с назначенным временем начала и окончания выполнения.

Шаг 1. Выполнить разбиение работ J_j на независимые задачи T_{js}, использующие различные ресурсы. Для каждой задачи оценить время выполнения.

Шаг 2. Составить список задач Q_T.

Шаг 3. Исключить из него все фиктивные задачи со временем обработки 0.

Шаг 4. Присвоить каждой задаче приоритет согласно используемому приоритето-порождающему функционалу.

Шаг 5. Отсортировать задачи согласно приоритету; в итоге получим отсортированный список задач Q_T.

Шаг 6. Взять задачу T_{js} из списка Q_T. `Если` список пуст, `то` **завершить** составление расписания.

Шаг 7. Выбрать случайный прибор P_l.

Шаг 8. Поставить задачу в очередь выбранного прибора P_l. Определить время начала её выполнения t_{js}^{start} как время окончания выполнения предыдущей задачи в очереди. Определить время окончания её выполнения как сумму времени начала и времени обработки задачи: $t_{js}^{\text{finish}} = t_{js}^{\text{start}} + p_{js}$.

Шаг 9. Рассмотреть очередную задачу $T'_{j's}$ из списка Q_s (начиная с первой). Обозначим время её начала и завершения как $t'^{\text{start}}_{j's}$ и $t'^{\text{finish}}_{j's}$. При достижении конца списка Q_s — `перейти` к шагу 12.

Шаг 10. `Если` $t_{js}^{\text{finish}} \le t'^{\text{start}}_{j's}$, `то перейти` к шагу 12. Таким образом, мы находим ближайший временной слот длительностью не менее p_{js}, когда ресурс R_s свободен.

Шаг 11. Изменить время начала выполнения задачи $t_{js}^{\text{start}} = t'^{\text{finish}}_{j's}$. Изменить время окончания выполнения задачи $t_{js}^{\text{finish}} = t_{js}^{\text{start}} + p_{js}$. `Перейти` к шагу 9.

Шаг 12. Зафиксировать для задачи T_{js} время её начала и завершения t_{js}^{start} и t_{js}^{finish}. Добавить задачу T_{js} в список задач Q_s ресурса R_s. `Перейти` к шагу 6.

Анализ наихудшего случая

Пусть дано $m < S$ приборов P_l, $l = 1, \ldots, m$; S ресурсов R_s, $s = 1, \ldots, S$; и n работ J_j, $j = 1, \ldots, n$, а k — допустимое число задач в каждой работе, $k = 1, \ldots, S$. Худшим случаем для выполнения алгоритма будет ситуация $k = S$. Проанализируем количество элементарных операций, выполняемых алгоритмом в данном случае.

***Теорема* 3.3.** *Предел худшего времени исполнения алгоритмов, основанных на приоритето-порождающих функционалах, в случае индивидуальных независимых приборов и $m < S$ (АППФ1-НП) равен $T(n, S) = O(n^2 S \log S)$.*

Доказательство. Шаг 0. На данном шаге вначале производятся операции составления списка приборов и ресурсов ($m + S$ элементарных операций). Для выделения собственного прибора для ресурса удобно использовать операцию перемешивания списка (вычислительная сложность $O(m \log m)$ для наилучших алгоритмов) с последующим попарным сопоставлением (вычислительная сложность m. Таким образом, вычислительная сложность этого шага составляет $O(m \log m) < O(S \log S)$.

Шаг 1. На данном шаге производятся операции чтения массива работ J_j (n элементарных операций) и инициализации задач, количество которых мы оцениваем как $n \times S$. Итого, вычислительная сложность этого шага составляет $O(nS)$.

Для шагов 2–4 операции производятся для каждой задачи независимо от других, таким образом, вычислительные затраты линейно зависят от числа задач. Вычислительная сложность $O(nS)$.

Шаг 5. Вычислительная сложность данного шага соответствует сложности наиболее эффективных алгоритмов сортировки, $O(N \log N)$, где N — число элементов в списке, $N = nS$. Таким образом, вычислительная сложность этого шага составляет $O(nS \log nS)$.

Проход по шагам 6–12 повторяется, пока список задач не пуст, т.е. $n \times S$ раз для худшего случая. При этом:

- На Шаге 6 выполняется 2 элементарных операции — проверка списка на пустоту и, в случае непустоты, извлечение задачи из начала списка в переменную текущей задачи задачи.
- Далее на Шаге 7 выбирается случайный прибор — 1 составная операция, сложность которой не зависит от числа приборов.
- На Шаге 8 задача ставится в очередь выбранного прибора и определяется время начала и окончания её выполнения. Количество элементарных операций зависит от деталей реализации, но является постоянным.
- Число проходов по шагам 9–11 равно числу рассмотренных задач, использующих определённый ресурс. Очевидно, оно не может быть больше числа работ n. При этом:
  - Сложность Шага 9 постоянна и равна $O(1)$.
  - Сложность Шага 10 также постоянна. На этом шаге может произойти выход из цикла, но для анализа худшего случая следует рассматривать вариант, когда этого не происходит.
  - Сложность Шага 11 составляет 3–4 элементарных операции в зависимости от вычислительной архитектуры: присвоение, суммирование с присвоением и переход.
- Итого, сложность цикла 9–11 составляет $O(n)$.
- На шаге 12 выполняется фиксированное число элементарных операций, его сложность — $O(1)$.

Таким образом, вычислительная сложность цикла 6–12 составляет $O(n^2 \times S)$.

Получается, что наиболее вычислительно сложными элементами алгоритма являются цикл 6–12 и шаг 5, и его вычислительная сложность для худшего случая составляет $O(n^2 S \log S)$. □

Сводная информация по анализу худшего случая приведена в таблице 7

Таблица 7 — Асимптотический анализ худшего времени выполнения для алгоритмов, основанных на приоритето-порождающих функционалах, в случае индивидуальных независимых приборов и $m < S$ (АППФ1-НП)

Шаг			**Элементарных операций**	**Вычислительная сложность**
0			• Составление списков приборов и ресурсов, $m + S$; • Перемешивание списка, $O(m \log m)$; • Попарное сопоставление, m	$O(m \log m)$
1			nS	$O(nS)$
2			nS	$O(nS)$
3			nS	$O(nS)$
4			nS	$O(nS)$
5			Сортировка массива из nS элементов, $O(nS \log nS)$	$O(nS \log nS)$
Цикл $n \times S$	6		2	$O(1)$
	7		const	$O(1)$
	8		const	$O(1)$
	Цикл n	9	const	$O(1)$
		10	const	$O(1)$
		11	const	$O(1)$
		Всего по циклу	const $\times n$	$O(n)$
	12		const	$O(1)$
	Всего по циклу		const $\times n^2 S$	$O(n^2 S)$
Итого				$O(n^2 S \log S)$

3.2.2 Эвристический алгоритм диспетчеризации задач в многолинейной системе с ограниченными восполнимыми ресурсами для систем с агрегацией

Предложим следующий алгоритм распределения задач по индивидуальным независимым приборам:

Шаг 0. Выполнить разбиение работ J_j на независимые задачи T_{js}, использующие различные ресурсы.

Шаг 1. Рассчитать время выполнения для каждой задачи по формуле $p_{js} = \mu_s V_{js}$.

Шаг 2. Для каждой работы рассчитать суммарное время выполнения относящихся к ней задач $p_j = \sum_{s=1}^{S} p_{js}$.

Шаг 3. Отсортировать работы по возрастанию p_j и задачи в каждой работе по возрастанию p_{js}; в итоге получим отсортированный список задач Q_T. Исключить из списка Q_T все фиктивные задачи со временем обработки 0.

Шаг 4. Для каждого прибора объявить время завершения последней задачи $F_m = 0$.

Шаг 5. Взять задачу T_{is} из списка Q_T. `Если` список пуст, `то` **завершить** составление расписания.

Шаг 6. `Если` у одного из приборов для последней задачи $T_{j's}$, в его очереди выполняется $s = s$, `то` поставить задачу T_{js} в очередь этого прибора. Перейти к шагу 8.

Шаг 7. Поставить задачу T_{js} в очередь прибора с минимальным F_m. Если таких приборов несколько — поставить в очередь любого из них.

Шаг 8. Увеличить F_m на p_{js} : $F_m = F_m + p_{js}$. `Перейти` к шагу 5.

Назовём его алгоритмом двухуровневой диспетчеризации для случая индивидуальных независимых приборов (АДД-НП).

Приведём примеры использования алгоритма для двух случаев:

– $m = S$: Пример 3.1;
– $m < S$: Пример 3.2.

Пример 3.1

Рассмотрим в качестве примера систему, в которой имеется 3 обслуживающих прибора $\{P_l, P_2, P_3\}$ и 3 ресурса $\{R_l, R_2, R_3\}$, удовлетворяющие характеристикам модели. Пусть даны работы $\{J_1, J_2, J_3, J_4, J_5\}$, для которых известна следую-

щая матрица ресурсоёмкости:

$$V = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Коэффициенты μ_s, $s = 1,2,3$ будем считать равными 1, 2 и 5, соответственно.

Составим расписание обслуживания задач согласно АДД-НП.

Шаг 0. Выполним разбиение работ на независимые задачи T_{js}:

$$J_1 = \{T_{11}, T_{12}, T_{13}\},$$
$$J_2 = \{T_{21}, T_{22}\},$$
$$J_3 = \{T_{31}, T_{32}\},$$
$$J_4 = \{T_{41}, T_{43}\},$$
$$J_5 = \{T_{52}\}.$$

Шаг 1. Рассчитаем ожидаемое время выполнения для каждой задачи. Представим результаты расчетов в виде матрицы $\bar{P} = \{p_{ji}\}$:

$$\bar{P} = \begin{pmatrix} 1 & 2 & 5 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 5 \\ 0 & 2 & 0 \end{pmatrix}$$

Шаг 2. Сгруппируем задачи по работам и рассчитаем суммарное время их выполнения:

$$J_1 : \{T_{11}, T_{12}, T_{13}\}, \quad p_1 = 8,$$
$$J_2 : \{T_{21}, T_{22}\}, \quad p_2 = 3,$$
$$J_3 : \{T_{31}, T_{32}\}, \quad p_3 = 3,$$
$$J_4 : \{T_{41}, T_{43}\}, \quad p_4 = 6,$$
$$J_5 : \{T_{52}\}, \quad p_5 = 2.$$

Таким образом, последовательность Q_T на *шаге 3* будет иметь вид:

$$Q_T = \{J_5, J_2, J_3, J_4, J_l\} = \{T_{52}, T_{21}, T_{22}, T_{31}, T_{32}, T_{41}, T_{43}, T_{11}, T_{12}, T_{13}\}.$$

После *шага 4* $F_1 = F_2 = F_3 = 0$.

На *шаге 5* берём задачу T_{52} и ставим её в очередь любого из приборов, например прибора P_2. После этого $F_2 = 2$. Таким образом, задачу T_{21} мы ставим в очередь одного из приборов P_1 или P_3 - например, P_l. После этого $F_l = 1$. Задачу T_{22} мы должны поставить в очередь прибора P_2, т.к. до этого он обрабатывал задачу T_{52}, использующую тот же ресурс R_2. Его $F_2 = 4$. Продолжаем аналогично, пока не дойдём до задачи T_{43}– её необходимо поставить в очередь прибора P_3, т.к. на этот момент у него минимальная величина $F_3 = 0$. Все последующие задачи будут поставлены в очередь приборов с соответствующим номером.

Расписание, полученное по результатам расчета, показано на рисунке 3.1.

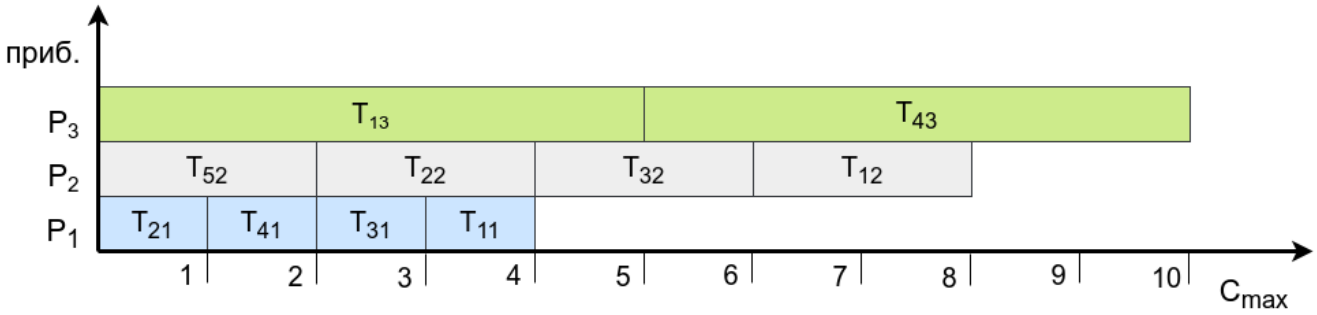


Рисунок 3.1 — Расписание обработки задач, составленное с использованием АДД-НП.

Рассчитаем значения критериев оптимальности для данного расписания:

$$C_{\max} = 10,$$

$$\sum_{j=1}^{n} \sum_{k=1}^{S} C_{jk} = 45,$$

$$C_a = 10 + 4 + 6 + 5 + 2 = 27.$$

Как можно видеть, в случае $m = S$ мы получили распределение, где задачи, потребляющие конкретный ресурс, оказались направлены для обслуживания на отдельном приборе. Можно показать, что в случае $m \geq S$ оптимальным будет направление обработки задач, требующих определённого ресурса, на один и тот же прибор (что из-за ограничения на одновременный доступ приборов к одному и тому же ресурсу может приводить к простоям в случае $m < S$). Рассмотрим различие между расписанием, составленным с использованием АДД-НП и расписанием, представленным на рисунке 3.2, составленным в предположении, что выполнение задач, задействующих тот или иной ресурс, производится на строго определённом приборе в порядке поступления задач в систему.

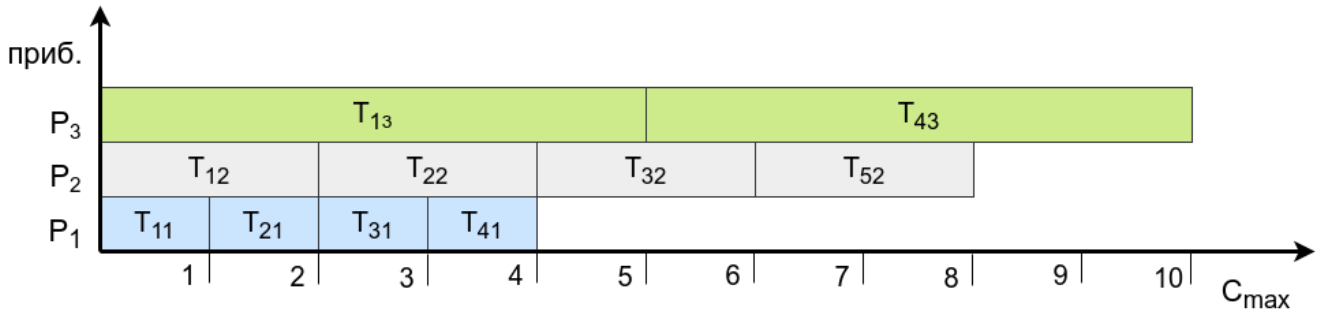


Рисунок 3.2 — Расписание обработки задач, составленное с привязкой ресурсов к определённому прибору с обслуживанием в порядке следования работ.

Выполним расчёт значений критериев оптимальности:

$$C_{\max} = 10,$$

$$\sum_{j=1}^{n} \sum_{k=1}^{S} C_{jk} = 45,$$

$$C_a = 8 + 6 + 9 + 14 + 8 = 45.$$

Таким образом, расписание, составленное АДД-НП, оказывается более оптимальным по критерию C_a, не ухудшая показатели по критериям $\sum_{j=1}^{n} \sum_{k=1}^{S} C_{jk}$ и $C_{\max}$.

Пример 3.2. Пусть дано 3 прибора P_1, P_2, P_3, 5 ресурсов R_1, R_2, R_3, R_4, R_5, и задачи $J_1, \ldots, J_{10}$. Ресурсы характеризуются следующим вектором скорости доступа $\mu = (1, 1.5, 2, 0.5, 1)$, работы — матрицей ресурсоёмкости

$$V = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 3 & 0 & 0 & 2 & 3 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 0 & 2 \end{pmatrix}$$

Составим расписание обслуживания задач согласно АДД-НП.

Ш0. Рассчитаем матрицу ожидаемых времен выполнения работ:

$$\bar{P} = \begin{pmatrix} 1 & 1.5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 3 & 0 & 0 & 1 & 3 \\ 0 & 1.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1.5 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1.5 & 0 \\ 1 & 1.5 & 2 & 0.5 & 1 \\ 0 & 0 & 4 & 0 & 2 \end{pmatrix}$$

и подсчитаем ожидаемое время выполнения работ (*Ш2*, таблица 8).

Таблица 8 — Ожидаемые времена выполнения работ $J_1, \ldots, J_{10}$

Величина	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
Значение	2,5	4	7	2	3	1	3,5	1,5	6	6

Ш2. Вычислим вектор $Q_T = \{J_6, J_8, J_4, J_1, J_5, J_7, J_2, J_9, J_{10}, J_3\} =$ $= \{T_{61}, T_{84}, T_{44}, T_{42}, T_{11}, T_{12}, T_{54}, T_{55}, T_{72}, T_{73}, T_{23}, T_{94}, T_{95}, T_{91}, T_{92}, T_{93}, T_{10,5}, T_{10,3}, T_{31}, T_{35}, T_{34}\}$.

Ш4. Инициализируем загрузку приборов: $F_1 = F_2 = F_3 = 0$.

Далее, следуя по шагам 5-8, получим следующий список назначений (таблица 9).

Таблица 9 — Назначение задач на приборы P_1, $P_2 P_3$ согласно АДД-НП

Итер	Назначение	Итер	Назначение	Итер	Назначение
1	$T_{61} \to P_1$	8	$T_{55} \to P_1$	15	$T_{92} \to P_1$
2	$T_{84} \to P_2$	9	$T_{72} \to P_3$	16	$T_{93} \to P_2$
3	$T_{44} \to P_2$	10	$T_{72} \to P_2$	17	$T_{10,5} \to P_3$
4	$T_{42} \to P_3$	11	$T_{23} \to P_2$	18	$T_{10,3} \to P_2$
5	$T_{11} \to P_1$	12	$T_{94} \to P_1$	19	$T_{31} \to P_1$
6	$T_{12} \to P_3$	13	$T_{95} \to P_1$	20	$T_{35} \to P_3$
7	$T_{54} \to P_2$	14	$T_{91} \to P_3$	21	$T_{34} \to P_1$

и следующие изменения загрузки приборов (таблица 10).

Таблица 10 — Изменения значений загрузки приборов по мере назначения задач на приборы

Назнач. задача	F_1	F_2	F_3
–	0	0	0
T_{61}	1	0	0
T_{84}	1	1,5	0
T_{44}	1	2	0
T_{42}	1	2	1,5
T_{11}	2	2	1,5
T_{12}	2	2	3
T_{54}	2	3	3
T_{55}	4	3	3
T_{72}	4	3	4,5
T_{73}	4	5	4,5
T_{23}	4	9	4,5
T_{94}	4,5	9	4,5
T_{95}	5,5	9	4,5
T_{91}	5,5	9	5,5
T_{92}	7	9	5,5
T_{93}	7	11	5,5
$T_{10,5}$	7	11	7,5
$T_{10,3}$	7	15	7,5
T_{31}	10	15	10,5
T_{35}	10	15	10,5
T_{34}	11	15	10,5

Таким образом, мы составили следующее расписание, показанное на рисунке 3.3.

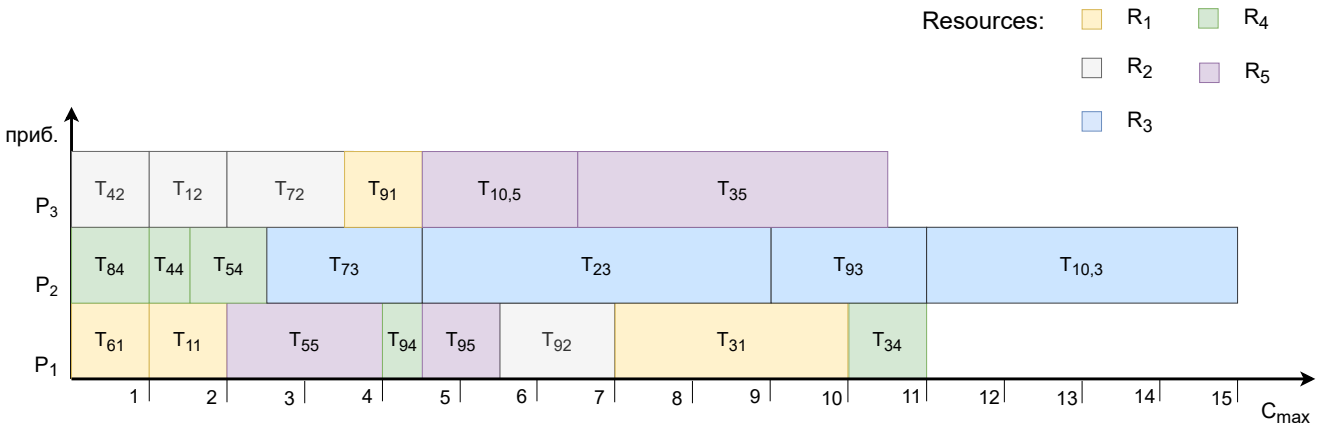


Рисунок 3.3 — Результирующее расписание для примера 3.2. Исполльзование ресурсов задачами обозначено следующим образом: R_1-желтый, R_2 - серый, R_3 - голубой, R_4 зелёный, R_5 - фиолетовый.

Для полученного расписания рассчитаем критерий C_a : $C_a = \max(2{,}2.5)+9+\max(10{,}11{,}11.5)+\max(1.5{,}2)+\max(3{,}4)+1+\max(4{,}5)+1.5+ +\max(4.5{,}5.5{,}7{,}11.5{,}5)+\max(7{,}15) = 2{,}5 + 9 + 10.5 + 2 + 4 + 1 + 5 + 1.5 + 11.5+ +15 = 62$

Значение $C_{max} = 15$.

Анализ наихудшего случая

Пусть дано $m \leq S$ приборов P_l, $l = 1, \ldots, m$; S ресурсов R_s, $s = 1, \ldots, S$; и n работ J_j, $j = 1, \ldots, n$, а k — допустимое число задач в каждой работе, $k = 1, \ldots, S$. Худшим случаем для выполнения алгоритма будет ситуация $k = S$. Проанализируем количество элементарных операций, выполняемых алгоритмом в данном случае.

***Теорема* 3.4.** *Предел худшего времени исполнения АДД-НП равен* $T(n, S) = O(nS \log nS)$.

Доказательство. Шаг 0. На данном шаге производятся операции чтения массива работ J_j (n элементарных операций) и инициализации задач, количество которых мы оцениваем как $n \times S$. Итого, вычислительная сложность этого шага составляет $O\,(n \times S)$.

Шаг 1. Производится поэлементное умножение каждой из n строк матрицы V на вектор μ_s, S элементарных операций на каждую из n строк. Вычислительная сложность $O\,(n \times S)$.

Шаг 2. Выполняется суммирование величин p_{js} по S для $j = 1, \ldots, n$. Минимальная вычислительная сложность суммирования в строке может достигать $O(\log S)$ (при суммировании сдваиванием [??]). Общая вычислительная сложность этого шага составляет $O(n \log S)$.

Шаг 3 содержит следующие ступени:

– отсортировать работы по значению величины p_j — сортировка массива из n элементов, сложность $O(n \log n)$ для случая таких сортировок как Merge sort или Quick sort.
– сортировать задачи в каждой работе: для каждой из n работ будет выполнена сортировка S элементов сложностью $O(S \log S)$. Общая сложность данного шага составит $O(nS \log nS)$.

Шаг 4. Выполняется m операций инициализации $F_l = 0, l = 1, \ldots, m$. Линейная вычислительная сложность $O(m)$, $m \leq S$.

Проход по шагам 5-8 повторяется, пока список задач не пуст, т.е. $n \times S$ раз для худшего случая. При этом:

– На Шаге 5 выполняется 2 элементарных операции — проверка списка на пустоту и, в случае непустоты, извлечение задачи из начала списка в переменную текущей задачи задачи.
– Далее на Шаге 6 выполняется проверка условия (1 операция) и либо постановка задачи в очередь прибора (1 операция) и переход к Шагу 8, либо переход к Шагу 7.
– На Шаге 7 выполняется одна элементарная операция.
– На Шаге 8 выполняются 2 элементарные операции сложения и присвоения значения.

Таким образом, до окончания работы цикла будет выполнено $6n \times S$ элементарных операций. Вычислительная сложность $O(n \times S)$.

Очевидно, что наиболее вычислительно сложным шагом алгоритма является Шаг 3. Таким образом, вычислительная сложность алгоритма для худшего случая составляет $O(nS \log nS)$. □

Сводная информация по анализу худшего случая приведена в таблице 11.

Таблица 11 — Асимптотический анализ худшего времени выполнения для АДД-НП

Шаг		**Элементарных операций**	**Вычислительная сложность**
0		nS	$O(nS)$
1		nS	$O(nS)$
2		n суммирований по S сложностью $O(\log S)$	$O(n \log S)$
3		• Сортировка массива из n элементов, $O(n \log n)$; • n сортировок подмножеств из S элементов, сложность каждой $O(S \log S)$	$O(nS \log nS)$
4		m операций инциализации, $m < S$	$O(m) < O(S)$
Цикл $n \times S$	5	2	$O(1)$
	6	1 или 2	$O(1)$
	7	1	$O(1)$
	8	2	$O(1)$
	Всего по циклу	$6nS$	$O(nS)$
Итого			$O(nS \log nS)$

3.3 Выводы и результаты по главе 3

В данном главе диссертационного исследования приведены алгоритмы численного решения задач составления расписаний в многоприборных системах с ресурсами, обладающими ограниченной качественной доступностью, основанные на приоритето-порождающих функционалах для случаев числа приборов равного числу ресурсов и числа приборов меньшего числа ресурсов.

Совокупность разработанных алгоритмов включает:

– Алгоритм численного решения задач, основанный на функционалах 1-приоритета, для случая выделенных приборов (АППФ1-ВП);
– Эвристический алгоритм двухуровневой диспетчеризации для случая выделенных приборов (АДД-ВП);
– Алгоритм численного решения задач, основанный на функционалах 1-приоритета, для случая независимых приборов, количество которых меньше числа ресурсов (АППФ1-НП);

– Эвристический алгоритм двухуровневой диспетчеризации для случая независимых приборов, количество которых меньше числа ресурсов (АДД-НП).

Таблица 12 — Асимптотический анализ времени выполнения для алгоритмов, разработанных в рамках диссертационного исследования

Алгоритмы		**Разновидности архитектуры СА**	
		Приборы	
		Выделенные	Независимые
		Соотношение приборов к ресурсам	
Принцип	Характеристики	$m = S$	$m < S$
ППФ 1-го порядка	Название	АППФ1-ВП	АППФ1-НП
	$T(n,S)$	$O(nS \log nS)$	$O(n^2 S \log S)$
Двухуровневой диспетчеризации	Название	АДД-ВП	АДД-НП
	$T(n, S)$	$O(nSlognS)$	$O(nSlognS)$

Доказан ряд теорем о асимптотическом пределе $T(n,S)$ худшего времени выполнения алгоритмов. Сводная информация по основным результатам главы приведена в таблице 12.

Глава 4. Численные эксперименты и комплексы программ

В главах 2 и 3 нами была задана математическая модель СА, определена задача поиска оптимального расписания для как задача целочисленного программирования, предложены алгоритмы для решения этой задачи и исследованы их асимптотические характеристики.

Известно [18], что для большинства детерминированных задач теории расписаний решения для задач дискретного целочисленного программирования могут быть без потери общности расширены для использования в задачах выпуклого программирования. В то же время, при таком расширении области допустимых значений параметров получение аккуратных аналитических оценок производительности алгоритмов становится трудновыполнимым.

Вычислительный эксперимент позволяет не только получить качественные оценки точности алгоритмов, но и сравнить их характеристики выполнения, а также исследовать поведение модели для различных комбинаций параметров модели, в т.ч. и для разных значений и распределений стохастических параметров модели.

В данной главе описаны постановка и проведение вычислительных экспериментов, представлены их результаты и их анализ. Было проведено два различных вычислительных экспериментов, и для каждого из них реализован комплекс программ. Описание каждого из комплексов программ следует за описанием численного эксперимента.

4.1 Применение приоритето-порождающих функционалов 1-го порядка для построения расписаний в многолинейной СА

4.1.1 Постановка численного эксперимента

На данном этапе численного эксперимента осуществляется преобразование математической модели в моделирующий алгоритм. Ключевым элементом данного алгоритма является блок генерации примеров. Приведём блок-схемы основной программы на рисунке 4.1 и блок-схему генератора примеров на рисунке 4.2.

4.1.2 Описание хода численного эксперимента

Целью данного численного эксперимента был поиск оптимального эвристического правила диспетчеризации (ПД) из класса приоритето-порождающих

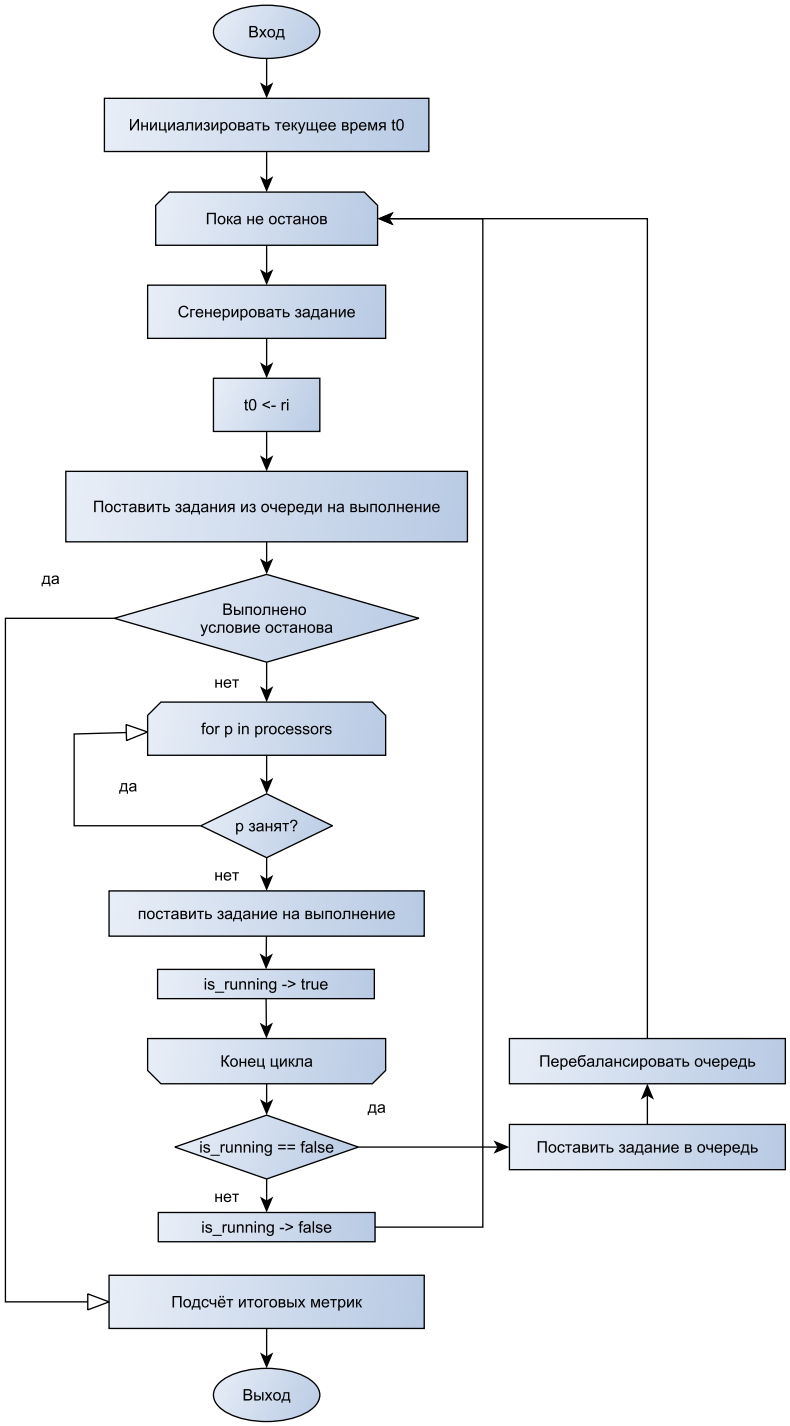


Рисунок 4.1 — Блок-схема основной программы для проведения вычислительного эксперимента.

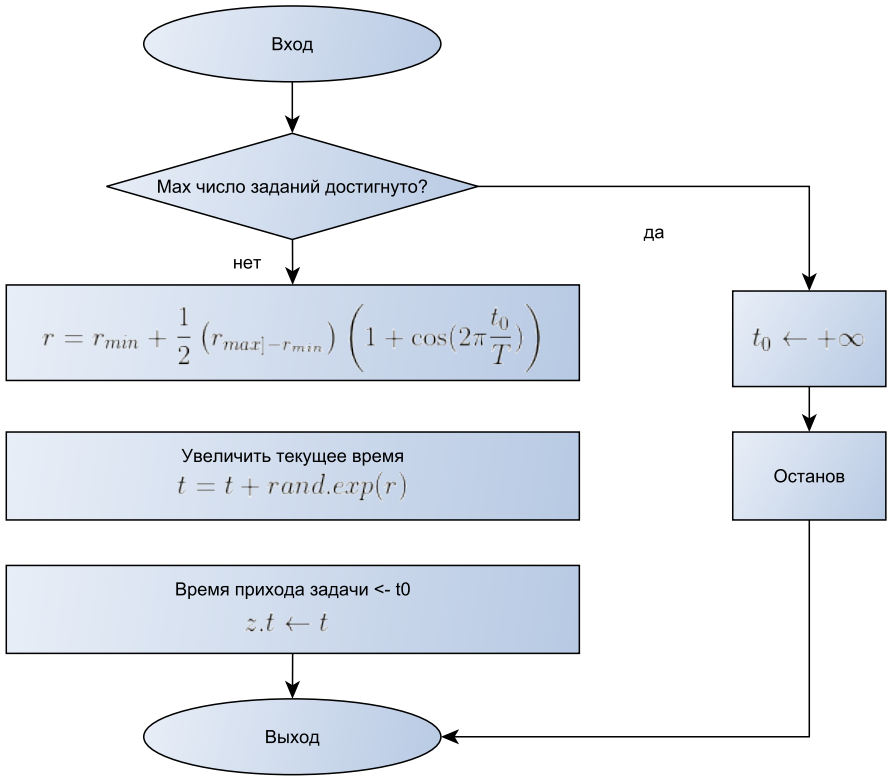


Рисунок 4.2 — Блок-схема генератора примеров.

функционалов 1-го порядка для многолинейной модели СА с ограниченными информационными ресурсами. Для практических задач характерно динамическое поступление работ, которое вносит в модель стохастические компоненты. Также вероятностным является выбор пользователем сочетания k из K ресурсов, которое считалось случайным, независимым и равномерно распределенным, т.е. выбиралось сочетание C_K^k с вероятностью использования каждого ресурса q_k. Для каждого ресурса генерируется объём задачи V_{jk}. Таким образом, работе соответствует подмножество используемых ресурсов $[s]_j$ и объём информации, извлекаемой из каждого ресурса $[V]_j$. Ожидаемое число запросов в короткий интервал времени распределено по Пуассону, соответственно интервал времени между запросами определяется экспоненциальным распределением. Однако частота запросов на длинных промежутках времени может изменяться. В данной статье при моделировании был принят вариант, когда частота меняется от времени суток по синусоиде

$$r_j = r_{\min} + \frac{1 - \cos(2\pi t/\tau)}{2}(r_{\max} - r_{\min})$$

с минимумом ночью и максимумом днем по внутреннему времени модели. Здесь τ — период колебаний, $t \in \mathbb{R}$ — текущее время.

Далее, была поставлена задача поиска эвристического правила диспетчеризации, которая бы минимизировала совокупность критериев оптимальности расписания. Выбор критериев оптимальности производился из критериев, представленных в [135—137], и обусловлен их применимостью в рассматриваемой задаче.

Было произведено математическое моделирование для каждого ПД по следующему алгоритму. Число ресурсов при моделировании варьировалось от двух до десяти. Задачи, генерируемые при моделировании, обращались к каждому из ресурсов с вероятностью $q_k = 50$ % и затребовали из ресурса объём информации, распределенное равновероятно от нуля до максимального количества. В моделировании эмулировались ресурсы, способные обработать $r_{\max} = 0.75$ задач в час при условии равномерного распределения их объёма. Было промоделировано поведение системы за месяц для различных средних частот запросов r_{ave} от $0.1r_{\max}$ до $0.9r_{\max}$. Интервалы между событиями считались случайно распределёнными в соответствии с экспоненциальным распределением, параметр которого варьировался в зависимости от времени системы от $0.2r_{\text{ave}}$ до $1.8r_{\text{ave}}$ в период наименьшей и наибольшей загруженности соответственно.

Моделировалась работа системы в течение условного месяца (30 моделируемых суток). Значения критериев были усреднены по 100 запускам для каждого сочетания моделируемых параметров. Новые задачи генерировались в случайные моменты времени, распределенные экспоненциально с параметром распределения r, варьирующимся в течение модельных суток. Обслуживание заявок из очереди производилось в соответствии с выбранными правилами диспетчеризации. Время обслуживания оценивалось согласно приведенным ранее критериям. Результаты моделирования приведены в таблице 13 и на рисунке 4.3.

4.1.3 Анализ результатов вычислительного эксперимента

В таблице 13 и на графиках, приведённых на рисунке 4.3, приведены зависимости значений критериев оптимальности от интенсивности входящего потока заявок для различных эвристических правил диспетчеризации. Наименьшее увеличение значения критерия с ростом загруженности системы является показателем удачности выбора эвристики.

Можно заметить, что при использовании одного и того же критерия оптимальности увеличение количества ресурсов оказывает влияние на то, какое правило диспетчеризации оказывается более выгодной. Так, например, в случае двух ресурсов по показателю PT эвристика LTPT показывает наилучший результат, то-

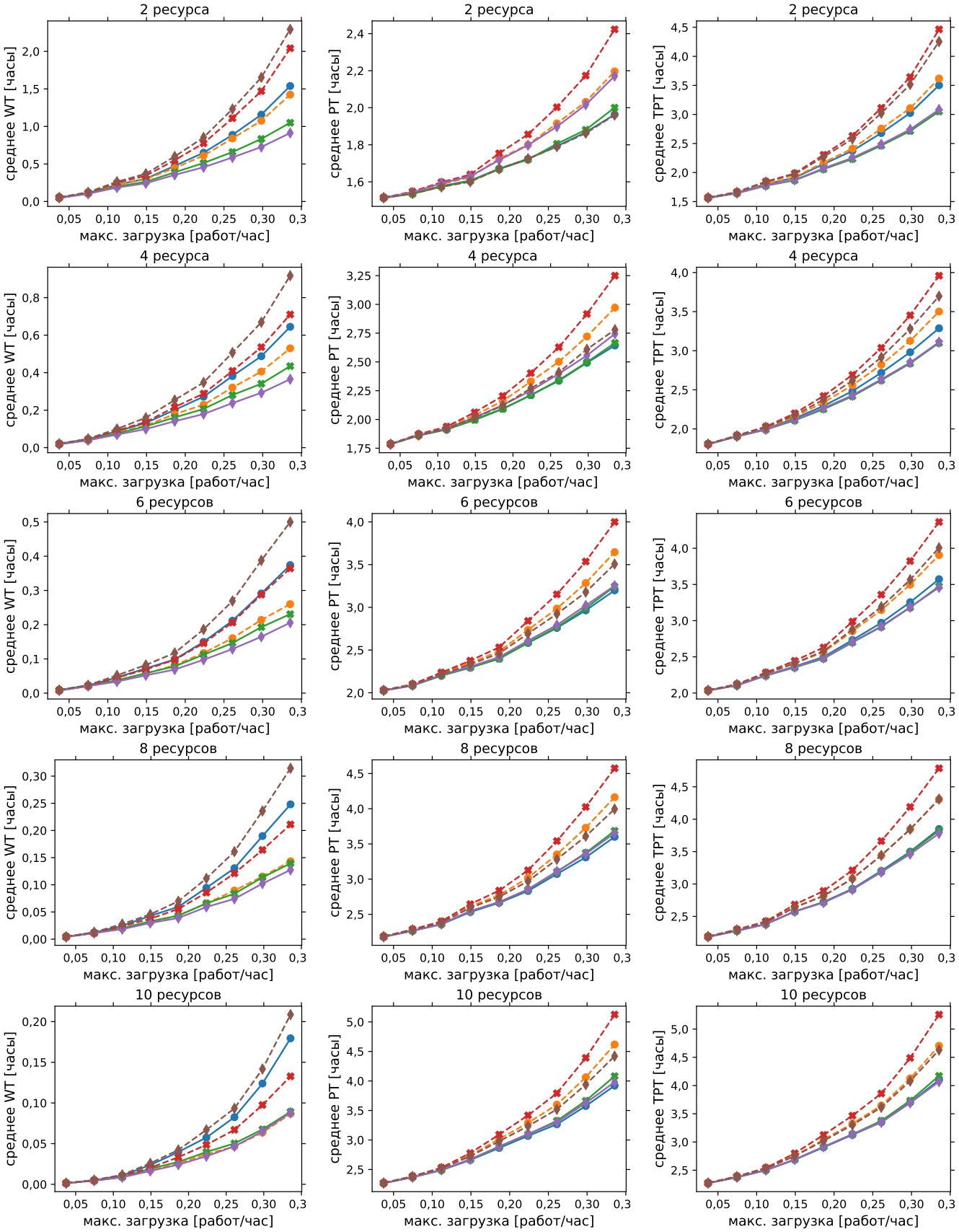


Рисунок 4.3 — Результаты моделирования времён-критериев оптимальности расписания для различных правил определения приоритета, количества ресурсов и средних частот запросов. Для маркировки ПД использованы следующие обозначения: FIFO, SPT, STPT, LIFO, LPT, LTPT.

гда как с увеличением количества ресурсов она начинает «проигрывать» таким дисциплинам, как STPT и SPT по рассматриваемому критерию.

Выбор критерия оптимальности также влияет на то, какое правило диспетчеризации окажется наиболее выгодной для распределения задач в системе. Так, в случае двух ресурсов можно заметить, что эвристика LTPT оказывается самой выгодной по критерию PT, тогда как по критериям WT и TPT эта стратегия оказывается одной из самых неудачных. Таким образом, конфигурация системы массового обслуживания и выбор критериев оптимизации обслуживания является существенным фактором, который следует учитывать при выборе используемого ПД.

Варьирование количества ресурсов и загрузки системы позволяет изучить, насколько часто то или иное правило диспетчеризации оказывается «лучшей» или «худшей» по рассматриваемым критериям для исследуемой СМО (см. рисунок 4.3). Так, линии, соответствующие эвристикам LPT и LTPT, оказываются в верхней части графика (среди худших результатов) более чем в 2/3 случаев, что позволяет сделать вывод о том, что применение данных ПД является невыгодным в рассмотренных условиях. В то же время эвристика STPT оказалась среди «лучших» в 0.93 % случаев (для некоторых из этих случаев близкий к ней результат показывают FIFO и SPT), что может служить рекомендацией к её использованию для решения данных классов задач. Качественно данный результат можно объяснить тем, что события, которые стоят в начале очереди, вносят больший вклад в сумму при последующих расчетах.

Результаты данного анализа могут быть использованы при разработке оригинальных планировщиков задач в информационно-вычислительных системах (ИВС) рассмотренной архитектуры и при настройке существующих планировщиков задач в ИВС реального времени.

Таблица 13 — Результаты моделирования времен-критериев оптимальности расписания для различных правил определения приоритета, количества хранилищ и средних частот запросов

			Алгоритм					
			FIFO	LIFO	SPT	LPT	STPT	LTPT
s	r_{max} [раб./ч]	Критерий оптим.						
2	0.04	WT	0.05	0.05	0.05	0.05	0.05	0.06
		PT	1.51	1.52	1.51	1.51	1.52	1.51
		TPT	1.56	1.57	1.56	1.57	1.56	1.57
	0.11	WT	0.22	0.22	0.20	0.25	0.19	0.26
		PT	1.58	1.59	1.57	1.60	1.59	1.57
		TPT	1.80	1.81	1.77	1.84	1.78	1.84
	0.19	WT	0.47	0.45	0.39	0.55	0.35	0.60
		PT	1.67	1.73	1.67	1.75	1.72	1.67
		TPT	2.15	2.17	2.06	2.30	2.07	2.27
	0.26	WT	0.89	0.84	0.66	1.11	0.59	1.23
		PT	1.79	1.92	1.81	2.00	1.90	1.79
		TPT	2.68	2.76	2.46	3.11	2.48	3.02
	0.34	WT	1.54	1.42	1.05	2.04	0.91	2.29
		PT	1.96	2.20	2.00	2.42	2.17	1.96
		TPT	3.50	3.62	3.05	4.46	3.08	4.25
3	0.04	WT	0.03	0.03	0.03	0.03	0.03	0.03
		PT	1.64	1.64	1.64	1.64	1.64	1.64
		TPT	1.67	1.67	1.67	1.67	1.67	1.67
	0.11	WT	0.12	0.12	0.11	0.13	0.10	0.14
		PT	1.75	1.76	1.75	1.77	1.76	1.75
		TPT	1.87	1.88	1.85	1.90	1.86	1.89
	0.19	WT	0.29	0.27	0.24	0.32	0.21	0.37
		PT	1.91	1.97	1.90	2.01	1.95	1.92
		TPT	2.20	2.24	2.14	2.33	2.16	2.29
	0.26	WT	0.54	0.47	0.40	0.62	0.34	0.73

Продолжение на следующей странице

			Алгоритм					
			FIFO	LIFO	SPT	LPT	STPT	LTPT
s	r_{max} [раб./ч]	Критерий оптим.						
		PT	2.09	2.23	2.10	2.33	2.17	2.13
		TPT	2.63	2.70	2.49	2.95	2.51	2.85
	0.34	WT	0.90	0.79	0.62	1.08	0.53	1.32
		PT	2.33	2.62	2.35	2.87	2.47	2.40
		TPT	3.23	3.41	2.97	3.95	3.00	3.71
4	0.04	WT	0.02	0.02	0.02	0.02	0.02	0.02
		PT	1.78	1.79	1.78	1.79	1.79	1.78
		TPT	1.80	1.81	1.80	1.81	1.80	1.81
	0.11	WT	0.09	0.08	0.08	0.09	0.07	0.10
		PT	1.91	1.93	1.91	1.94	1.92	1.92
		TPT	2.00	2.01	1.99	2.03	1.99	2.02
	0.19	WT	0.20	0.18	0.16	0.22	0.14	0.25
		PT	2.09	2.16	2.09	2.20	2.12	2.12
		TPT	2.29	2.34	2.25	2.42	2.27	2.37
	0.26	WT	0.38	0.32	0.28	0.41	0.24	0.51
		PT	2.34	2.50	2.34	2.63	2.40	2.41
		TPT	2.72	2.82	2.62	3.04	2.63	2.92
	0.34	WT	0.64	0.53	0.43	0.71	0.36	0.92
		PT	2.64	2.97	2.67	3.25	2.74	2.78
		TPT	3.29	3.50	3.10	3.96	3.11	3.70
5	0.04	WT	0.02	0.01	0.01	0.02	0.01	0.02
		PT	1.89	1.89	1.89	1.89	1.89	1.89
		TPT	1.91	1.91	1.91	1.91	1.91	1.91
	0.11	WT	0.06	0.05	0.05	0.06	0.05	0.07
		PT	2.05	2.07	2.04	2.08	2.05	2.06
		TPT	2.10	2.12	2.10	2.14	2.10	2.13
	0.19	WT	0.14	0.13	0.12	0.15	0.10	0.17
		PT	2.29	2.37	2.28	2.42	2.31	2.34

Продолжение на следующей странице

			Алгоритм					
			FIFO	LIFO	SPT	LPT	STPT	LTPT
s	r_{max} [раб./ч]	Критерий оптим.						
		TPT	2.43	2.50	2.40	2.57	2.41	2.52
	0.26	WT	0.27	0.22	0.20	0.28	0.17	0.36
		PT	2.55	2.74	2.55	2.89	2.60	2.67
		TPT	2.82	2.97	2.75	3.17	2.77	3.03
	0.34	WT	0.49	0.37	0.31	0.51	0.27	0.68
		PT	2.97	3.37	3.00	3.69	3.04	3.20
		TPT	3.46	3.74	3.31	4.21	3.31	3.88
6	0.04	WT	0.01	0.01	0.01	0.01	0.01	0.01
		PT	2.03	2.03	2.03	2.03	2.03	2.03
		TPT	2.04	2.04	2.03	2.04	2.03	2.04
	0.11	WT	0.05	0.04	0.04	0.04	0.03	0.05
		PT	2.20	2.23	2.20	2.24	2.21	2.21
		TPT	2.25	2.27	2.24	2.28	2.24	2.27
	0.19	WT	0.10	0.08	0.08	0.10	0.07	0.12
		PT	2.40	2.48	2.40	2.53	2.42	2.46
		TPT	2.50	2.57	2.48	2.63	2.49	2.58
	0.26	WT	0.21	0.16	0.15	0.21	0.13	0.27
		PT	2.76	2.99	2.77	3.15	2.79	2.92
		TPT	2.97	3.15	2.91	3.36	2.92	3.19
	0.34	WT	0.37	0.26	0.23	0.36	0.21	0.50
		PT	3.20	3.65	3.25	4.00	3.25	3.51
		TPT	3.57	3.91	3.48	4.36	3.46	4.01
7	0.04	WT	0.01	0.01	0.01	0.01	0.01	0.01
		PT	2.08	2.08	2.08	2.08	2.08	2.08
		TPT	2.08	2.09	2.08	2.09	2.08	2.09
	0.11	WT	0.03	0.03	0.03	0.03	0.02	0.03
		PT	2.29	2.32	2.29	2.33	2.29	2.31
		TPT	2.32	2.34	2.32	2.36	2.32	2.34

Продолжение на следующей странице

s	r_{max} [раб./ч]	Критерий оптим.	Алгоритм FIFO	LIFO	SPT	LPT	STPT	LTPT
	0.19	WT	0.08	0.06	0.06	0.07	0.05	0.09
		PT	2.56	2.67	2.57	2.73	2.58	2.65
		TPT	2.64	2.73	2.63	2.81	2.63	2.74
	0.26	WT	0.17	0.12	0.11	0.16	0.10	0.21
		PT	2.94	3.20	2.96	3.38	2.97	3.13
		TPT	3.11	3.32	3.08	3.54	3.07	3.34
	0.34	WT	0.32	0.20	0.18	0.29	0.16	0.42
		PT	3.40	3.94	3.50	4.32	3.48	3.75
		TPT	3.72	4.14	3.68	4.62	3.64	4.17
8	0.04	WT	0.00	0.00	0.00	0.00	0.00	0.00
		PT	2.18	2.18	2.18	2.19	2.18	2.18
		TPT	2.19	2.19	2.18	2.19	2.18	2.19
	0.11	WT	0.02	0.02	0.02	0.02	0.02	0.03
		PT	2.36	2.39	2.36	2.40	2.36	2.38
		TPT	2.38	2.41	2.38	2.42	2.38	2.41
	0.19	WT	0.06	0.04	0.04	0.06	0.04	0.07
		PT	2.66	2.77	2.67	2.84	2.67	2.75
		TPT	2.72	2.82	2.72	2.89	2.71	2.82
	0.26	WT	0.13	0.09	0.08	0.12	0.07	0.16
		PT	3.07	3.35	3.10	3.54	3.11	3.28
		TPT	3.20	3.44	3.19	3.66	3.18	3.44
	0.34	WT	0.25	0.14	0.14	0.21	0.13	0.31
		PT	3.60	4.16	3.69	4.57	3.65	4.00
		TPT	3.85	4.31	3.83	4.79	3.78	4.31
9	0.04	WT	0.00	0.00	0.00	0.00	0.00	0.00
		PT	2.25	2.25	2.25	2.25	2.25	2.25
		TPT	2.25	2.25	2.25	2.25	2.25	2.25
	0.11	WT	0.02	0.01	0.01	0.02	0.01	0.02

Продолжение на следующей странице

s	r_{max} [раб./ч]	Критерий оптим.	Алгоритм FIFO	LIFO	SPT	LPT	STPT	LTPT
		PT	2.43	2.46	2.43	2.47	2.44	2.45
		TPT	2.44	2.48	2.45	2.49	2.45	2.47
	0.19	WT	0.04	0.03	0.03	0.04	0.03	0.05
		PT	2.74	2.87	2.75	2.93	2.76	2.84
		TPT	2.79	2.90	2.79	2.97	2.78	2.89
	0.26	WT	0.11	0.06	0.06	0.09	0.06	0.13
		PT	3.21	3.54	3.27	3.74	3.26	3.44
		TPT	3.32	3.60	3.34	3.83	3.32	3.57
	0.34	WT	0.22	0.12	0.12	0.17	0.11	0.27
		PT	3.77	4.45	3.92	4.90	3.84	4.24
		TPT	4.00	4.56	4.03	5.07	3.95	4.51
10	0.04	WT	0.00	0.00	0.00	0.00	0.00	0.00
		PT	2.27	2.27	2.27	2.27	2.27	2.27
		TPT	2.27	2.27	2.27	2.27	2.27	2.27
	0.11	WT	0.01	0.01	0.01	0.01	0.01	0.01
		PT	2.49	2.52	2.49	2.53	2.49	2.52
		TPT	2.50	2.53	2.50	2.54	2.50	2.53
	0.19	WT	0.04	0.03	0.03	0.03	0.02	0.04
		PT	2.87	3.02	2.88	3.09	2.89	2.98
		TPT	2.90	3.04	2.91	3.12	2.92	3.02
	0.26	WT	0.08	0.05	0.05	0.07	0.05	0.09
		PT	3.27	3.60	3.32	3.79	3.30	3.52
		TPT	3.35	3.65	3.37	3.86	3.35	3.61
	0.34	WT	0.18	0.09	0.09	0.13	0.09	0.21
		PT	3.92	4.62	4.08	5.12	3.98	4.42
		TPT	4.10	4.70	4.17	5.26	4.07	4.63

4.1.4 Комплекс программ для реализации численного эксперимента

В данном параграфе приводится описание комплекса программ, написанных на ЯП Python 3 для моделирования процесса составления адаптивных распи-

саний для многолинейной системы с ограниченными восполнимыми ресурсами для зависимых по времени выполнения запросов.

Программа работает по следующему алгоритму. В опциях командной строки задаются параметры численного эксперимента, после чего программа запускается из консоли командой `python3 main.py` из директории, в которую установлена программа. Запуск главной программы инициирует процесс моделирования поступления заявок в систему, и составления расписания обслуживания поступающих заявок, в которое оперативно вносятся изменения, согласно вновь возникающей информации о поступлении новых заявок в систему. Каждая очередная версия расписаний записывается в логи системы. Итоговое расписание для обработки задач записывается в файл `results.csv`. Затем скрипт draw.py на основе полученных результатов строит графики для выбранных параметров моделирования и критериев оптимальности.

Результатами выполнения программы являются файл `results.csv` с результатами моделирования в виде таблицы и графики, отображающие результаты моделирования в визуальном формате.

Структура взаимодействия модулей представлена на рисунке 4.4.

Общие сведения о программе.

Программа называется «Программный комплекс для моделирования процесса составления адаптивных расписаний для многолинейной системы с ограниченными восполнимыми ресурсами». Для работа программы необходим интерпретатор языка Python3.8, пакеты math, copy, weakref, locale, random, csv, matplotlib и numpy. Для работы программы необходимо порядка 20 МБ оперативной памяти. Объем исходного текста программы составляет 9397 байта.

Функциональное назначение. Предназначена для для проведения вычислительных экспериментов, моделирующих составление адаптивных расписаний обработки задач в многолинейной системе массового обслуживания с ограниченными в терминах качественной доступности ресурсами.

Структура программы.

Модуль `utils/generator.py` генерирует задачи с временами постановки в очередь, случайно распределёнными по Пуассону с изменяющимся параметром λ, и объёмами, равномерно случайно распределёнными от нуля до некоторого максимума. Задачи описываются классами `Task` из модуля `classes/task.py` для подзадач и `Job` из модуля `classes/job.py` для совокупных запросов из нескольких подзадач. Далее задачи ставятся в очередь

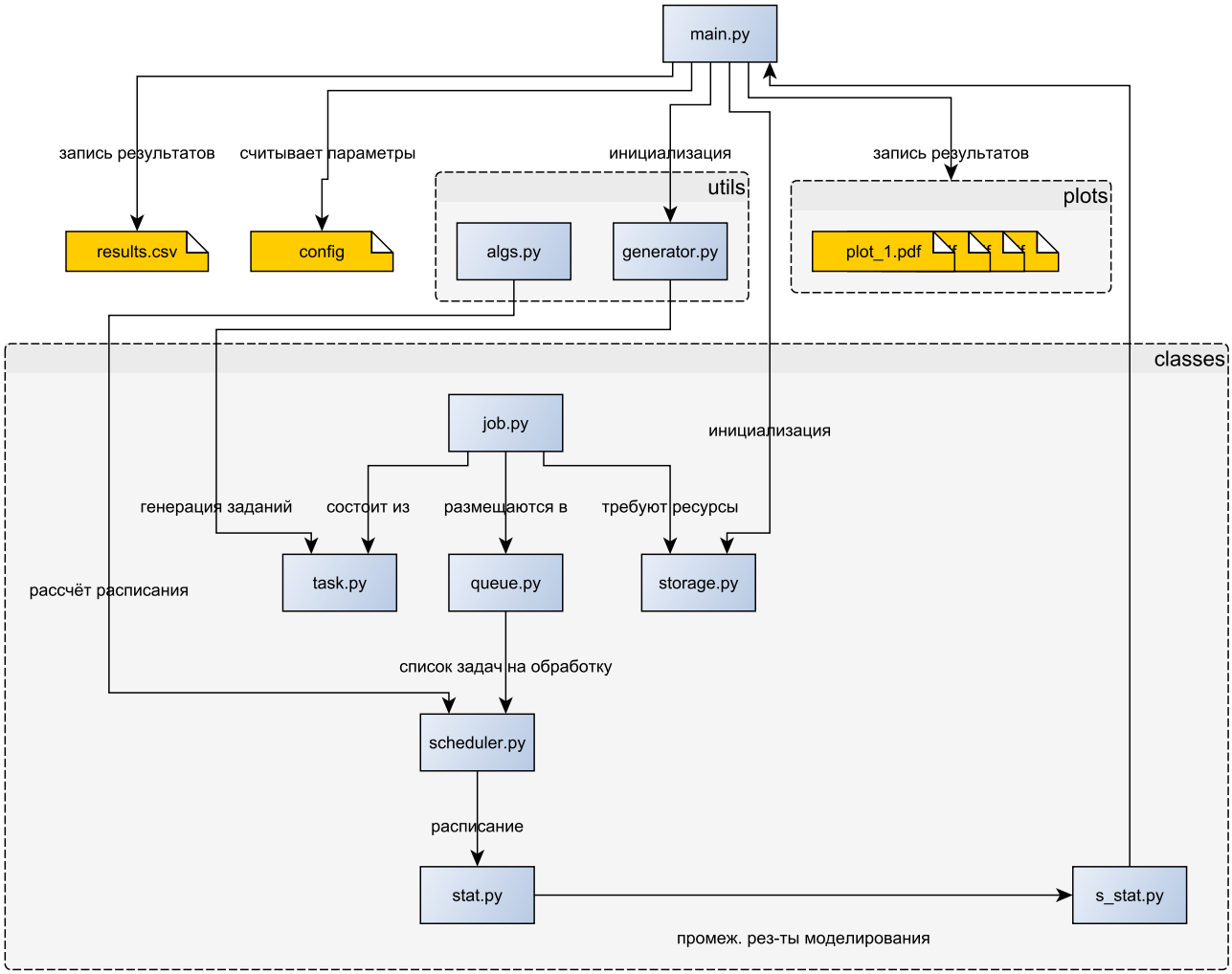


Рисунок 4.4 — Диаграмма потока данных при взаимодействии модулей.

(класс `Queue` из модуля `classes/queue.py`). К очередям привязаны хранилища (класс `Storage` из модуля `classes/storage.py`), ресурсы которых задействуются при обработке запросов. Порядком задач в очереди управляет планировщик (класс `Scheduler` из модуля `classes/scheduler.py`), задействующий правила диспетчеризации (ПД) из класса приоритето-порождающих функционалов 1-го порядка, описанные в модуле `utils/algs.py`. Далее в классе `Stat` (модуль `classes/stat.py`) для очереди рассчитываются различные критерии оптимальности, такие как суммарное время ожидания, время обработки, и общее время нахождения в очереди. Сводная статистика для нескольких очередей с различными ПД формируется классом `SStat` из модуля `classes/s_stat.py`, и передаётся в основной модуль программы `main.py` для вывода. Также в комплекс входит модуль `utils/draw.py` для визуализации результатов моделирования в виде графиков, которые записываются в папку `plots` для промежуточных

результатов и выводятся на экран и записываются в папку `plots` для обобщенных итоговых результатов.

Требования к программному окружению. Операционная система Linux, Windows 10, 8 либо 7, MacOS 10.0.4 и новее. Интерпретатор языка Python 3.6 или новее.

Эксплуатация программы. Для запуска программы необходимо задать параметры модели: число хранилищ, объём данных в хранилище, вероятность обращения к хранилищу в задаче, разница между максимальной и минимальной частотой поступления задач в течение периода, моделируемый диапазон времени, число повторов моделирования для набора статистики. На выходе программы получаются следующие файлы:

– файл results.csv;
– графики в формате .pdf для каждого критерия оптимальности, которые записываются в поддиректорию plots директории, в которую установлена программа.

После задания параметров достаточно запустить скрипт `main.py`.

На рисунках 4.5, 4.6 показана работающая программа, осуществляющая: вывод промежуточных данных моделирования в командную строку (рисунок 4.5) и вывод финальных результатов работы программы в графическом виде (рисунок 4.6).

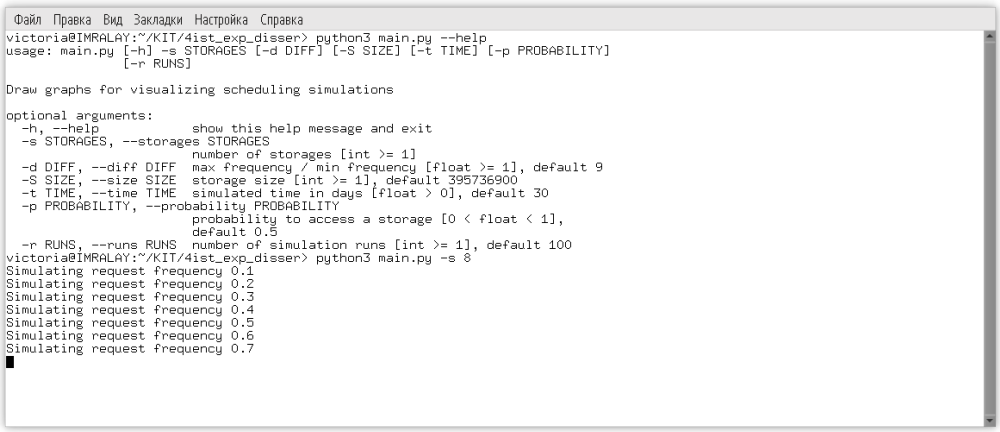


Рисунок 4.5 — Запущенная программа на Python, вывод информации в командную строку.

Свидетельство о регистрации программы представлено в приложении Б.2.

4.2 Создание имитационной модели СА

Под имитационным моделированием (simulation modelling) понимают разновидность аналогового моделирования, реализуемого с помощью набора ма-

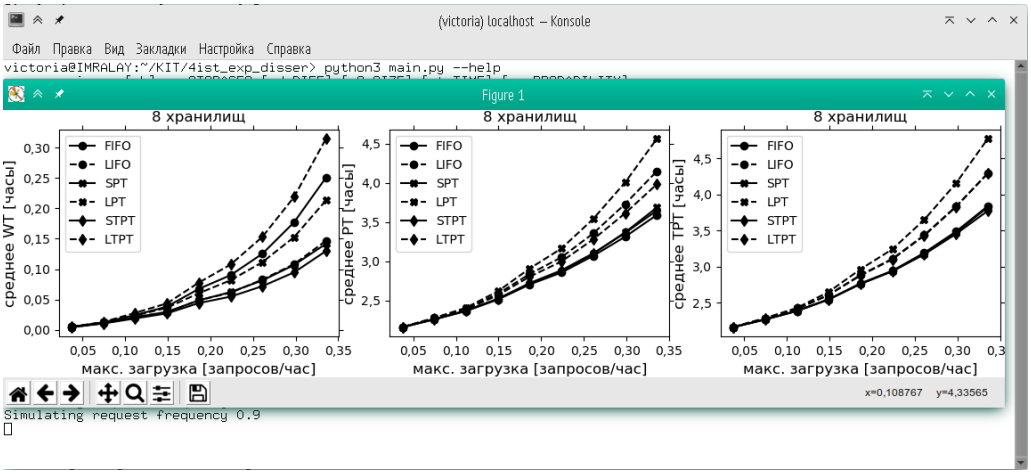


Рисунок 4.6 — Запущенная программа на Python, вывод информации в виде графика (показан один из графиков для случая 8 хранилищ.

тематических инструментальных средств, специальных имитирующих компьютерных программ и технологий программирования, позволяющих посредством процессов-аналогов провести целенаправленное исследование структуры и функций реального сложного процесса в памяти компьютера в режиме «имитации», выполнить оптимизацию некоторых его параметров [146]. Программный комплекс, используемый для имитации поседения СА будем называть имитационной моделью.

В процессе вычислительного эксперимента будем исследовать поведение системы при различных комбинациях параметров её внутреннего устройства и внешних переменных факторов согласно модели 2.7, описанной в гл. 2. Различные комбинации таких параметров, подаваемые на вход имитационной модели, будет называть примером. Экземпляр объекта имитационной модели, сгенерированный программным комплексом для соответствующего входящего примера будем называть симуляцией.

4.2.1 Постановка численного эксперимента

На данном этапе численного эксперимента осуществляется преобразование математической модели в моделирующий алгоритм. Выполним это преобразование в 2 этапа:

– Исследование принципиальной архитектуры системы моделирования с помощью метода моделирования «Чёрный ящик»;
– Уточнение реализации компонентов системы моделирования.

Система имитационного моделирования как «Чёрный ящик»

Используем метод чёрного ящика (см. рисунок 4.7) для определения желаемого поведения системы имитационного моделирования.

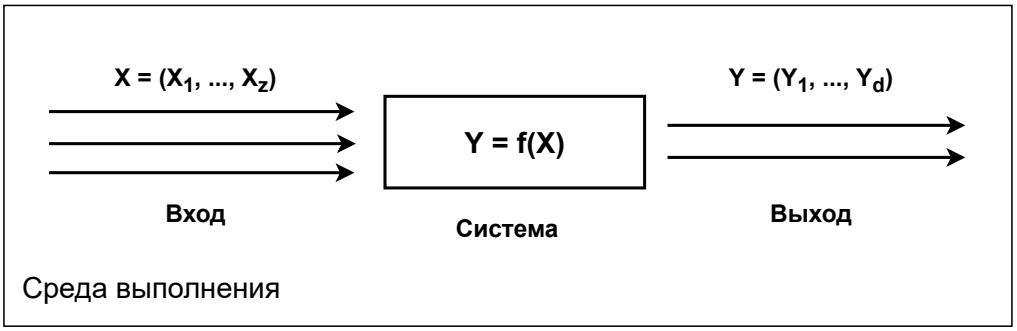


Рисунок 4.7 — Модель «Чёрный ящик».

Определим начальные условия, которые подаются на вход системы:

– Количество ресурсов S и скорости доступа к ним μ_s;
– Количество приборов m, информация о соответствии приборов ресурсам (идентичные параллельные приборы либо специальные приборы, соответствующие определённым ресурсам);
– Тип и параметры генерации работ: $\{n, p_j, V_j = (V_{ji_1}, \ldots, V_{ji_k}), k, r_j\}$;
– Алгоритмы составления расписаний - выбор из множества алгоритмов, рассмотренный в главе 3;
– Количество генерируемых задач различной размерности - для формирования репрезентативной статистической выборки;
– Используемый критерий оптимальности расписания из множества $F_{aim} = \{C_{max}, \sum_{j=1}^{n} C_j, C_a\}$.

Определим следующие выходные данные имитационного моделирования:

– Расписание для каждого сгенерированного примера, сформулированное следующим образом:

$$P_l : (s_{ji_1}, \ldots, s_{ji_k}), \quad l = 1, \ldots, m, j = 1, \ldots, n, i_k \in \{1, \ldots, k\}, k = 1, \ldots, S$$

сохраняемое в подпапку `schedules` корневой директории ПО ЭВМ в формате `<run-id>-<alg_name>.txt`;
– Графическое изображение каждого построенного расписания в виде диаграммы Ганта с именем в формате `<run-id>-<alg_name><job-id>-<task-id>.pdf`, сохраненное в подпапке `images` корневой директории ПО ЭВМ;
– Файлы регистрации событий (т.н. логи), записанные в подпапку `logs` корневой директории ПО ЭВМ в формате `<текущее время>:<сообщение>`.

– Сравнительную таблицу производительности исследуемых алгоритмов в формате `.csv`.

Тогда принципиальными ступенями обработки данных внутри ЧЯ будут выступать:

– Чтение входящих данных;
– Генерация примеров;
– Составление расписаний с использованием алгоритмов, описанных в главе 3;
– Запись событий системы в файлы регистрации событий;
– Запись выходных файлов с результатами моделирования в форматах, описанных выше.

Таким образом, имеем следующее принципиальное устройство системы (рисунок 4.8).

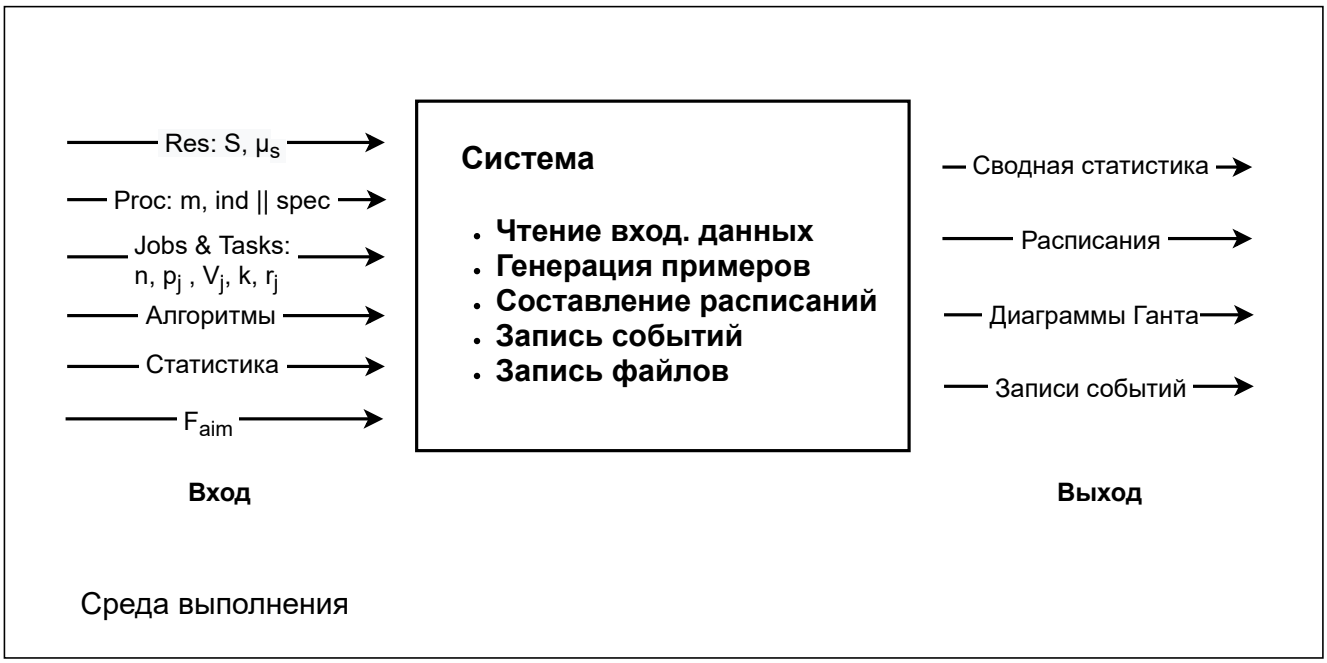


Рисунок 4.8 — Представление системы имитационного моделирования в виде «Белого ящика».

Перейдём к описанию внутреннего устройства системы имитационного моделирования. В ходе вычислительного эксперимента испытания производились с использованием моделирующего алгоритма, представленного в виде блок-схемы на рисунке 4.9.

Суть моделирующего алгоритма состоит в следующем. На вход имитационной модели подаются начальные условия эксперимента. Далее в цикле происходит изменение свободных переменных модели, генерация примеров и симуляций и расчёт расписаний по выбранным пользователем алгоритмами с последующей

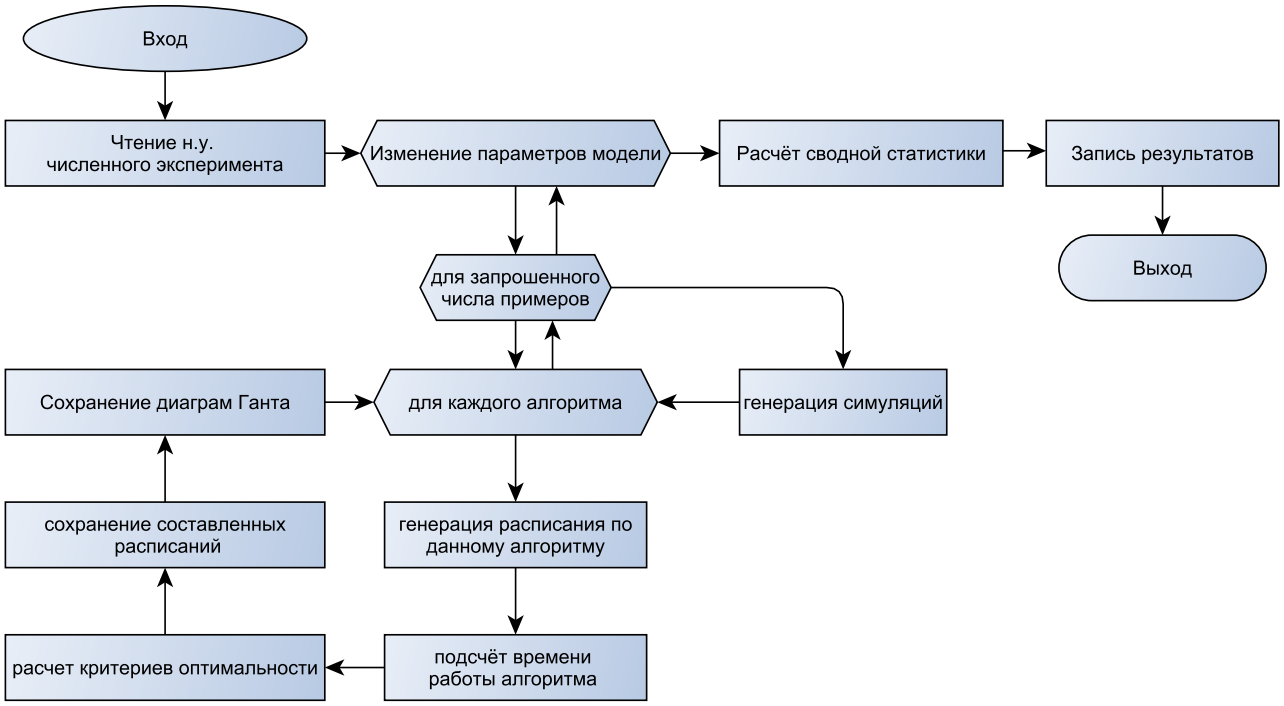


Рисунок 4.9 — Блок-схема моделирующего алгоритма для проведения численного эксперимента.

оценкой составленного расписания по критериям оптимальности, указанным в начальных условиях.

Для достижения точности результатов для каждого заданного примера испытания производились выбранное пользователем числом раз. Этот параметр также входят в начальные условия эксперимента, служащие входом СИМ.

По окончании проведения статистических испытаний производился расчёт сводных статистических характеристик и запись результатов в выбранных пользователем форматах.

Ключевой подпрограммой системы имитационного моделирования является модуль генерации симуляций, реализующий соответствующий алгоритм генерации, блок-схема которого изображена на рисунке 4.10.

Модуль создаёт объекты, описывающие приборы и ресурсы, и добавляет их в симуляцию. Затем в цикле создаются пустые работы со своими временами прихода. Для каждой работы генерируется число содержащихся в ней задач, случайно распределённое от единицы до максимума, и производится выборка соответствующих им ресурсов без повторений. Далее для каждого ресурса из выборки создаётся задача случайно распределённого по Пуассону объёма и добавляется к

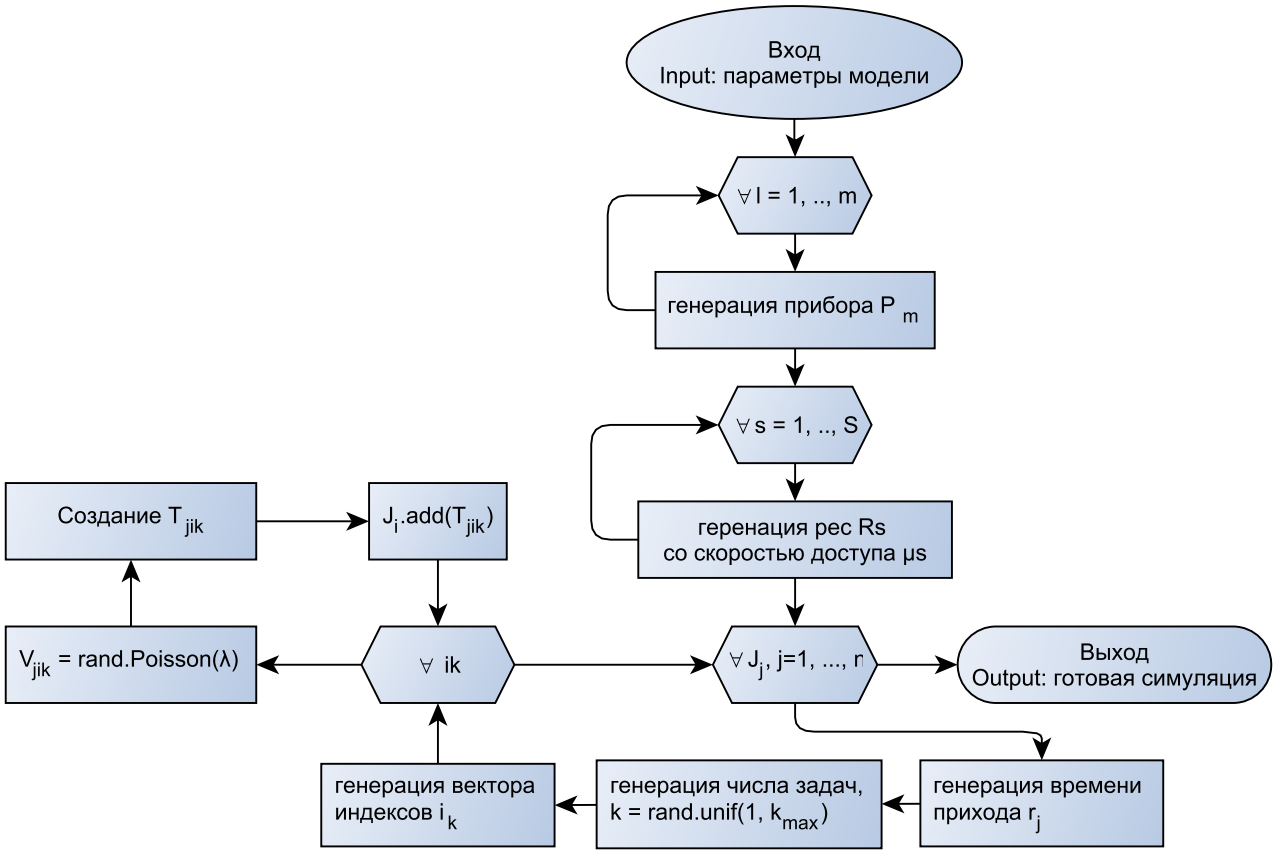


Рисунок 4.10 — Блок-схема алгоритма генерации задач составления расписаний.

работе. На этом генератор завершает работу и возвращает полученную симуляцию.

4.2.2 Описание реализации численного эксперимента

Разработанная имитационная модель позволяет нам изучить следующие свойства алгоритма двухуровневой диспетчеризации (АДД), важные для решения задачи (2.7):

– Произвести анализ производительности АДД для среднего случая и установить численные значения констант C_l, C_u, n_0 для различных параметров модели;
– Произвести сравнительное исследование эффективности составления расписаний с т.з. критериев оптимальности C_{max}, $\sum_{ji} C_{ji}$, C_a для АДД и алгоритмов, основанных на приоритето-порождающих функционалах первого порядка.

Для достижения указанных целей были произведены описанные ниже численные эксперименты.

Эксперимент 1

Для фиксированного числа приборов m и ресурсов S с было произведено исследование производительности АДД при растущем $n \in [1, 10^5]$.

Для каждого n усреднение времени работы алгоритма было произведено по $X = 100$ запускам.

Рассматривались следующие варианты значений m и S:

– Малые значения параметров m и S (≈ 10);
– Средние значения m, S ($\approx 10^2$);
– Относительно большие значения m, S ($\geq 10^3$).

Число k_{max} было взято равным S. Таким образом, количество задач в работе k в статистических испытаниях можно оценить как

$$k = M[1,S] = \frac{S+1}{2},$$

поскольку считаем k распределённым равномерно. Таким образом, данный вычислительный эксперимент позволяет нам оценить время выполнения алгоритма для усредненного случая.

Также был проведён схожий численный эксперимент, но с фиксированным m, n и с изменением числе ресурсов S от 1 до 10^5.

Анализ результатов данного численного эксперимента один изложен в части 4.2.3.

Эксперимент 2

Целью данного эксперимента являлось сравнение эффективности построения расписаний по выбранным критериям оптимальности для алгоритмов, введённых рассмотрение в главе 3. А именно: для изменяющегося числа работ n и некоторых фиксированных m, S производились повторные статистические испытания с целью сравнения средних значений критериев оптимальности для различных алгоритмов.

В качестве критериев оптимальности были взяты C_{max}, $\sum_{ji} C_{ji}$, C_a. В качестве алгоритмов — эвристический алгоритм двухуровневой диспетчеризации и семейство алгоритмов, основанных на приоритето-порождающих функционалах 1-го порядка, включающие в себя такие правила диспетчеризации как: FIFO, LIFO, SPT, LPT, STPT, LTPT.

В ходе численного эксперимента моделирование производилось относительно трёх следующих случаев:

– $m < S$;
– $m = S$;
– $m > S$.

Анализ результатов представлен в параграфе 4.2.3 данной работы.

4.2.3 Анализ результатов вычислительного эксперимента

Для проверки аналитически полученной временной сложности алгоритма было произведено моделирование времён работы алгоритма двухуровневой диспетчеризации для разного количества работ. Полученные времена работы алгоритма двухуровневой диспетчеризации были аппроксимированы функцией $C\,nS\log nS$. Из рисунка 4.11 видно, что в исследованной области время наихудшего случая составляет не более $8\times10^{-5}nS\log nS$, а время наилучшего случая — не менее $2\times10^{-6}nS\log nS$, что согласуется с аналитически полученным значением временной сложности

$$T = O(nS\log nS) = \Omega(nS\log nS).$$

На рисунке 4.12 представлены графически результаты численного эксперимента 2. Для каждого правила диспетчеризации приведено отношение критерия оптимальности для этого правила и для алгоритма двухуровневой диспетчеризации.

Данные результаты показывают, что из приоритето-порождающих функционалов 1-го порядка по критерию $\sum_{ji} C_{ji}$ стабильно худшим является LPT, лучшим — SPT. При этом лучший по критерию C_a — STPT, худший — LTPT. Критерий $C_{\max}$ оказывается практически нечувствительным к выбору функционала.

Что касается алгоритма двухуровневой диспетчеризации, он превосходит приоритето-порождающие функционалы почти при всех параметрах модели, при этом особенно ярко его преимущества проявляются при условии, что число приборов m меньше числа ресурсов S. Если $m \geq S$, то с ростом числа работ алгоритмы SPT и STPT начинают работать так же хорошо, как алгоритм двухуровневой диспетчеризации, а SPT даже может превосходить его по критерию $\sum_{ji} C_{ji}$ на 2–3%.

Также численные значения критериев оптимальности приведены в таблице 14.

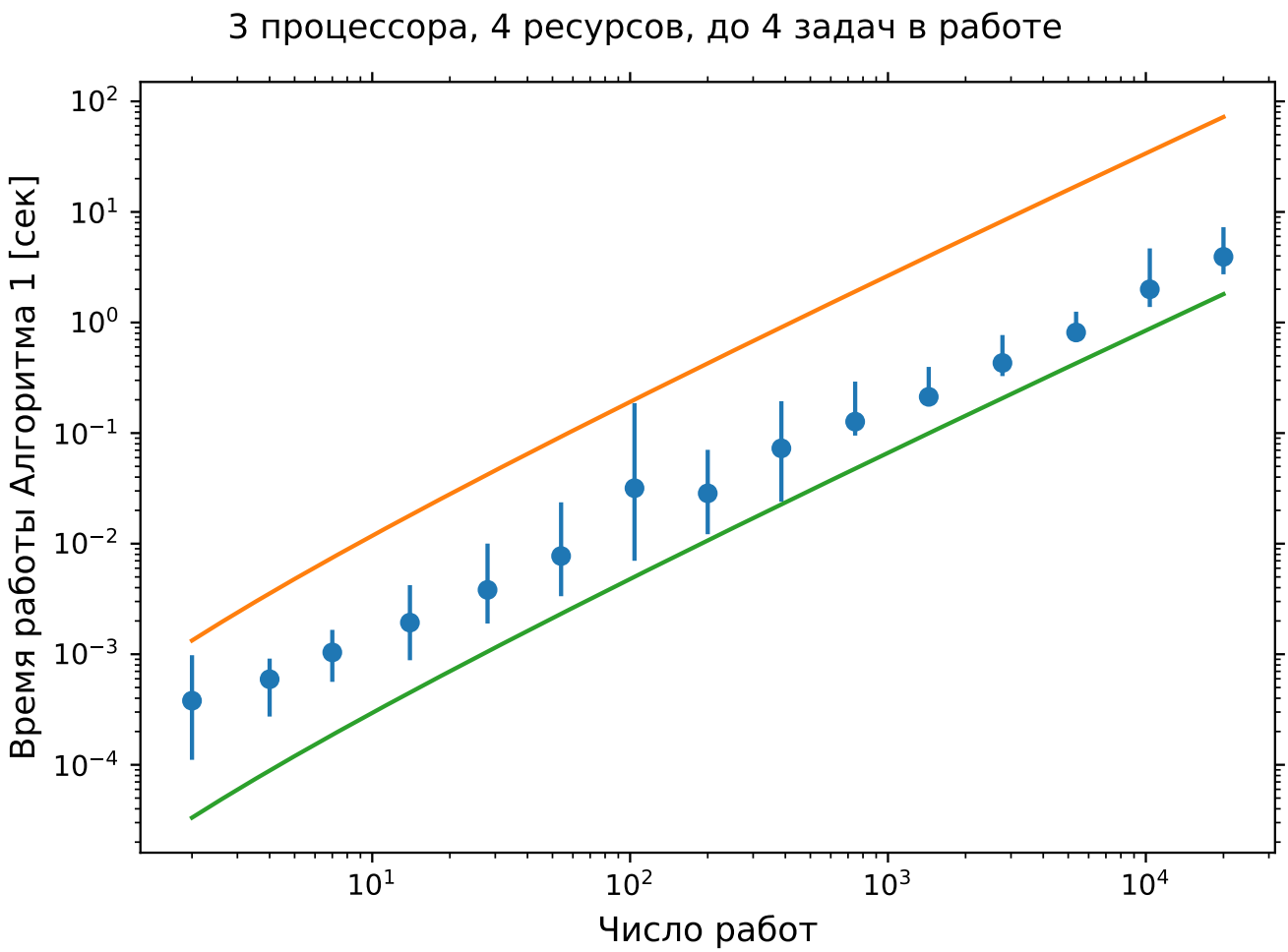


Рисунок 4.11 — Времена работы алгоритма двухуровневой диспетчеризации для различного количества работ при количестве приборов равном 3, количестве ресурсов и максимальном числе задач в работе равном 4. Аппроксимированы сверху и снизу функцией $C\,nS\log nS$, $S = 4$, $C_u = 8 \times 10^{-5}$ для ограничения сверху и $C_l = 2 \times 10^{-6}$ для ограничения снизу.

Таблица 14 — Результаты моделирования критериев оптимальности расписания для различных правил определения приоритета. Здесь АДД — алгоритм двухуровненевой диспетчеризации

Критерий оптимальности				C_a	$C_{\max}$	C_{sum}
Число проц.	Число ресурсов	Число работ	Алгоритм			
3	4	4	АДД	90	38	213
			FIFO	133	49	272

Продолжение на следующей странице

Критерий оптимальности				C_a	$C_{\max}$	C_{sum}
Число проц.	Число ресурсов	Число работ	Алгоритм			
			LIFO	131	49	269
			SPT	119	49	261
			LPT	143	50	283
			STPT	112	48	262
			LTPT	145	49	271
		12	АДД	572	105	1592
			FIFO	870	129	1931
			LIFO	859	129	1923
			SPT	744	128	1799
			LPT	994	130	2052
			STPT	697	126	1870
			LTPT	1035	130	1962
		42	АДД	5751	352	17916
			FIFO	9059	415	21463
			LIFO	9029	414	21419
			SPT	7473	412	19617
			LPT	10579	415	23270
			STPT	6874	414	20847
			LTPT	11237	417	22205
		260	АДД	205687	2177	675316
			FIFO	331620	2530	821988
			LIFO	331697	2530	822268
			SPT	270887	2534	752721
			LPT	392201	2531	889642
			STPT	241160	2523	787625
			LTPT	418759	2530	847709
		1612	АДД	7829404	13450	25949763
			FIFO	12563916	15581	31422194
			LIFO	12576138	15580	31453225

Продолжение на следующей странице

Критерий оптимальности				C_a	$C_{\max}$	C_{sum}
Число проц.	Число ресурсов	Число работ	Алгоритм			
			SPT	10296246	15580	28842258
			LPT	14843675	15586	34025446
			STPT	9138213	15523	30223354
			LTPT	15922792	15544	32447155
		10000	АДД	300622615	83372	996317229
			FIFO	482817412	96563	1206897240
			LIFO	482629378	96535	1206430089
			SPT	395465787	96481	1107852390
			LPT	569141680	96430	1302826118
			STPT	350706434	96178	1160449682
			LTPT	611479664	96203	1244159775
7	15	4	АДД	283	119	1874
			FIFO	446	167	2503
			LIFO	455	173	2491
			SPT	418	174	2467
			LPT	498	171	2556
			STPT	397	172	2526
			LTPT	521	175	2541
		12	АДД	1660	307	14310
			FIFO	2931	428	18370
			LIFO	2950	431	18472
			SPT	2534	433	17803
			LPT	3360	432	19433
			STPT	2310	426	18081
			LTPT	3567	431	18645
		42	АДД	15708	994	162801
			FIFO	29295	1339	210940
			LIFO	29111	1325	208755
			SPT	23938	1332	201142

Продолжение на следующей странице

Критерий оптимальности				C_a	$C_{\max}$	C_{sum}
Число проц.	Число ресурсов	Число работ	Алгоритм			
			LPT	34756	1326	222296
			STPT	21160	1335	206895
			LTPT	37441	1338	214679
		260	АДД	542359	6030	6167742
			FIFO	1034204	7898	8150049
			LIFO	1035142	7897	8160754
			SPT	824380	7944	7746419
			LPT	1253907	7912	8658066
			STPT	714301	7882	8009061
			LTPT	1349771	7867	8289077
		1612	АДД	20279288	37003	233280980
			FIFO	38821543	48153	310093671
			LIFO	38841199	48162	310239972
			SPT	30910196	48552	295151237
			LPT	47446054	48507	329813502
			STPT	26538832	48169	304376065
			LTPT	51211577	48168	315587501
		10000	АДД	775548405	228665	8940309607
			FIFO	1486658580	297356	11887525368
			LIFO	1487160670	297548	11891492888
			SPT	1181908024	299725	11324646193
			LPT	1817164073	299838	12655678412
			STPT	1014868738	297539	11684283565
			LTPT	1961091957	297553	12111351298
5	5	4	АДД	176	77	369
			FIFO	234	91	488
			LIFO	234	91	498
			SPT	206	90	469
			LPT	267	92	527

Продолжение на следующей странице

Критерий оптимальности				C_a	$C_{\max}$	C_{sum}
Число проц.	Число ресурсов	Число работ	Алгоритм			
			STPT	211	92	495
			LTPT	262	93	505
		12	АДД	1135	223	2687
			FIFO	1529	240	3407
			LIFO	1560	246	3494
			SPT	1314	248	3207
			LPT	1811	240	3706
			STPT	1295	243	3321
			LTPT	1811	242	3570
		42	АДД	10758	741	27902
			FIFO	14429	753	31734
			LIFO	14436	756	31598
			SPT	11749	773	29755
			LPT	18228	751	35392
			STPT	11798	777	31109
			LTPT	17954	757	34222
		260	АДД	384220	4637	1045517
			FIFO	515252	4650	1114362
			LIFO	515509	4652	1115623
			SPT	405612	4751	1049797
			LPT	659158	4647	1215299
			STPT	400813	4727	1081194
			LTPT	639885	4649	1165038
		1612	АДД	14673495	28946	40274510
			FIFO	19546933	28962	42231226
			LIFO	19550371	28961	42250270
			SPT	15118408	29248	39641958
			LPT	25033312	28953	45387830
			STPT	14928353	29141	40750223

Продолжение на следующей странице

Критерий оптимальности				C_a	$C_{\max}$	C_{sum}
Число проц.	Число ресурсов	Число работ	Алгоритм			
			LTPT	24421335	28962	44094149
		10000	АДД	562818664	179777	1546876440
			FIFO	749667308	179793	1617903168
			LIFO	749687287	179793	1617971580
			SPT	571566358	180393	1505542110
			LPT	961412828	179785	1740224606
			STPT	569410794	180644	1557939955
			LTPT	938183026	179799	1689644801
15	6	4	АДД	207	90	496
			FIFO	234	92	540
			LIFO	240	94	547
			SPT	221	95	539
			LPT	264	93	573
			STPT	219	94	546
			LTPT	267	96	567
		12	АДД	1209	233	3339
			FIFO	1485	239	3629
			LIFO	1463	238	3581
			SPT	1279	239	3450
			LPT	1767	240	3881
			STPT	1274	240	3564
			LTPT	1723	238	3698
		42	АДД	12132	798	36984
			FIFO	15654	803	38710
			LIFO	15692	805	38842
			SPT	12877	807	36746
			LPT	19315	802	41784
			STPT	12416	805	37789
			LTPT	19250	799	40327

Продолжение на следующей странице

Критерий оптимальности				C_a	$C_{\max}$	C_{sum}
Число проц.	Число ресурсов	Число работ	Алгоритм			
		260	АДД	402870	4648	1323871
			FIFO	544622	4653	1378197
			LIFO	544475	4654	1378249
			SPT	428250	4667	1291765
			LPT	680411	4650	1475777
			STPT	408018	4676	1334319
			LTPT	689571	4654	1432598
		1612	АДД	15062457	28509	51227772
			FIFO	20756267	28515	53209108
			LIFO	20756042	28515	53209343
			SPT	16063340	28560	49727514
			LPT	25933707	28516	56792522
			STPT	15155809	28591	51388387
			LTPT	26386717	28515	55140132
		10000	АДД	573786958	175582	1966533935
			FIFO	791781682	175587	2038678055
			LIFO	791782529	175586	2038698258
			SPT	612239245	175764	1906514597
			LPT	991830161	175586	2177601000
			STPT	574620399	175706	1968002707
			LTPT	1011049884	175585	2114513950

4.2.4 Программная реализация системы имитационного моделирования

Опишем программную реализацию имитационной системы. Программа написана на ЯП Python 3 для работы в дистрибутивах системы Linux. Она служит для имитации поведения СА, описанной математической моделью 2.1, и позволяет проведение вычислительных экспериментов для исследования качеств составления расписаний в таких системах при различных значениях параметров модели. Работа программы организована следующим образом. Запуск программы производится из командной строки командой `python3 main.py` из корневой директории программы. Входные данные задаются в командной строке после

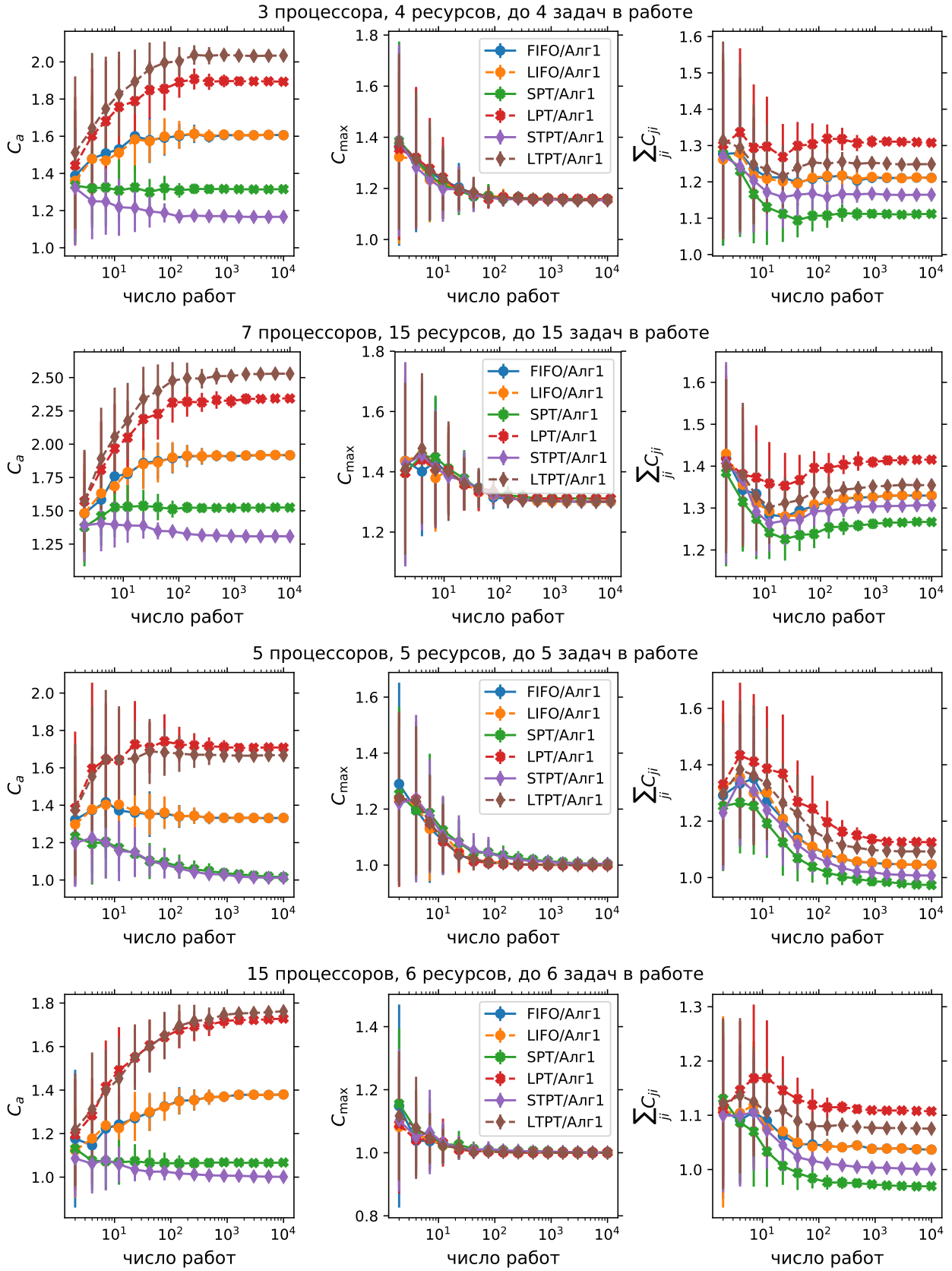


Рисунок 4.12 — Отношения критериев оптимальности для различных правил диспетчеризации из класса приоритето-порождающих функционалов 1-го порядка к критериям оптимальности для алгоритма двухуровневой диспетчеризации, для различного количества работ при разных параметрах модели.

имени программы способом, описанным в части “Эксплуатация программы” данного параграфа. Далее, согласно заданным параметрам, генерируются заявки по алгоритму, описанному в части 4.2.3 данной главы, рисунок 4.10. Моделируется обслуживание заявок выбранными пользователями алгоритмами.

Результатами выполнения программы являются файлы `time_<run-id>.csv` с временами работы алгоритмов в виде таблицы, `criteria_<run-id>.csv` с критериями оптимальности для рассматриваемых алгоритмов в виде таблицы, файлы регистрации событий в подпапке `logs`. Также, в зависимости от выбранного режима, программа может записывать составленные расписания в формате `.json` либо создавать изображения этих расписаний в виде диаграмм Ганта.

Блок-схема моделирующего алгоритма изображена на рисунке 4.9 в части 4.2.1. Структура взаимодействия модулей представлена на рисунке 4.13.

Общие сведения о программе. Программа называется «Компьютерная система для исследования проблем составления расписаний в распределенных системах с ограниченными ресурсами». Программа реализована на языке программирования Python3 в виде консольного приложения для запуска в UNIX подобных операционных системах. Требования к оперативной памяти: 200 МБ. Объем исходного текста программы составляет 130,152 байта.

Функциональное назначение. Данное программное обеспечение (ПО) предназначено для создания имитационных моделей систем распределённой обработки информации с ограничениями по одновременному доступу и объему ресурсов, необходимых для обработки работ, и проведения комплексного статистического исследования особенностей составления расписаний в таких системах.

Структура программы.

Файловая структура программного комплекса изображена на рисунке 4.14

Требования к программному окружению. Требования к программному окружению. Операционная система Linux или MacOS 10.0.4 и новее. Интерпретатор языка Python 3.8 или новее, библиотеки statistics, pandas, logging, math, numpy, time, weakref, json copy, plotly, locale, random, argparse, csv, matplotlib.

Эксплуатация программы. Запуск программы осуществляется из командной строки следующим образом: `python3 main.py <опции запуска>`, перечень допустимых опций приведён в таблице 15.

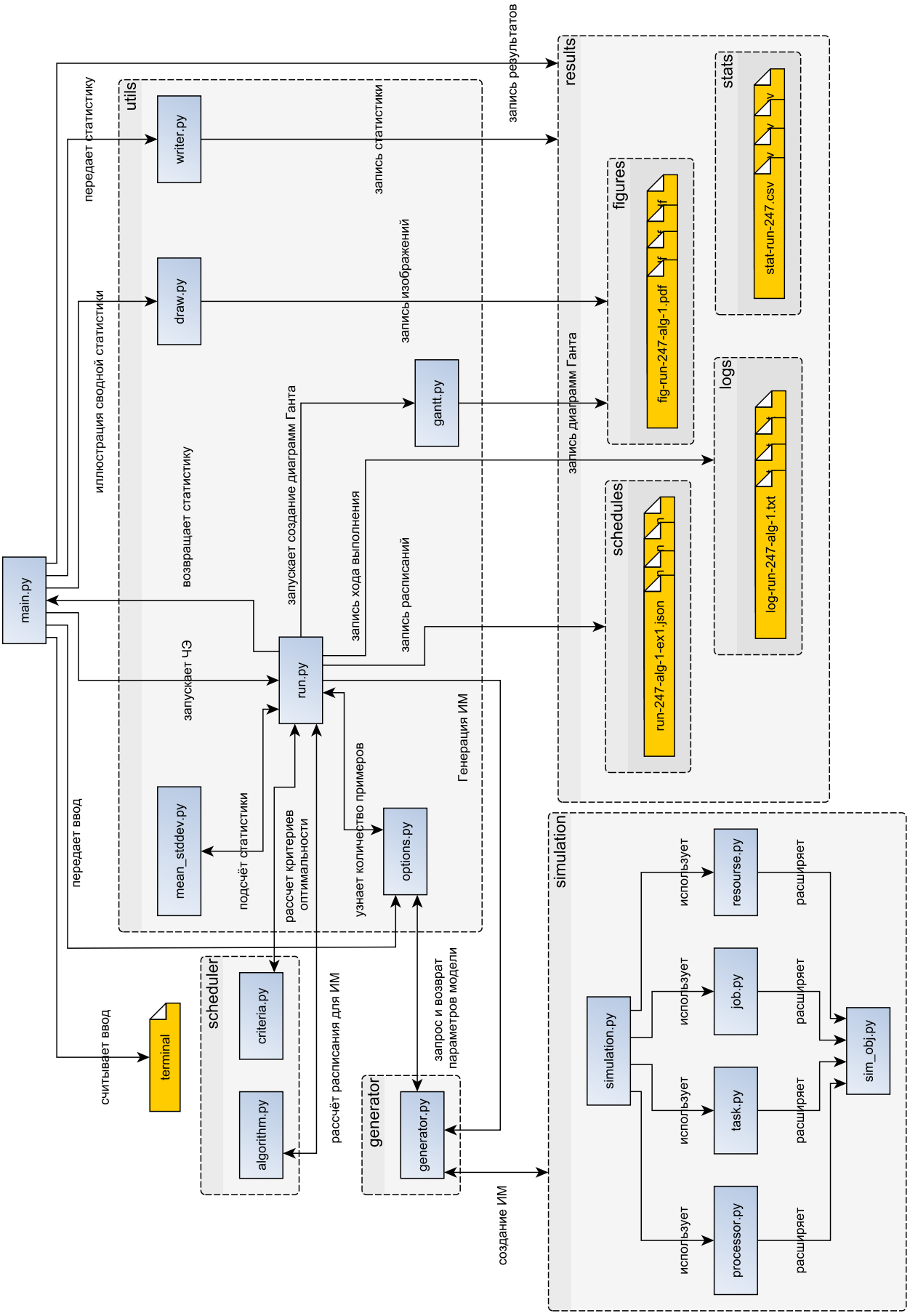


Рисунок 4.13 — Диаграмма потока данных при взаимодействии модулей системы имитационного моделирования.

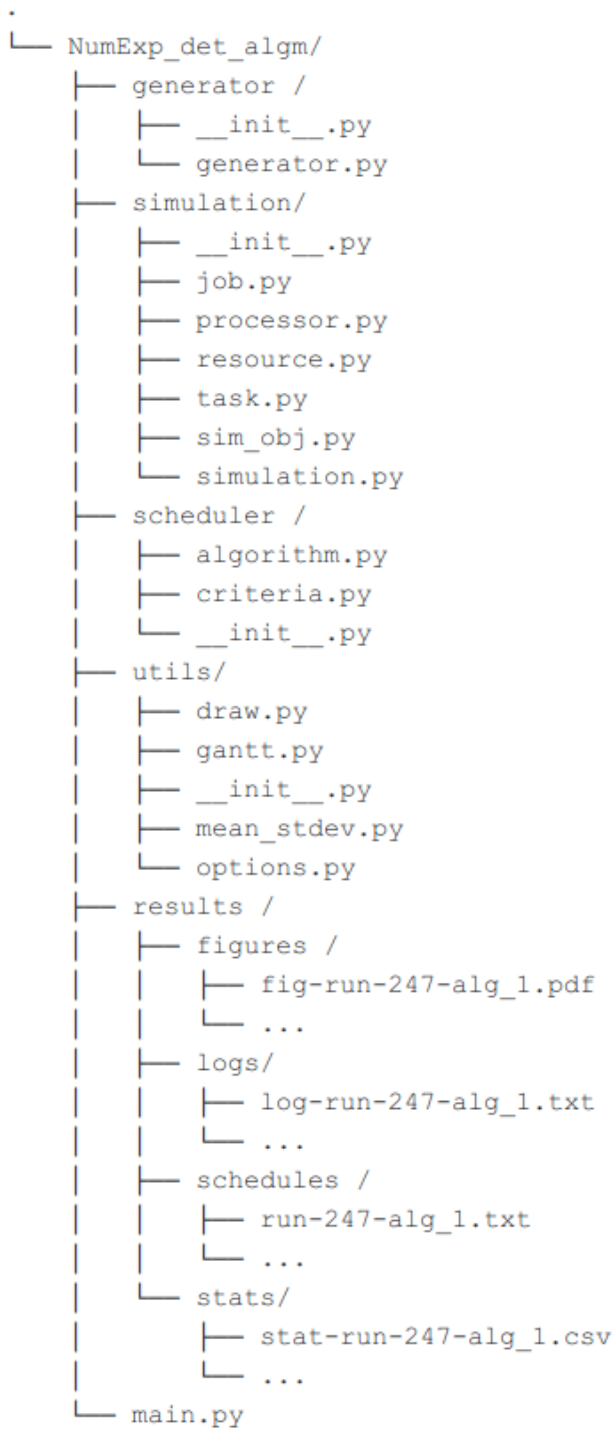


Рисунок 4.14 — Файловая структура СИМ.

Таблица 15 — Допустимые опции запуска программного обеспечения

Параметр	**Опция**	**Короткая опция**	**Возможные значения**
Количество приборов	`--processors`	`-p`	$\in \mathbf{N}$
$(\mu_1, \ldots, \mu_S)$	`--speed`	`-s`	$\in \mathbf{R}^+$
Max количество задач в одной работе	`--tasks`	`-k`	$\in [1, \ldots, S]$, по умолчанию равно S
Количество запусков для усреднения	`--repeat`	`-r`	$\geqq 1$, по умолчанию равно 1
Количество работ	`--jobs`	`-j`	$\geqq 1$
Подробные логи	`--verbose`	`-v`	по умолчанию отключены
Средний объём задачи (параметр распределения Пуассона)	`--lambda`	`-l`	> 0, по умолчанию равен 10

Пример запуска программы приведён на рисунке 4.15. На рисунке 4.16, 4.17 показаны примеры файлов сводной статистики, сгенерированного по результатам запуска программы.

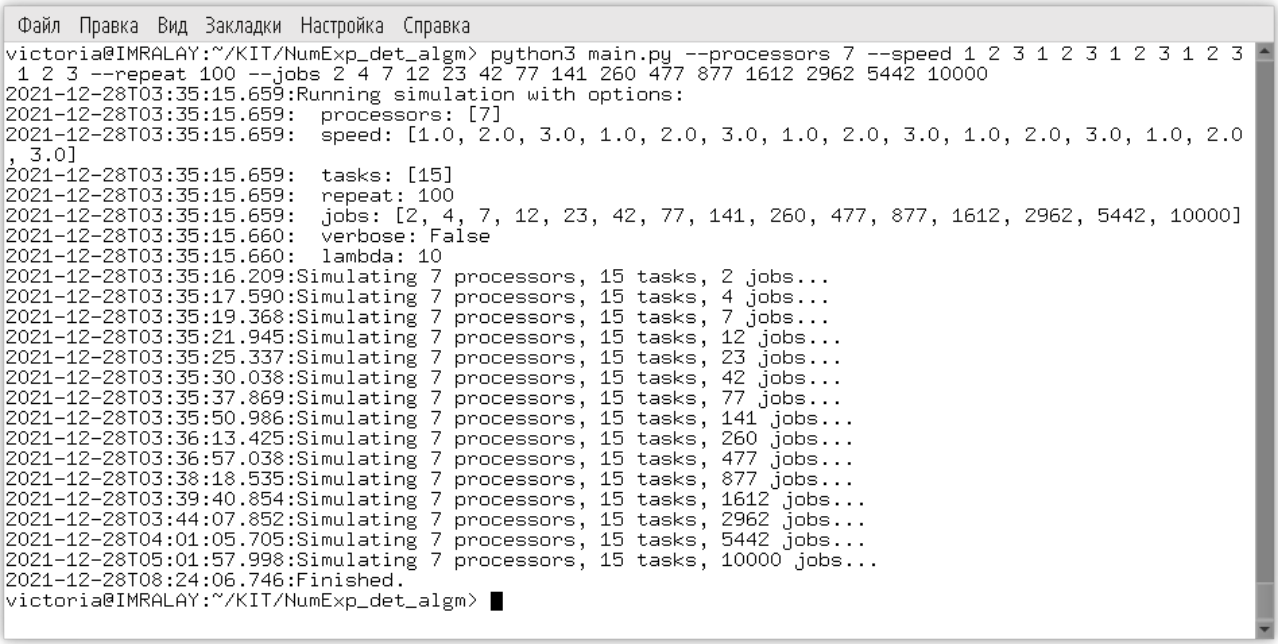
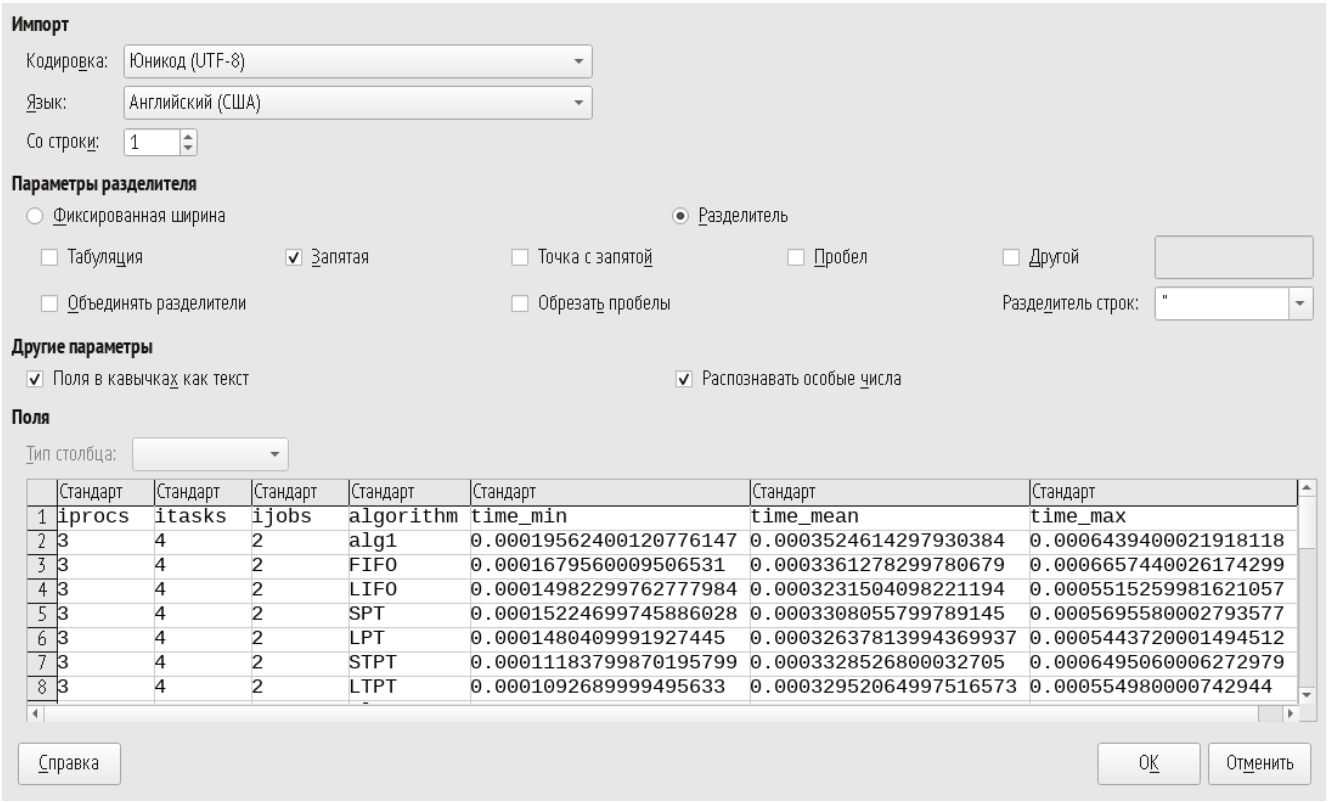


Рисунок 4.15 — Запуск СИМ в командной строке.

Свидетельство о регистрации программы представлено в приложении Б.3.



	Стандарт	Стандарт	Стандарт	Стандарт	Стандарт	Стандарт	Стандарт
1	iprocs	itasks	ijobs	algorithm	time_min	time_mean	time_max
2	3	4	2	alg1	0.0001956240012077614 7	0.0003524614297930384	0.0006439400021918118
3	3	4	2	FIFO	0.0001679560009506531	0.0003361278299780679	0.0006657440026174299
4	3	4	2	LIFO	0.0001498229976277798 4	0.0003231504098221194	0.0005515259981621057
5	3	4	2	SPT	0.0001522469974588602 8	0.0003308055799789145	0.0005695580002793577
6	3	4	2	LPT	0.0001480409991927445	0.0003263781399436993 7	0.0005443720001494512
7	3	4	2	STPT	0.0001118379987019579 9	0.0003328526800032705	0.0006495060006272979
8	3	4	2	LTPT	0.0001092689999495633	0.0003295206499751657 3	0.000554980000742944

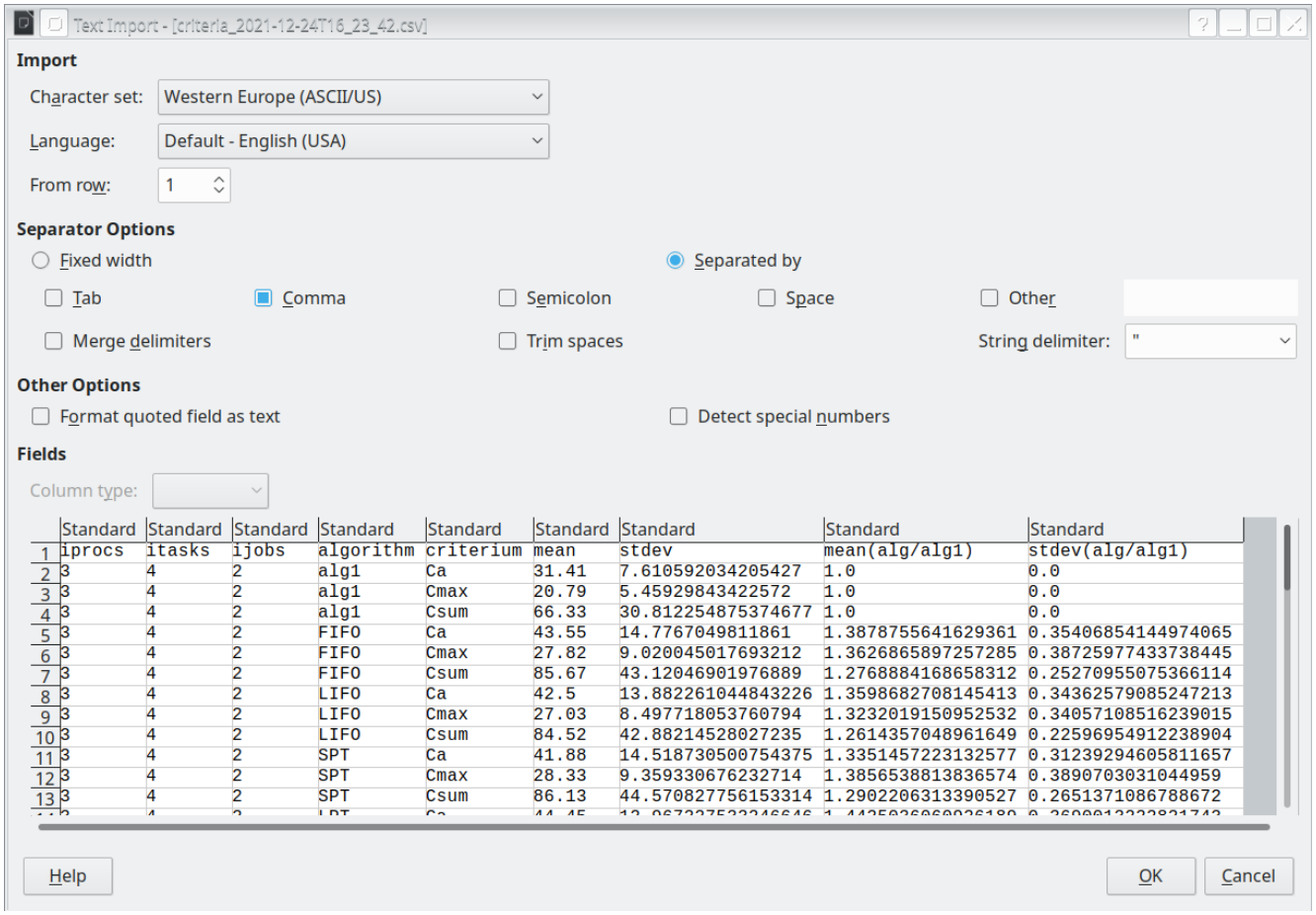
Рисунок 4.16 — Открытие .csv файла со статистикой по временам работы алгоритмов в программе Libre Office.

4.3 Реализация комплекса программ для агрегированной обработки распределенных смешанных данных в центре анализа и обработки данных инициативы GRADLCI

В данном параграфе приводится описание комплекса программ, написанных на ЯП Python (с использованием ЯП html, bash, yml), созданного автором в процессе работы в международном исследовательском проекте German-Russian Data Life Cycle (GRADLCI)1 [147].

Алгоритмы и программы в описываемом программном комплексе реализованы автором лично. Непосредственно автору принадлежат дизайн и создание пользовательских интерфейсов (GUI и API), разработка и реализация алгоритмов обработки агрегированных запросов и генерации расписаний их обработки, что является темой данного диссертационного исследования. Для операций, связанных со специальной обработкой данных экспериментов астрофизики частиц, были использованы сторонние библиотеки, в частности, написанные соавторами автора по проекту, специализирующимися в области обработки и анализа данных

[1]Работа была выполнена при поддержке грантов РНФ №18-41-06003 и фонда Гермгольца №HRSF-0027.



	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard
1	iprocs	itasks	ijobs	algorithm	criterium	mean	stdev	mean(alg/alg1)	stdev(alg/alg1)
2	3	4	2	alg1	Ca	31.41	7.610592034205427	1.0	0.0
3	3	4	2	alg1	Cmax	20.79	5.45929843422572	1.0	0.0
4	3	4	2	alg1	Csum	66.33	30.812254875374677	1.0	0.0
5	3	4	2	FIFO	Ca	43.55	14.7767049811861	1.3878755641629361	0.35406854144974065
6	3	4	2	FIFO	Cmax	27.82	9.020045017693212	1.3626865897257285	0.38725977433738445
7	3	4	2	FIFO	Csum	85.67	43.12046901976889	1.2768884168658312	0.25270955075366114
8	3	4	2	LIFO	Ca	42.5	13.882261044843226	1.3598682708145413	0.34362579085247213
9	3	4	2	LIFO	Cmax	27.03	8.497718053760794	1.3232019150952532	0.34057108516239015
10	3	4	2	LIFO	Csum	84.52	42.88214528027235	1.2614357048961649	0.22596954912238904
11	3	4	2	SPT	Ca	41.88	14.518730500754375	1.3351457223132577	0.31239294605811657
12	3	4	2	SPT	Cmax	28.33	9.359330676232714	1.3856538813836574	0.3890703031044959
13	3	4	2	SPT	Csum	86.13	44.570827756153314	1.2902206313390527	0.2651371086788672

Рисунок 4.17 — Открытие .csv файла с подробной статистикой по критериям оптимальности в программе Libre Office.

экспериментальной астрофизики частиц. Автор участвовал в проекте как прикладной математик и разработчик программного обеспечения.

Общие сведения о программе. Программа называется «Многопроцессорное клиент-серверное приложение с механизмом адаптивного составления расписаний для обработки информационных запросов в распределенных хранилищах данных».

Она представляет собой кроссплаформенное программное обеспечение, работающее на всех распространённых типах ОС, устанавливаемых на стационарные компьютеры, таких как Linux, MacOS X, Windows. Работа программы осуществляется в браузере и тестировалась в браузерах Google Chrome версии 94.0.4606.81 и Mozilla Firefox версии 93.0.

Объем программного комплекса составляет 957.1 КБайт.

Структура программы. Программа имеет следующую структуру, показанную на рисунке4.20. В общей сложности комплекс состоит из 30 файлов с кодом, написанных на ЯП Python 3, 13 html файлов (в директории templates), а также

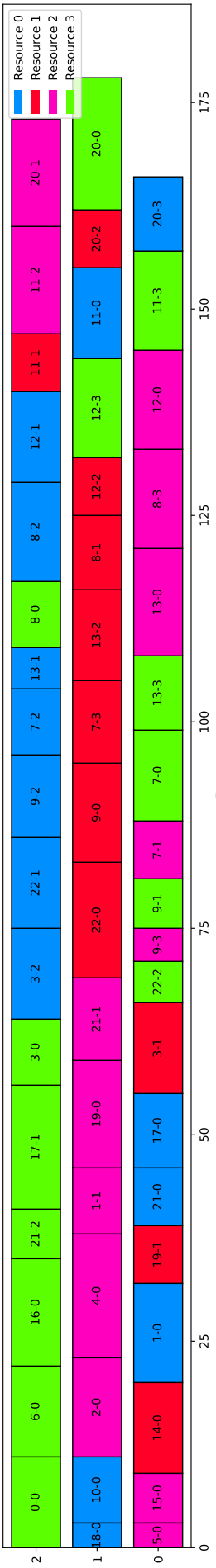


Рисунок 4.18 — Пример диаграммы Ганта, полученной программой для случая решения примера с параметрами $n = 23, m = 3, S = 4$ АДП.

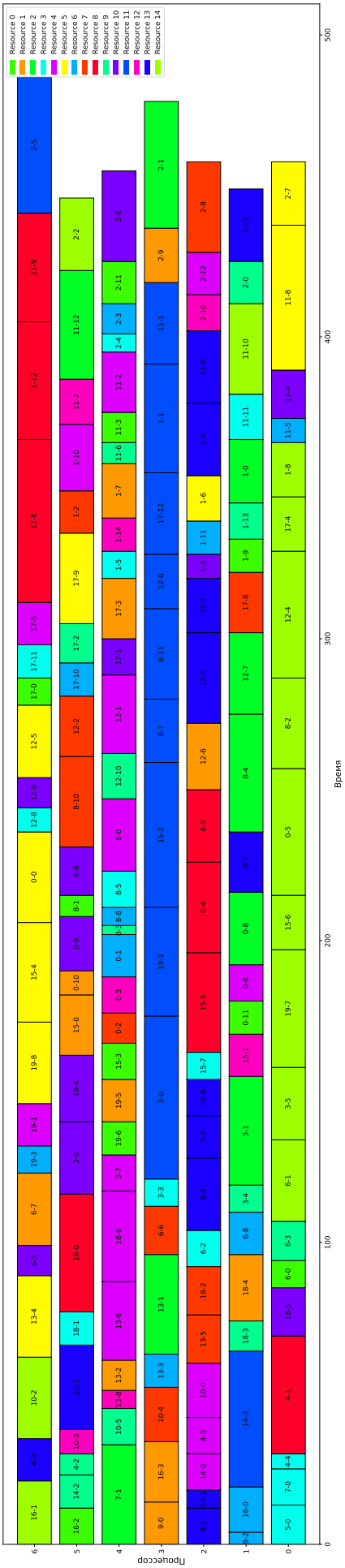


Рисунок 4.19 — Пример диаграммы Ганта, полученной программой для случая решения примера с параметрами $n = 20, m = 7, S = 15$ АДП.

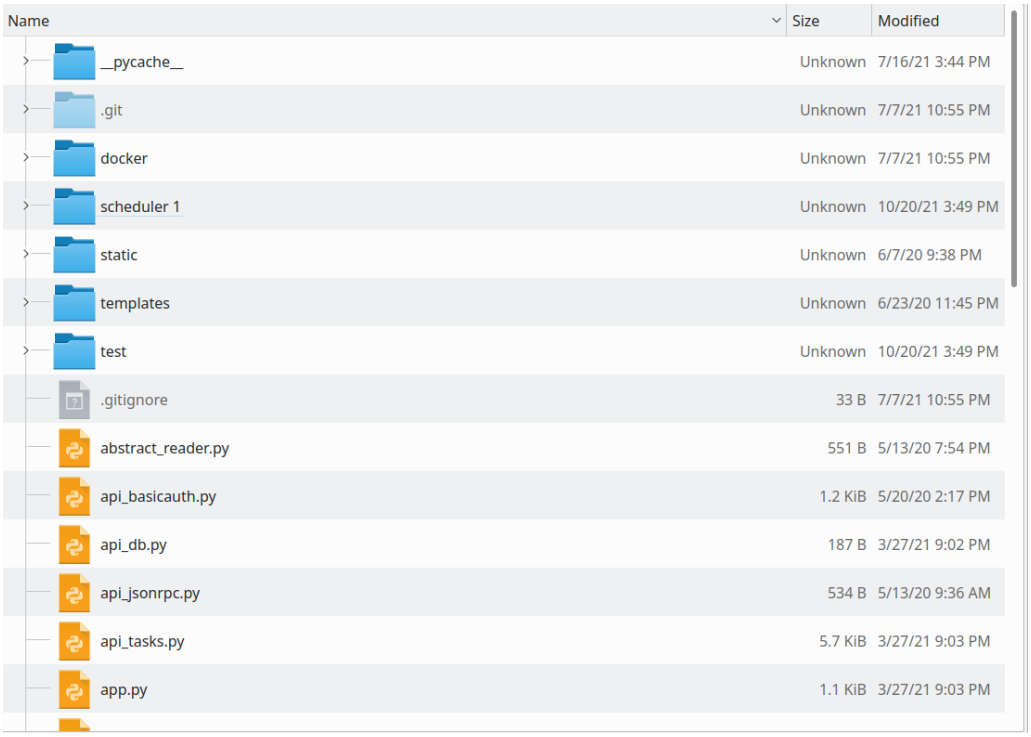


Рисунок 4.20 — Файловая структура программного комплекса.

вспомогательных скриптов в форматах .yml,.bash,.git,.txt,.sql, расположенных в как в основной директории программы, так и в директориях .git, static и docker. Директории __pycache__ и .git также содержат большое количество промежуточных технических файлов, генерируемых в процессе работы, поддержки и модификации программного комплекса.

Описание работы программы. Ключевыми объектами, используемыми в программном комплексе, являются:

– *Сервер агрегации.* Система массового обслуживания, принимающая пользовательские запросы и управляющая из обслуживанием.
– *GUI* - графический программный интерфейс, через который пользователи взаимодействуют с системой.
– *API* - прикладной программный интерфейс, через который пользователи взаимодействуют с системой.
– *Удаленные хранилища данных.* Дата-центры различных проектов в обрасти астрофизики частиц. Ключевой концепцией системы GRADLCI, является идея о простоте горизонтального масштабирования системы за счет подключений новых хранилищ. Число хранилищ, одновременно подключенных к агрегатору будет определяться целым натуральным числом, меньшим, чем $+\infty$, но достаточно большим. Хранилища данных и сами данные характеризуются существенной неоднородностью.

– *База метаданных (MDDB)*. Использование базы метаданных является инженерным решением, позволяющим ускорить получение данных из удалённых хранилищ. Более подробно использование базы метаданных разобрано далее в этом разделе.
– *Адаптеры*. Промежуточный API на стороне удалённых хранилищ, с которым взаимодействует сервер агрегации при отправке запросов.
– *Экстракторы*. Программные объекты, ответственные за автоматическое обновление базы метаданных в плановом режиме.
– *Модуль составления расписаний, Планировщик (Scheduler)* - использует алгоритмы, разработанные в главе 3, для составления и динамического изменения расписаний обработки запросов.
– Запрос - некоторое требование (возможно, включающее в себя взаимозависимые подзадачи), отправляемое пользователем на сервер агрегации посредством одного из программных интерфейсов.

Также важными внешними элементами по отношению к системе являются пользователи и экспериментальные установки, считывающие новые данные и предоставляющие их в хранилища данных. Общая смеха взаимодействия перечисленных объектов показана на рисунке 4.21.

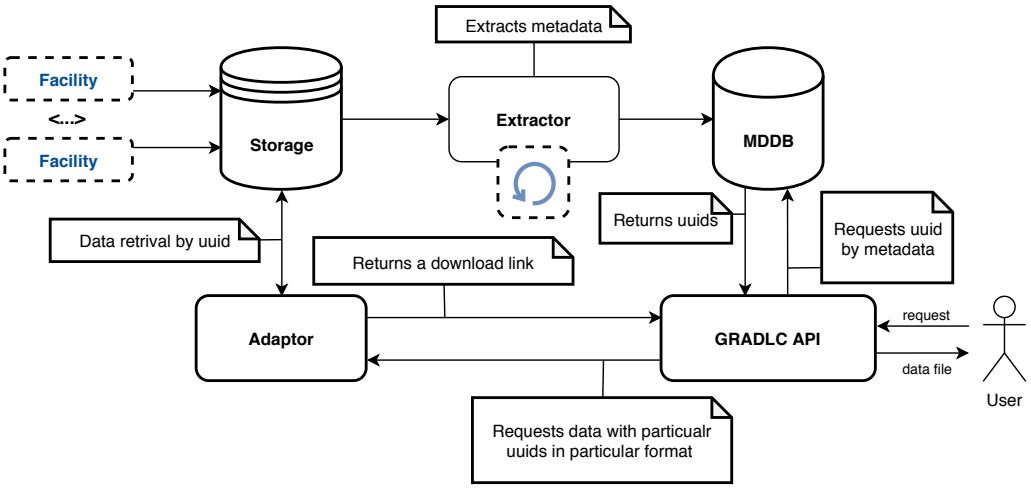


Рисунок 4.21 — Взаимодействие ключевых компонентов программного комплекса.

К системе подключены следующие хранилища данных KASCADE Comic Ray Data Center (KCDC) [148], Tunka-133 [149] и Tunka-Rex Virtual Observatory (TrVo) [150], предоставляющие доступ к данным таких экспериментов, как: KASCADE, KASCADE-GRANDE [151], COMBINED (объединённый набор данных, составленный на основе данных экспериментов KASCADE и KASCADE-

GRANDE) [152], LOPES [153], Tunka-133 [149], Tunka-Rex [154], Maket-Ani [155], а также наборы симуляций для различных установок.

Данных приведённых экспериментов имеют различный формат и структуру, в частности, для них характерны различные наборы метаданных.

Под метаданными будем понимать такие сведения о данных, как обозначения из атрибутов, величины измерения их атрибутов, области определения значений атрибутов, а также их значение, как ключей в базе метаданных [156]. Метаданные для экспериментальных данных и данных симуляций приведены в таблице 16 и таблице 17, соответственно. Схема базы метаданных приведена на рисунке 4.22.

Таблица 16 — Наименования аспектов данных различных экспериментальных установок, по которым возможна сортировка

Экспериментальная установка	Наименования аспектов данных экспериментальной установки, по которым возможна сортировка
KASCADE	Datetime, Energy, Electron number, Muon number, Zenith, Azimuth, Core distance, Shower age
GRANDE	Datetime, Zenith, Azimuth, Number of charged particles, Muon number, Core distance, Shower age
COMBINED	Datetime, Energy, Zenith, Azimuth, Core distance, Electron number, Muon number, Shower age
LOPES	LopesCompID, Datetime, Azimuth EW, Azimuth NS, CCheight EW, CCheight NS, ConeAngle EW, ConeAngle NS, EfieldMaxAbs, Eps EW, Eps NS, Elevation EW, Elevation NS, Eta EW, Eta NS, Geomagnetic Angle, Geomagnetic AngleG, NCCbeamAntennas EW, NCCbeamAntennas NS, Reconstruction, RmsCCbeam EW, RmsCCbeam NS, Xheight EW, Xheight NS
Tunka-133	Datetime, Energy, Zenith, Azimuth, Core distance, X_max
Tunka-Rex	Datetime, station_id

Функциональное назначение программы состоит в сборе и обработке пользовательских запросов к удалённым хранилищам. Расписания обработки запросов строятся модулем составления расписаний, реализующем алгоритмы, разработанные в главе 3.

Таблица 17 — Наименования и принимаемые значения аспектов данных имитационного моделирования (симуляций), по которым возможна сортировка

Параметр	Возможные значения
Datasets	KASCADE, KASCADE-GRANDE, COMBINED
Particles	«gamma», «proton», «helium», «carbon», «silicon», «iron»
Models	QGSjet-II-02, QGSjet-II-04, EPOS 1.99, EPOS LHC, SIBYLL 2.1, SIBYLL 2.3, SIBYLL 2.3c
Energy	1.0×10^{14} to 1.0×10^{18} eV, 5.62×10^{17} to 3.16×10^{18} eV

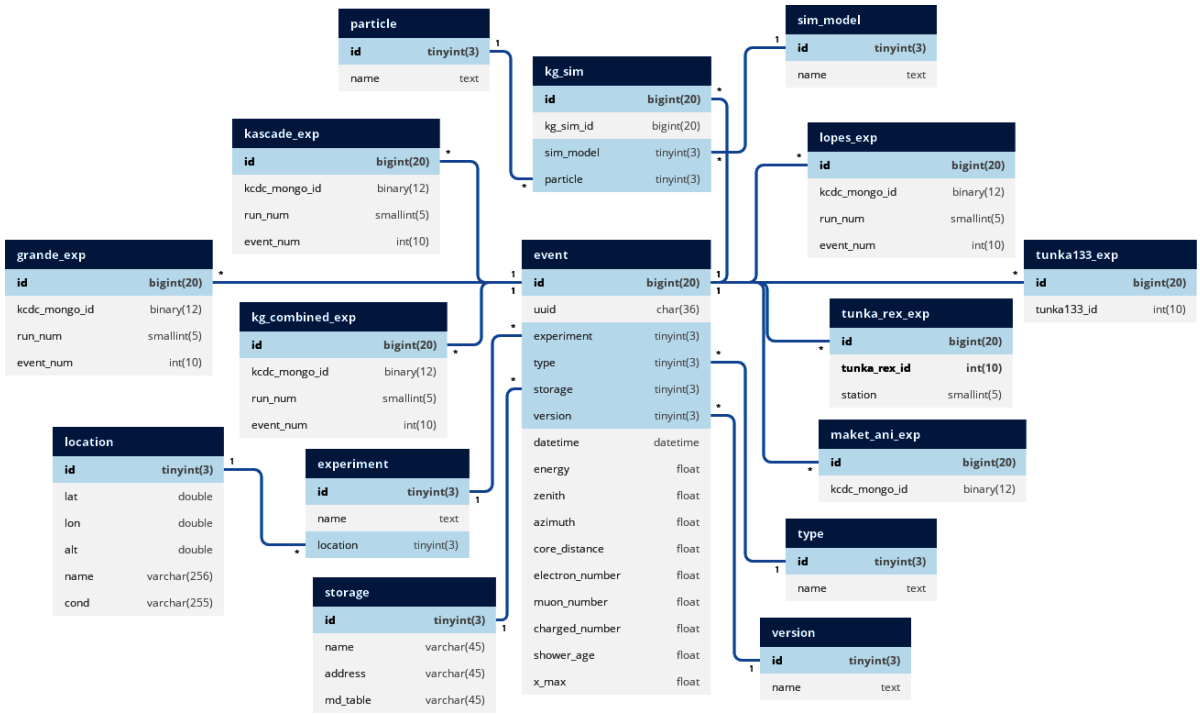


Рисунок 4.22 — Схема реляционной базы метаданных.

Требования к программному окружению. Операционная система Linux, MacOS X или Windows. Браузеры Google Chrome или Mozilla Firefox. Установленные библиотеки для языка Python 3.6 или выше, MySQL версии 8.0, Docker версии 18.0 и выше.

Эксплуатация программы.

Интерфейсы взаимодействия.

Взаимодействие пользователя с сервером агрегации осуществляется посредством следующих интерфейсов:

– Интерфейс прикладного программирования (англ. Application programming interface, API), представляющий собой совокупность команд, процедур, функ-

ций и параметров, позволяющих взаимодействовать с сервером агрегации посредством их использования в компьютерных программах;

– Графический интерфейс пользователя (англ. Graphical User Interface, GUI), представляющий собой набор визуальных элементов, таких как вкладки, кнопки, поля страницы, ссылки и т.д., посредством которых пользователь может взаимодействовать с сервером агрегации.

Интерфейс прикладного программирования.

Для обеспечения взаимодействия пользователя с агрегатором был реализован интерфейс прикладного программирования GRADLC API, использующий протокол удаленного вызова процедур JSON-RPC 2.0 [157] с передачей данных по http.

Через API агрегатора можно запросить пять методов: извлечение данных, статус запроса, список запросов, отмена запроса и загрузка данных. Примеры запросов и подробные объяснения можно найти в официальной документации [158].

Через два дня после выполнения запроса связанные с ним данные автоматически удаляются с сервера, а запрос получает статус «Просрочен». В системе существует шесть возможных статусов запроса: «Назначен», «Выполняется», «Закончен», «Не удалось», «Просрочен» и «Удален».

Графический интерфейс пользователя.

Является обёрткой над API, таким образом, имеет совпадающий набор команд и решаемых задач. Преимуществом графического интерфейса является его интуитивная простота и доступность для пользователей, не обладающих навыками программирования.

Рассмотрим примеры взаимодействия пользователя с графическим интерфейсом (рисунки 4.23–4.24).

Рисунок 4.23 показывает пример создания запроса на выгрузку данных. Для создания запроса в графическом интерфейсе в окне «Новая задача» следует зайти во вкладку с именем интересующего эксперимента, отметить его название галочкой внутри вкладки и указать желаемые значения параметров выборки. После того, параметры во всех необходимых вкладках отмечены, можно оправить запрос, нажав кнопку «Создать запрос». Для автоматизации выборки данных с API бывает полезно так же уметь генерировать корректный запрос в JSON-RPC формате из графического интерфейса. Воспользоваться такой возможностью можно, используя кнопку «Get JSON».

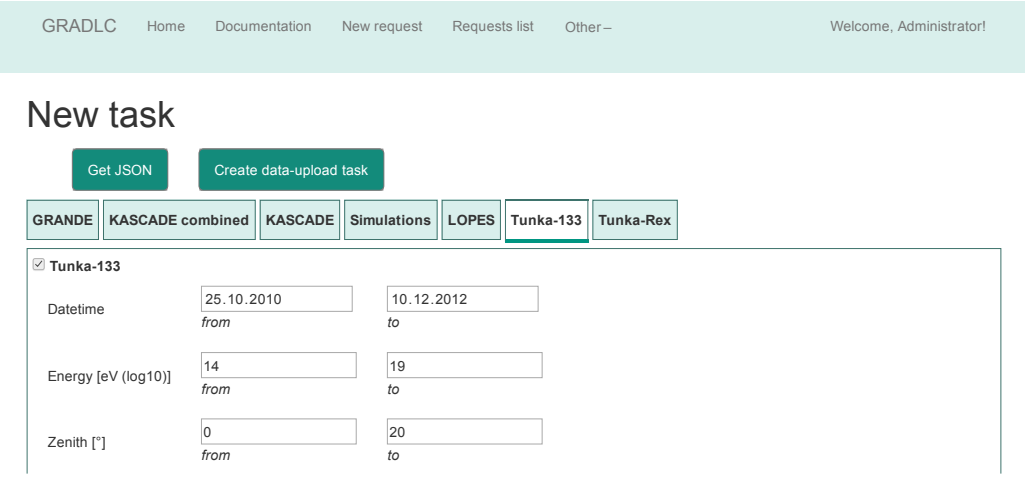


Рисунок 4.23 — Интерфейс создания нового запроса на выгрузку данных в веб-интерфейсе пользователя GRADCLI.

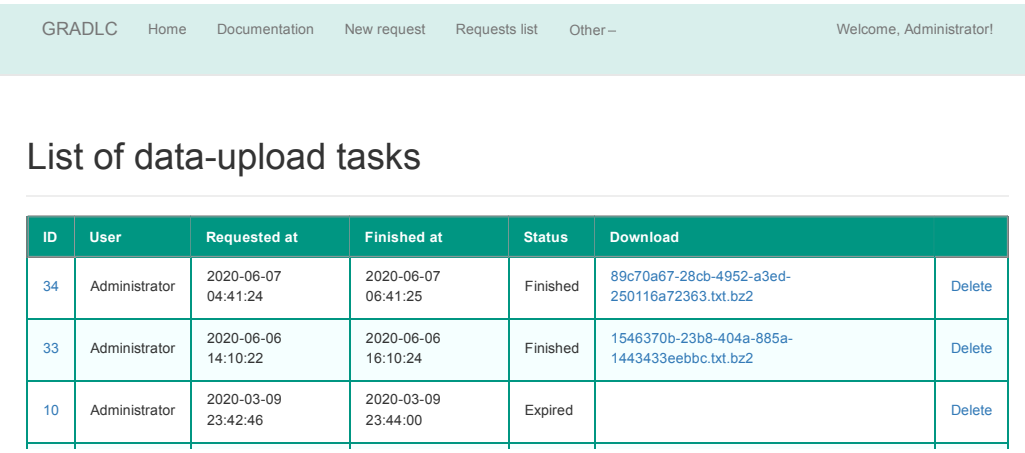


Рисунок 4.24 — Управление запросами в пользовательском интерфейсе программного комплекса GRADCLI.

Для управления своими запросами на выгрузку данных используется раздел «Список запросов» веб-интерфейса, который показан на рисунке 4.24. Здесь можно получить список всех запросов на выгрузку данных, которые были сделаны пользователем. В таблице можно увидеть все запросы, их статус, время начала и завершения. Запрос можно отменить, нажав на ссылку «Удалить» в интерфейсе. При нажатии на идентификатор задания отображается подробная информация о запрошенных данных.

Также веб-интерфейс предоставляет пользователю некоторые дополнительные возможности, такие как управление учетной записью в системе (т.е. регистрация, вход, изменение пароля, выход) и доступ к документации.

Обработка данных на стороне сервера.

Рассмотрим более подробно поведение системы при получении запроса на получение данных. На рисунке 4.25 пользователь отправляет запрос агрегатору,

используя GRADLCI API или GUI. На стороне агрегатора генерируется уникальный идентификатор запроса (UUID), который в дальнейшем используется системой для выполнения всех действий, связанных с его обработкой. Далее запрос добавляется в базу данных агрегатора со статусом запроса «Назначен», а идентификатор запроса возвращается пользователю для выполнения дальнейших действий, связанных с ним.

Демон планировщика, работающий на стороне сервера регистрирует добавление нового запроса в базу данных агрегатора, и запускает его обработку. Для этого, при наличии свободных вычислительных ресурсов на сервере, создается процесс для извлечения запрошенных данных из хранилищ, более подробно описанный на рисунке 4.26. Про составлении расписаний обработки запросов используются методы и алгоритмы, изложенные в главе 3. После того, как запущено выполнение запроса, его статус в системе изменяется на «Выполняется».

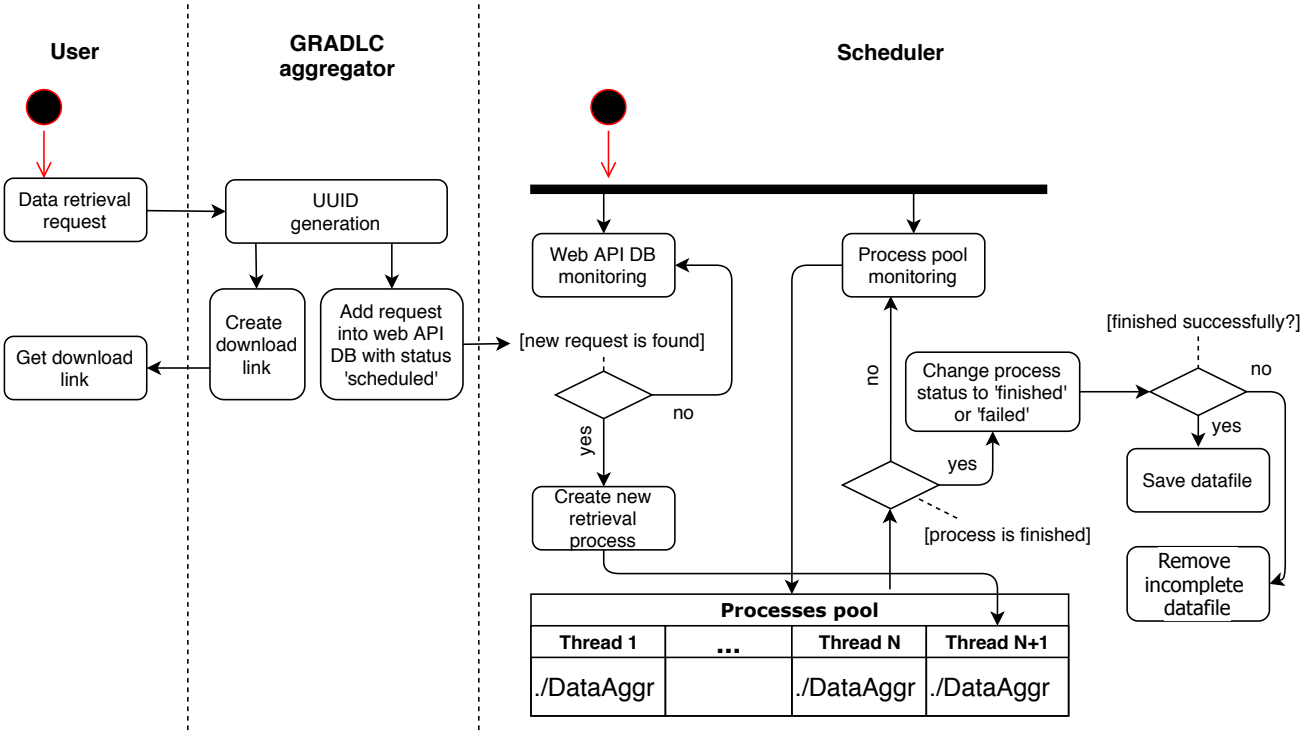


Рисунок 4.25 — Асинхронная параллельная обработка данных на стороне сервера.

Планировщик производит мониторинг запущенных процессов и по завершении процесса изменяет его статус запроса в системе на «Не удалось» или на «Закончен» в случае успешного завершения. Для успешно завершенных процессов результаты запросов записываются на сервере и архивируются для последующей выкачки пользователем. Для неудачно завершившихся процессов промежуточные файлы, созданные в процессе обработки, удаляются с сервера.

Взглянем более подробно на процесс извлечения данных по запросу пользователя, изображенный на рисунке 4.26.

Планировщик передает параметры запроса экземпляру процесса агрегатора, который выполняет запрос к MDDB и получает в качестве ответа список UUID, соответствующих указанным параметрам. Далее агрегатор обращается к адаптерам нужных хранилищ данных для быстрого извлечения необходимой информации.

Можно видеть, что новые хранилища могут быть включены в данную систему достаточно легко: для этого требуются подключение нового адаптера хранилища удалённых ресурсов, экстрактор для извлечения метаданных и индексация данных хранилища с UUID.

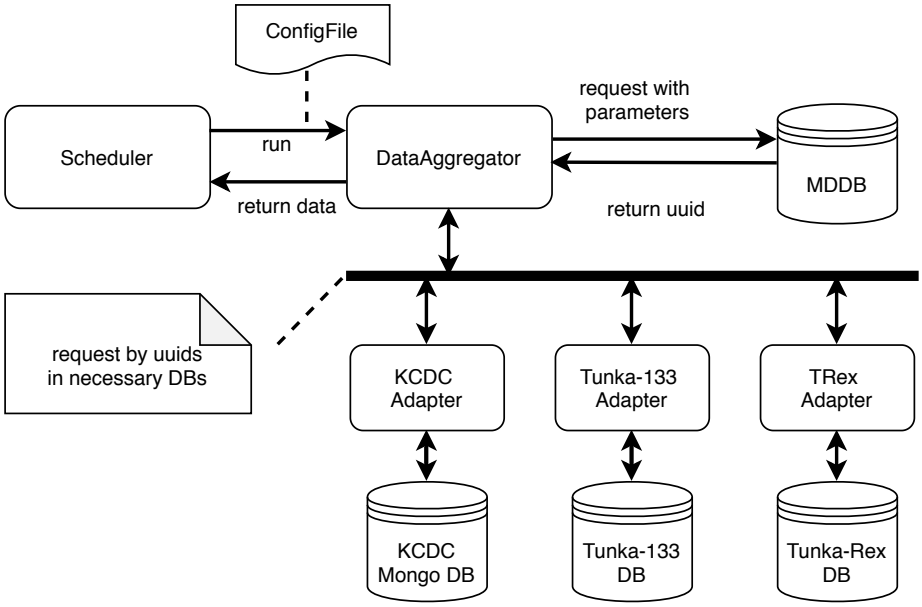


Рисунок 4.26 — Агрегирование данных из распределенных хранилищ.

Свидетельство о регистрации представлено в приложении Б.1.

4.4 Результаты и выводы по главе 4

1. В данной главе нами были произведены следующие численные исследования:
   1.1 Для случая выделенных приборов, $m \leq S$ и произвольных вещественных времён поступления работ r_j был произведен сравнительный анализ эффективности алгоритмов, основанных на приоритето-порождающих функционалах первого порядка. Исследование показало значимую зависимость эффективности алгоритмов от параметров модели и внешней среды. Были сформулированы рекомендации по использованию алгоритмов данной

группы для решения задач составления расписаний с ресурсами ограниченной доступности в информационно-вычислительных системах;

1.2 Была произведена верификация результатов анализа времени выполнения алгоритма 1 для среднего случая, изложенного части 3.2.2. Результаты показывают, что $T(n,S) = \Theta(nSlognS)$. Описана процедура нахождения числовых параметров функции времени выполнения $T(n,S)$ для различных состояний системы;

1.3 Сравнительный анализ эффективности построения расписаний по выбранным критериям оптимальности показал, что эвриистический алгоритм двухуровневых перестановок, разработанный автором данного исследования, превосходит алгоритмы, основанные на функционалах 1-приоритета, по всем рассмотренным критериям оптимальности.

2. Для проведения описанных численных исследований были разработаны и описанны следующие комплексы программ:

2.1 «Программный комплекс для моделирования процесса составления адаптивных расписаний для многолинейной системы с ограниченными восполнимыми ресурсами». Данное ПО ЭВМ описано в части 4.1.4 этой главы. Свидетельство №2021681570 о регистрации по комплекса программ находится приложении Б.2;

2.2 «Система имитационного моделирования для исследования проблем составления расписаний в распределенных системах с ограниченными ресурсами». Описана в части 4.2.4 данной главы. Зарегистрирована в Роспатенте под №2022610178, свидетельсво о регистрации находится приложении Б.2.

3. Результаты исследования были внедрены в программное обеспечение «Многопроцессорное клиент-серверное приложение с механизмом адаптивного составления расписаний для обработки информационных запросов в распределенных хранилищах данных», используемой Институтом Астрофизики Частиц Технологического Института Карлсруэ (Германия) в рамках программного комплекса, используемого для обеспечения доступа к данным экспериментов космикимикрофизики.

   Внедрение описано в части 4.3 и подтверждено Актом о внедрении (приложение В).

   Свидетельство о регистрации данного программного обеспечения в Ростатенте за №2021680467 представлено приложении Б.1.

Заключение

Основной результат диссертационной работы заключается в разработке дискретных и дискретно-непрерывных математических моделей составления расписаний в СА, и разработке соответствующих данным моделям эвристических алгоритмов составления расписаний на основе приоритето-порождающих функционалов, что позволило разработать систему имитационного моделирования СА и комплекс программ для проведения численных экспериментов, позволивших выполнить комплексное исследование проблем составления расписаний в СА.

1. Осуществлен анализ проблематики моделирования ресурсных ограничений в современных информационно-вычислительных системах. Выделено понятие информационного ресурса и категория «качественная доступность» ресурсных ограничений. Выделен и описан класс распределённых информационно-вычислительных систем «системы агрегации», отличительной чертой которых является наличие доступа к ряду распределенных информационных ресурсов, используемых для решения задачи агрегированного поиска. Сформулированы определения допустимости, эффективности и оптимальности расписаний, поставлена оптимизационная задача построения расписаний в системах агрегации с ресурсными ограничениями качественной доступности.
2. Разработан комплекс математических моделей составления расписаний для нескольких приборов для случаев дискретных и дискретно-непрерывных ограничений качественного доступа к удалённым информационных ресурсам.
3. Проведено математическое исследование существования аналитического решения для случаев дискретных ограничений на ресурсы и различных видов функции расхода ресурса для модели с непрерывными ограничениями на ресурсы, использовано для обоснования необходимости разработки численных алгоритмов решения.
4. Разработаны новые алгоритмы численного решения задач составления расписаний в многоприборных системах с ресурсами, обладающими ограниченной качественной доступностью, основанные на приоритето-порождающих функционалах для случаев числа приборов равного числу ресурсов и числа приборов меньшего числа ресурсов;

5. Разработаны комплексы программ для направленного численного эксперимента по исследованию свойств разработанных алгоритмов, имитационного моделирования поведения СА при различных заданных параметрах;
6. Разработанные численные методы и алгоритмы реализованы в виде комплекса программ для работы с пользовательскими заявками и диспетчеризации задач в центре сбора и анализа данных экспериментальной астрофизики частиц GRADLCI.

Полученные в диссертационной работе результаты были **приняты к использованию** в центре анализа и обработки данных GRADLCI для составления расписаний обработки пользовательских запросов и подтвердили свою практическую целесообразность.

Список литературы

1. *Tokareva V.* Optimization of request processing times for a heterogeneous data aggregation platform / V. Tokareva // Journal of Physics: Conference Series. — 2021. — Vol. 1740, no. 1. — P. 012058.
2. *Tokareva V.* Towards a coherent Data Life Cycle in Astroparticle Physics / V. Tokareva, A. Haungs, D. Kang, D. Kostunin, F. Polgart, D. Wochele, J. Wochele // Journal of Physics: Conference Series. — 2020. — Vol. 1525, no. 1. — P. 012070.
3. *Tokareva V.* Data Aggregation Platform for Experiments of Astroparticle Physics / V. Tokareva, A. Haungs, D. Kang, F. Polgart, D. Wochele, J. Wochele // CEUR Workshop Proceedings. — 2020. — Vol. 2679. — P. 134–142.
4. *Wochele D.* Data structure adaption from large-scale experiment for public reuse / D. Wochele, J. Wochele, F. Polgart, V. Tokareva, D. Kang, A. Haungs // CEUR Workshop Proceedings. — 2019. — Vol. 2406. — P. 114–121.
5. *Tokareva V.* Development of a data infrastructure for a global data and analysis center in astroparticle physics / V. Tokareva, A. Haungs, D. Kang, D. Kostunin, F. Polgart, D. Wochele, J. Wochele // CEUR Workshop Proceedings. — 2019. — Vol. 2406. — P. 106–113.
6. *Haungs A.* German-Russian Astroparticle Data Life Cycle Initiative / A. Haungs, I. Bychkov, J. Dubenskaya, O. Fedorov, A. Heiss, D. Kang, Y. Kazarina, E. E. Korosteleva, D. Kostunin, A. Kryukov, A. Mikhailov, M.-D. Nguyen, F. Polgart, S. Polyakov, E. Postnikov, A. Shigarov, D. Shipilov, A. Streit, V. Tokareva, D. Wochele, J. Wochele, D. Zhurov // Proceedings of Science. — 2019. — Vol. 358. — P. 284.
7. *Bychkov I.* Russian–German astroparticle data life cycle initiative / I. Bychkov, A. Demichev, J. Dubenskaya, O. Fedorov, A. Haungs, A. Heiss, D. Kang, Y. Kazarina, E. Korosteleva, D. Kostunin, A. Kryukov, A. Mikhailov, M.-D. Nguyen, S. Polyakov, E. Postnikov, A. Shigarov, D. Shipilov, A. Streit, V. Tokareva, D. Wochele, J. Wochele, D. Zhurov // Data. — 2018. — Vol. 3, no. 4. — P. 56.

8. *Tokareva V. A.* Current status of data center for cosmic rays based on KCDC / V. A. Tokareva, D. G. Kostunin, A. Haungs // CEUR Workshop Proceedings. — 2018. — Vol. 2267. — P. 405–409.

9. *Tokareva V.* German-Russian Astroparticle Data Life Cycle Initiative to foster Big Data Infrastructure for Multi-Messenger Astronomy / V. Tokareva, I. Bychkov, A. Demichev, J. Dubenskaya, O. Fedorov, A. Haungs, D. Kang, Y. Kazarina, E. Korosteleva, D. Kostunin, A. Kryukov, A. Mikhailov, M.-D. Nguyen, F. Polgart, S. Polyakov, E. Postnikov, A. Shigarov, D. Shipilov, A. Streit, D. Wochele, J. Wochele, D. Zhurov // Proceedings of Science. — 2021. — Vol. 395. — P. 938.

10. *Tokareva V.* Optimization of aggregated requests scheduling in a system with non-separable resources and parallel data processing / V. Tokareva // AIP Conference Proceedings. Vol. 2377. — AIP Publishing LLC. 2021. — P. 040009.

11. *Tokareva V. A.* Schedules with Priorities for Online Resource Management Problems in Aggregated Data Access Systems / V. A. Tokareva // Automation and Remote Control. — 2021. — Vol. 82, no. 11. — P. 1939–1948.

12. Многопроцессорное клиент-серверное приложение с механизмом адаптивного составления расписаний для обработки информационных запросов в распределенных хранилищах данных : Свидетельство о государственной регистрации программы для ЭВМ № 2021680467 / В. А. Токарева, Ю. В. Бондаренко. — № 2021680026 ; заявл. 06.12.2021 ; опубл. 10.12.2021 (Рос. Федерация).

13. Программный комплекс для моделирования процесса составления адаптивных расписаний для многолинейной системы с ограниченными восполнимыми ресурсами : Свидетельство о государственной регистрации программы для ЭВМ № 2021681570 / В. А. Токарева. — № 2021668937 ; заявл. 23.11.2021 ; опубл. 23.12.2021 (Рос. Федерация).

14. Система имитационного моделирования для исследования проблем составления расписаний в распределенных системах с ограниченными ресурсами : Свидетельство о государственной регистрации программы для ЭВМ № 2022610178 / В. А. Токарева. — № 2021681892 ; заявл. 24.12.2021 ; опубл. 10.01.2022 (Рос. Федерация).

15. *Токарева В. А.* Расписания с приоритетами для задач онлайн-управления ресурсами в системе агрегированного доступа к данным / В. А. Токарева // Автоматика и телемеханика. — 2021. — № 11. — С. 135—147.

16. *Bellman R.* Mathematical aspects of scheduling theory / R. Bellman // Journal of the Society for Industrial and Applied Mathematics. — 1956. — Vol. 4, no. 3. — P. 168–205.

17. *Gantt H. L.* A graphical daily balance in manufacture / H. L. Gantt // Transactions of the American Society of Mechanical Engineers. — 1903. — Vol. 24. — P. 1322–1336.

18. *Blazewicz J.* Handbook on scheduling / J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, J. Weglarz. — Springer, 2019.

19. *Лазарев А.* Теория расписаний / А. Лазарев, Е. Гафаров. — Москва : МГУ, 2011. — (Задачи и алгоритмы).

20. *Танаев В. С.* Теория расписаний: одностадийные системы / В. С. Танаев, В. С. Гордон, Я. М. Шафранский. — Наука. Гл. ред. физ.-мат. лит., 1984.

21. *Бразовская Н. В.* Методы оптимизации: Учеб. пособие. / Н. В. Бразовская. — Барнаул : Изд-во АлтГТУ им. И. И. Ползунова, 2000. — С. 120.

22. *Pinedo M.* Scheduling. Vol. 29 / M. Pinedo. — Springer, 2012.

23. *Chretienne P.* Scheduling theory and its applications / P. Chretienne, E. G. Coffman Jr, J. K. Lenstra, Z. Liu // Journal of the Operational Research Society. — 1997. — Vol. 48, no. 7. — P. 764–765.

24. *Аничкин А.* Современные модели и методы теории расписаний / А. Аничкин, В. Семенов // Труды Института системного программирования РАН. — 2014. — Т. 26, № 3. — С. 5—50.

25. *Gupta S. K.* Single machine scheduling research / S. K. Gupta, J. Kyparisis // Omega. — 1987. — Vol. 15, no. 3. — P. 207–227.

26. *Cheng T.* A state-of-the-art review of parallel-machine scheduling research / T. Cheng, C. Sin // European Journal of Operational Research. — 1990. — Vol. 47, no. 3. — P. 271–292.

27. *Edis E. B.* Parallel machine scheduling with flexible resources / E. B. Edis, C. Oguz // Computers & Industrial Engineering. — 2012. — Vol. 63, no. 2. — P. 433–447.

28. *Senthilkumar P.* Literature review of single machine scheduling problem with uniform parallel machines / P. Senthilkumar, S. Narayanan. — 2010.

29. *Brucker P.* A branch & bound method for the general-shop problem with sequence dependent setup-times / P. Brucker, O. Thiele // Operations-Research-Spektrum. — 1996. — Vol. 18, no. 3. — P. 145–161.

30. *Amjad M. K.* Recent research trends in genetic algorithm based flexible job shop scheduling problems / M. K. Amjad, S. I. Butt, R. Kousar, R. Ahmad, M. H. Agha, Z. Faping, N. Anjum, U. Asgher // Mathematical Problems in Engineering. — 2018. — Vol. 2018.

31. *Xie J.* Review on flexible job shop scheduling / J. Xie, L. Gao, K. Peng, X. Li, H. Li // IET Collaborative Intelligent Manufacturing. — 2019. — Vol. 1, no. 3. — P. 67–77.

32. *Ahmadian M. M.* Four decades of research on the open-shop scheduling problem to minimize the makespan / M. M. Ahmadian, M. Khatami, A. Salehipour, T. Cheng // European Journal of Operational Research. — 2021. — Vol. 295, no. 2. — P. 399–426.

33. *Potts C. N.* Scheduling with batching: A review / C. N. Potts, M. Y. Kovalyov // European journal of operational research. — 2000. — Vol. 120, no. 2. — P. 228–249.

34. *Li S.* Unbounded parallel-batching scheduling with two competitive agents / S. Li, J. Yuan // Journal of Scheduling. — 2012. — Vol. 15, no. 5. — P. 629–640.

35. *Mohagheghi E.* A survey of real-time optimal power flow / E. Mohagheghi, M. Alramlawi, A. Gabash, P. Li // Energies. — 2018. — Vol. 11, no. 11. — P. 3142.

36. *Hartmann S.* An updated survey of variants and extensions of the resource-constrained project scheduling problem / S. Hartmann, D. Briskorn // European Journal of Operational Research. — 2022. — Vol. 297, no. 1. — P. 1–14.

37. *Pellerin R.* A survey of hybrid metaheuristics for the resource-constrained project scheduling problem / R. Pellerin, N. Perrier, F. Berthaut // European Journal of Operational Research. — 2020. — Vol. 280, no. 2. — P. 395–416.

38. *Самсонова Н. В.* Составление расписания в высшем учебном заведении: математические методы и программные продукты / Н. В. Самсонова, А. Б. Симонов // E-Management. — 2018. — Т. 1, № 1. — С. 60—69.

39. *Зак Ю.* Последовательные и стохастические алгоритмы решения многоэкстремальных задач и задач теории расписаний в условиях системы ограничений / Ю. Зак. — 2017.

40. *Nouri H. E.* A classification schema for the job shop scheduling problem with transportation resources: state-of-the-art review / H. E. Nouri, O. B. Driss, K. Ghédira // Artificial intelligence perspectives in intelligent systems. — 2016. — P. 1–11.

41. *Patel D.* A systematic review on scheduling public transport using IoT as tool / D. Patel, Z. Narmawala, S. Tanwar, P. K. Singh // Smart innovations in communication and computational sciences. — 2019. — P. 39–48.

42. *Операйло К. В.* Обзор моделей оптимизации расписания городского транспорта / К. В. Операйло, М. А. Якимов, Е. Н. Новикова // Теоретические и практические аспекты формирования и развития. — 2021. — С. 151.

43. *Karp R. M.* Reducibility among combinatorial problems / R. M. Karp // Complexity of computer computations. — Springer, 1972. — P. 85–103.

44. *Karp R. M.* On the computational complexity of combinatorial problems / R. M. Karp // Networks. — 1975. — Vol. 5, no. 1. — P. 45–68.

45. *Lenstra J. K.* Complexity of scheduling under precedence constraints / J. K. Lenstra, A. Rinnooy Kan // Operations Research. — 1978. — Vol. 26, no. 1. — P. 22–35.

46. *Ullman J.* Bounds on the complexity of the longest common subsequence problem / J. Ullman, A. Aho, D. Hirschberg // Journal of the ACM (JACM). — 1976. — Vol. 23, no. 1. — P. 1–12.

47. *Ullman J. D.* NP-complete scheduling problems / J. D. Ullman // Journal of Computer and System sciences. — 1975. — Vol. 10, no. 3. — P. 384–393.

48. *Ullman J. D.* Parallel complexity of logical query programs / J. D. Ullman, A. Van Gelder // Algorithmica. — 1988. — Vol. 3, no. 1. — P. 5–42.

49. *Garey M. R.* Computers and intractability. Vol. 174 / M. R. Garey, D. S. Johnson. — freeman San Francisco, 1979.

50. *Garey M. R.* The complexity of computing Steiner minimal trees / M. R. Garey, R. L. Graham, D. S. Johnson // SIAM journal on applied mathematics. — 1977. — Vol. 32, no. 4. — P. 835–859.

51. *Танаев В.* Теория расписаний. Групповые технологии / В. Танаев, М. Ковалев, Я. Шафранский // Минск: ИТК НАН Беларуси. — 1998.

52. *Танаев В. С.* Введение в теорию расписаний / В. С. Танаев, В. В. Шкурба. — Наука. Гл. ред. физ.-мат. лит., 1975.

53. *Танаев В. С.* Теория расписаний: многостадийные системы / В. С. Танаев, Ю. Н. Сотсков, В. А. Струсевич. — Наука. Гл. ред. физ.-мат. лит., 1989.

54. *Jansen K.* On the Complexity of Scheduling Problems With a Fixed Number of Identical Machines / K. Jansen, K. Kahler // arXiv preprint arXiv:2202.07932. — 2022.

55. *Brucker P.* On the Complexity of Scheduling. / P. Brucker, S. Knust. — 2009.

56. *S. K.* Lower bounds on the minimum project duration / K. S. // Handbook on Project Management and Scheduling. — 2015. — Vol. 1. — P. 43–55.

57. *Корбут А. А.* Методы ветвей и границ. Обзор теорий, алгоритмов, программ и приложений / А. А. Корбут, И. Х. Сигал, Ю. Ю. Финкельштейн // Operations Forsch. Statist., Ser. Optimiz. — 1977. — Т. 8, № 2. — С. 253—280.

58. *Корбут А. А.* Приближенные методы дискретного программирования / А. А. Корбут, Ю. Ю. Финкельштейн // Изв. АН СССР. Техн. кибернет. — 1983. — Т. 1. — С. 165—176.

59. *Севастьянов С. В.* Геометрические методы и эффективные алгоритмы в теории расписаний : дисс. д-ра физ.-мат. наук / Севастьянов С. В. — Новосибирск, 2000. — 280 с.

60. *Sevastianov S. V.* Computer-Aided Way to Prove Theorems in Scheduling / S. V. Sevastianov, I. D. Tchernykh // European Symposium on Algorithms. — Berlin, Heidelberg : Springer, 1998. — P. 502–513.

61. *Vazirani V. V.* Approximation algorithms. Vol. 1 / V. V. Vazirani. — Springer, 2001.

62. *Arora S.* Polynomial time approximation schemes for Euclidean TSP and other geometric problems / S. Arora // Proceedings of 37th Conference on Foundations of Computer Science. — IEEE. 1996. — P. 2–11.

63. *Ковалёв М. Я.* Интервальные e-приближённые алгоритмы решения дискретных экстремальных задач : дисс. канд. физ.-мат. наук / Ковалёв М. Я. — Минск : Белорусский государственный университет, 1986. — 110 с.

64. *Mastrolilli M.* Efficient Approximation Schemes for Scheduling Problems with Release Dates and Delivery Times / M. Mastrolilli // Journal of Scheduling. — 2003. — Vol. 6, no. 6. — P. 521–531.

65. *Kellerer H.* A fast FPTAS for single machine scheduling problem of minimizing total weighted earliness and tardiness about a large common due date / H. Kellerer, K. Rustogi, V. A. Strusevich // Omega. — 2020. — Vol. 90. — P. 101992.

66. *Woeginger G. J.* When does a dynamic programming formulation guarantee the existence of an FPTAS? / G. J. Woeginger // Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms. — 1999. — P. 820–829.

67. *Mika M.* A metaheuristic approach to scheduling workflow jobs on a grid / M. Mika, G. Waligora, J. Węglarz // Grid resource management. — Springer, 2004. — P. 295–318.

68. *Weglarz J.* Modelling and control of dynamic resource allocation project scheduling systems / J. Weglarz // Optimization and control of dynamic operational research models. — 1982. — Vol. 105. — P. 140.

69. *Gupta A. K.* Design and performance evaluation of Smart Job First Dynamic Round Robin (SJFDRR) scheduling algorithm with smart time quantum / A. K. Gupta, N. S. Yadav, D. Goyal // American Academic Scientific Research Journal for Engineering, Technology, and Sciences. — 2016. — Vol. 26, no. 4. — P. 66–78.

70. *Arora N.* Review on Task Scheduling Algorithms in Cloud Computing Environment / N. Arora // International Journal of Advanced Research in Computer Science. — 2017. — Vol. 8, no. 4. — P. 401–403.

71. *Alaa F.* Improved Round Robin Scheduling Algorithm With Varying Time Quantum / F. Alaa, M. M. Zoulikha, B. Hayat // 2020 Second International Conference on Embedded & Distributed Systems (EDiS). — IEEE, 2020. — P. 33–37.

72. *Lenstra J. K.* Complexity of machine scheduling problems / J. K. Lenstra, A. H. G. R. Kan, P. Brucker // Annals of Discrete Mathematics. — 1977. — Vol. 1. — P. 343–362.

73. *Graham R. L.* Optimization and approximation in deterministic sequencing and scheduling: a survey / R. L. Graham, E. L. Lawler, J. K. Lenstra, A. R. Kan // Annals of discrete mathematics. Vol. 5. — Elsevier, 1979. — P. 287–326.

74. *Blazewicz J.* Scheduling subject to resource constraints: classification and complexity / J. Blazewicz, J. K. Lenstra, A. R. Kan // Discrete applied mathematics. — 1983. — Vol. 5, no. 1. — P. 11–24.

75. *Brucker P.* Resource-constrained project scheduling: Notation, classification, models, and methods / P. Brucker, A. Drexl, R. Möhring, K. Neumann, E. Pesch // European journal of operational research. — 1999. — Vol. 112, no. 1. — P. 3–41.

76. *Noori S.* Multi-mode resource constrained project scheduling problem: a survey of variants, extensions, and methods / S. Noori, K. Taghizadeh // International Journal of Industrial Engineering & Production Research. — 2018. — Vol. 29, no. 3. — P. 293–320.

77. *Blazewicz J.* Review of properties of different precedence graphs for scheduling problems / J. Blazewicz, D. Kobler // European Journal of Operational Research. — 2002. — Vol. 142, no. 3. — P. 435–443. — URL: https://www.sciencedirect.com/science/article/pii/S0377221701003794.

78. *Błażewicz J.* Scheduling computer and manufacturing processes / J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, J. Weglarz. — springer science & Business media, 2001.

79. *Комлев Н.* Словарь иностранных слов / Н. Комлев. — 2006.

80. *Юрасов А. В.* Основы электронной коммерции. Т. 480 / А. В. Юрасов. — Москва : Горячая линия-Телеком, 2008.

81. *Van Steen M.* Distributed systems principles and paradigms / M. Van Steen, A. Tanenbaum // Network. — 2002. — Vol. 2. — P. 28.

82. *Воеводин В. В.* Параллельные вычисления / В. В. Воеводин. — БХВ-Петербург, 2004.

83. *Turay B.* Analysis of Seven Layered Architecture of Osi Model / B. Turay // Journal For Innovative Development in Pharmaceutical and Technical Science (JIDPTS) Volume. — 2019. — Vol. 2. — P. 73–77.

84. *Скобелев П. О.* Открытые мультиагентные системы для оперативной обработки информации в процессах принятия решений / П. О. Скобелев // Автометрия. — 2002. — Т. 38, № 6. — С. 45.

85. *Виттих В. А.* Мультиагентные модели взаимодействия для построения сетей потребностей и возможностей в открытых системах / В. А. Виттих, П. О. Скобелев // Автоматика и телемеханика. — 2003. — № 1. — С. 177—185.

86. *Кузнецов Н. А.* Информационное взаимодействие в технических и живых системах / Н. А. Кузнецов. — 2001.

87. *Кузнецов Н. А.* Методы анализа и синтеза модульных информационно-управляющих систем / Н. А. Кузнецов, В. В. Кульба, С. С. Ковалевский, С. А. Косяченко. — 2002.

88. *Городецкий В. И.* Прикладные многоагентные системы группового управления / В. И. Городецкий, О. В. Карсаев, В. В. Самойлов, С. В. Серебряков // Искусственный интеллект и принятие решений. — 2009. — № 2. — С. 3—24.

89. *Городецкий В.* Самоорганизация и многоагентные системы. I. Модели многоагентной самоорганизации / В. Городецкий // Известия Российской академии наук. Теория и системы управления. — 2012. — № 2. — С. 92—92.

90. *Зайцев И. Д.* Многоагентные системы в моделировании социально-экономических отношений: исследование поведения и верификация свойств с помощью цепей Маркова : дисс. канд. тех. наук: 05.13.10 / Зайцев Иван Дмитриевич. — Новосибирск : Сибирский Государственный Университет Телекоммуникаций и Информатики, 2014. — 142 с.

91. *Шабунин А.* Разработка мультиагентной системы адаптивного управления ресурсами ОАО «РЖД» / А. Шабунин, Н. Кузнецов, П. Скобелев, И. Бабанин, С. Кожевников, Е. Симонова, М. Степанов, А. Царев // Мехатроника, автоматизация, управление. — 2013. — № 1. — С. 23—29.

92. *Kelley Jr J. E.* Critical-path planning and scheduling: Mathematical basis / J. E. Kelley Jr // Operations research. — 1961. — Vol. 9, no. 3. — P. 296–320.

93. *Brand J. D.* The resource scheduling problem in construction / J. D. Brand, W. Meyer, L. R. Shaffer. — Department of Civil Engineering, University of Illinois, 1964.

94. *Davis E. W.* Resource allocation in project network models-A survey / E. W. Davis // Journal of Industrial Engineering. — 1966. — Vol. 17, no. 4. — P. 177–188.

95. *Moodie C. L.* Project resource balancing by assembly line balancing techniques / C. L. Moodie, D. E. MANDEVIL // Journal of Industrial Engineering. — 1966. — Vol. 17, no. 7. — P. 377.

96. *Johnson T. J. R.* An algorithm for the resource constrained project scheduling problem : PhD thesis / Johnson Thomas Joel Russell. — Massachusetts Institute of Technology, 1967.

97. *Wiest J. D.* A heuristic model for scheduling large projects with limited resources / J. D. Wiest // Management Science. — 1967. — Vol. 13, no. 6. — B–359.

98. *Garey M. R.* Bounds for multiprocessor scheduling with resource constraints / M. R. Garey, R. L. Graham // SIAM Journal on Computing. — 1975. — Vol. 4, no. 2. — P. 187–200.

99. *Garey M. R.* Complexity results for multiprocessor scheduling under resource constraints / M. R. Garey, D. S. Johnson // SIAM Journal on Computing. — 1975. — Vol. 4, no. 4. — P. 397–411.

100. *Garey M. R.* Resource constrained scheduling as generalized bin packing / M. R. Garey, R. L. Graham, D. S. Johnson, A. C.-C. Yao // Journal of Combinatorial Theory, Series A. — 1976. — Vol. 21, no. 3. — P. 257–298.

101. *Goyal D. K.* Scheduling equal execution time tasks under unit resource restriction / D. K. Goyal. — Washington State University, Computer Science Department, 1976.

102. *Davis E. W.* Project scheduling under resource constraints—historical review and categorization of procedures / E. W. Davis // AIIE Transactions. — 1973. — Vol. 5, no. 4. — P. 297–313.

103. *Herroelen W. S.* Resource-constrained project scheduling—the state of the art / W. S. Herroelen // Journal of the Operational Research Society. — 1972. — Vol. 23, no. 3. — P. 261–275.

104. *Błażewicz J.* Scheduling under resource constraints: Deterministic models / J. Błażewicz. — JC Baltzer, 1986.

105. *Blazewicz J.* Scheduling independent fixed-type tasks / J. Blazewicz, W. Kubiak, J. Szwarcfiter // Advances in project scheduling. — Elsevier, 1989. — P. 225–236.

106. *Ullman J.* NP-complete scheduling problems / J. Ullman // J. Comput. System Sci. — 1975. — Vol. 10. — P. 384–393.

107. *Horvath E.* A level algorithm for preemptive scheduling / E. Horvath, S. Lam, R. Sethi // J. Assoc. Comput. Mach. — 1977. — Vol. 24, no. 1. — P. 32–43.

108. *Johnson S.* Optimal Two-and-Three-Stage Production Schedules with Set-up Times included / S. Johnson // Naval Res. Logist. Quart. — 1954. — Vol. 1. — P. 61–68.

109. *Blazewicz J.* Complexity of computer scheduling algorithms under resource constraints / J. Blazewicz // Proc. of 1^< st> meeting AFCET-SMF on Applied Mathematics, 1978. — 1978.

110. *Krause K. L.* Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems / K. L. Krause, V. Y. Shen, H. D. Schwetman // Journal of the ACM (JACM). — 1975. — Vol. 22, no. 4. — P. 522–550.

111. *Coffman E. G.* Approximation algorithms for bin-packing—an updated survey / E. G. Coffman, M. R. Garey, D. S. Johnson // Algorithm design for computer system design. — Springer, 1984. — P. 49–106.

112. *Röck H.* Machine aggregation heuristics in shop scheduling / H. Röck, G. Schmidt. — Univ.-Bibl. d. Techn. Univ., 1982.

113. *Józefowska J.* On a methodology for discrete–continuous scheduling / J. Józefowska, J. Węglarz // European Journal of Operational Research. — 1998. — Vol. 107, no. 2. — P. 338–353.

114. *Węglarz J.* Project scheduling with continuously-divisible, doubly constrained resources / J. Węglarz // Management Science. — 1981. — Vol. 27, no. 9. — P. 1040–1053.

115. *Nowicki E.* Optimal control of a complex of independent operations / E. Nowicki, S. Zdrzałka // International Journal of Systems Science. — 1981. — Vol. 12, no. 1. — P. 77–93.

116. *Nowicki E.* Optimal control policies for resource allocation in an activity network / E. Nowicki, S. Zdrzałka // European Journal of Operational Research. — 1984. — Vol. 16, no. 2. — P. 198–214.

117. *Nowicki E.* Scheduling jobs with controllable processing times as an optimal control problem / E. Nowicki, S. Zdrzałka // International Journal of Control. — 1984. — Vol. 39, no. 4. — P. 839–848.

118. *Węglarz J.* Scheduling under continuous performing speed vs. resource amount activity models / J. Węglarz // Advances in Project Scheduling. — Elsevier, 1989. — P. 273–295.

119. *Janiak A.* On time-optimal control of a sequence of projects of activities under time-variable resource / A. Janiak, A. Stankiewicz // IEEE transactions on automatic control. — 1988. — Vol. 33, no. 3. — P. 313–316.

120. *Węglarz J.* Project scheduling with finite or infinite number of activity processing modes–A survey / J. Węglarz, J. Józefowska, M. Mika, G. Waligóra // European Journal of operational research. — 2011. — Vol. 208, no. 3. — P. 177–205.

121. *Janiak A.* Single machine sequencing with linear models of jobs subject to precedence constraints / A. Janiak // Archiwum Automatyki i Telemechaniki. — 1988. — Vol. 33. — P. 203–210.

122. *Janiak A.* Time-optimal control in a single machine problem with resource constraints / A. Janiak // Automatica. — 1986. — Vol. 22. — P. 745–747.

123. *Janiak A.* Exact and Approximation Algorithms of Job Scheduling and Resource Allocation in Discrete Industrial Processes / A. Janiak // Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej. — Wrocław, 1991. — Vol. 87, no. 20.

124. *Janiak A.* Selected Problems and Algorithms of Scheduling and Resource Allocation / A. Janiak. — Warszawa : Akademicka Oficyna Wydawnicza PLJ, 1999.

125. *Lawler E. L.* Optimal sequencing of a single machine subject to precedence constraints / E. L. Lawler // Manage. Sci. — 1973. — Vol. 19. — P. 544–546.

126. *Lenstra J. K.* Complexity of machine scheduling problems / J. K. Lenstra, A. R. Kan, P. Brucker // Annals of discrete mathematics. T. 1. — Elsevier, 1977. — C. 343—362.

127. *Janiak A.* Single machine scheduling problem with a common deadline and resource dependent release dates / A. Janiak // Eur. J. Oper. Res. — 1991. — Vol. 53. — P. 317–325.

128. *Janiak A.* Minimization of the blooming mill standstills - mathematical model. Suboptimal algorithms / A. Janiak // Zeszyty Naukowe AGH, Mechanika. — 1989. — Vol. 8. — P. 37–49.

129. *Janiak A.* Resource optimal control in some simple-machine scheduling problems / A. Janiak, T.-C. E. Cheng // IEEE Trans. Aut. Contr. — 1994. — Vol. 39. — P. 1243–1246.

130. *Janiak A.* Computational complexity analysis of single machine scheduling problems with job release dates dependent on resources / A. Janiak // Operations Research Proceedings / ed. by U. Zimmermann, U. Derigs, W. Gaul, R. Möhring, K. Schuster. — Berlin : Springer, 1997. — P. 203–207.

131. *Janiak A.* Scheduling to minimize the total weighted completion time with a constraint on the release time resource consumption / A. Janiak, C.-L. Li // Math. Comput. Model. — 1994. — Vol. 20. — P. 53–58.

132. *Bruno J.* Scheduling independent tasks to reduce mean finishing time / J. Bruno, E. G. Coffman Jr, R. Sethi // Communications of the ACM. — 1974. — Vol. 17, no. 7. — P. 382–387.

133. *Шафранский Я.* Оптимизация детерминированных систем обслуживания с древовидным частичным порядком / Я. Шафранский // Известия АН БССР. Сер. физ.-мат. наук. — 1978. — № 2. — С. 119.

134. *Цуканов М. А.* Алгоритмизация процесса диспетчеризации работы разливочных кранов как подвод к уменьшению длительности простоев сталеплавильного производства / М. А. Цуканов, О. П. Ульянова // Электрометаллургия. — 2018. — № 3. — С. 9—17.

135. *Nguyen S.* A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem / S. Nguyen, M. Zhang, M. Johnston, K. C. Tan // IEEE Transactions on Evolutionary Computation. — 2012. — Vol. 17, no. 5. — P. 621–639.

136. *Rajendran C.* A comparative study of dispatching rules in dynamic flowshops and jobshops / C. Rajendran, O. Holthaus // European journal of operational research. — 1999. — Vol. 116, no. 1. — P. 156–170.

137. *Chen B.* A flexible dispatching rule for minimizing tardiness in job shop scheduling / B. Chen, T. I. Matis // International Journal of Production Economics. — 2013. — Vol. 141, no. 1. — P. 360–365.

138. *Смирнов А. В.* О задаче упаковки в контейнеры / А. В. Смирнов // Успехи математических наук. — 1991. — Т. 46, № 4. — С. 173—174.

139. *Танаев В. С.* Теория расписаний. Одностадийные системы / В. С. Танаев, В. С. Гордон, Я. М. Шафранский. — Москва : Наука, 1984.

140. *Tanaev V. S.* Scheduling Theory. Single-Stage Systems / V. S. Tanaev, V. S. Gordon, Y. M. Shafransky. — Dordrecht/Boston/London : Kluwer Academic Publ., 1994.

141. *Танаев В. С.* Некоторые оптимизируемые функции одностадийного производства / В. С. Танаев // Докл. АН БССР. — 1965. — Т. 9, № 1. — С. 11—14.

142. *Танаев В. С.* К теории расписаний / В. С. Танаев // Докл. АН БССР. — 1964. — Т. 8, № 12. — С. 792—794.

143. *Гордон В. С.* Детерминированные системы обслуживания с одним прибором, древовидным упорядочением требований и экспоненциальными функциями штрафа / В. С. Гордон, В. С. Танаев // Вычисл. техника в машиностроении. — Ин-т техн. кибернетики АН БССР, 1973. — С. 3—10.

144. *Гордон В. С.* Об оптимальном упорядочении при последовательно-параллельных ограничениях предшествования / В. С. Гордон, Я. М. Шафранский // Изв. АН БССР. Сер. физ.-мат. наук. — 1978. — Т. 5. — С. 135.

145. *Monma C. L.* Sequencing with series-parallel precedence constraints / C. L. Monma, J. B. Sidney // Math. Oper. Res. — 1979. — Vol. 4. — P. 215–234.

146. *Березин Д. А.* Учебно-методический комплекс дисциплины «Имитационное моделирование» / Д. А. Березин. — 2008.

147. *Bychkov I.* Russian-German Astroparticle Data Life Cycle Initiative / I. Bychkov, A. Demichev, J. Dubenskaya, O. Fedorov, A. Haungs, A. Heiss, D. Kang, Y. Kazarina, E. Korosteleva, D. Kostunin, A. Kryukov, A. Mikhailov, M.-D. Nguyen, S. Polyakov, E. Postnikov, A. Shigarov, D. Shipilov, A. Streit, V. T. va, D. Wochele, J. Wochele, D. Zhurov // Data. — 2018. — Vol. 3. — P. 56.

148. KASCADE Cosmic Ray Data Centre. — Retrieved 20 June 2021. https://kcdc.iap.kit.edu.

149. *Antokhonov B.* TUNKA-133: A new array for the study of ultra-high energy cosmic rays / B. Antokhonov, S. Berezhnev, D. Besson, N. Budnev, R. Wischnevski, O. Gress, A. Diachok, A. Zablotsky, A. Zagorodnikov, N. Kalmykov, [et al.] // Bulletin of the Russian Academy of Sciences: Physics. — 2011. — Vol. 75, no. 3. — P. 367–370.

150. *Bezyazeekov P.* Towards the Tunka-Rex virtual observatory / P. Bezyazeekov, N. Budnev, O. Fedorov, O. Gress, O. Grishin, A. Haungs, T. Huege, Y. Kazarina, M. Kleifges, D. Kostunin [и др.] // arXiv preprint arXiv:1906.10425. — 2019.

151. *Antoni T.* The cosmic-ray experiment KASCADE / T. Antoni, W. Apel, F. Badea, K. Bekk, A. Bercuci, H. Blümer, H. Bozdog, I. Brancus, C. Büttner, A. Chilingarian, [et al.] // Nuclear Instruments and Methods in Physics Research Section A: accelerators, spectrometers, detectors and associated equipment. — 2003. — Vol. 513, no. 3. — P. 490–510.

152. *Schoo S.* Energy Spectrum and Mass Composition of Cosmic Rays and How to Publish Air-Shower Data / S. Schoo. — 2016.

153. *Apel W.* Final results of the LOPES radio interferometer for cosmic-ray air showers / W. Apel, J. Arteaga-Velázquez, L. Bähren, K. Bekk, M. Bertaina, P. Biermann, J. Blümer, H. Bozdog, E. Cantoni, A. Chiavassa, [et al.] // The European Physical Journal C. — 2021. — Vol. 81, no. 2. — P. 1–25.

154. *Schröder F.* Tunka-Rex: Status, Plans, and Recent Results / F. Schröder, P. Bezyazeekov, N. Budnev, O. Fedorov, O. Gress, A. Haungs, R. Hiller, T. Huege, Y. Kazarina, M. Kleifges, [et al.] // EPJ Web of Conferences. Vol. 135. — EDP Sciences. 2017. — P. 01003.

155. *Chilingarian A.* Study of extensive air showers and primary energy spectra by MAKET-ANI detector on mountain Aragats / A. Chilingarian, G. Gharagyozyan, S. Ghazaryan, G. Hovsepyan, E. Mamidjanyan, L. Melkumyan, V. Romakhin, A. Vardanyan, S. Sokhoyan // Astroparticle Physics. — 2007. — Vol. 28, no. 1. — P. 58–71.

156. *Sen A.* Metadata management: past, present and future / A. Sen // Decision Support Systems. — 2004. — Vol. 37, no. 1. — P. 151–173.

157. *JSON-RPC Working Group*. Json-rpc 2.0 specification / JSON-RPC Working Group. — 2013. — https://www.jsonrpc.org/specification, last acessed: October 22, 2021.

158. *Tokareva V.* GRADLC API Documentation / V. Tokareva. — Retrieved 20 June 2021. https://gradlc-dc.ikp.kit.edu/web/doc/.

159. *Freitas Rodrigues R. de*. Scheduling problem with multi-purpose parallel machines / R. de Freitas Rodrigues, M. C. Dourado, J. L. Szwarcfiter // Discrete Applied Mathematics. — 2014. — Vol. 164. — P. 313–319. — URL: https://www.sciencedirect.com/science/article/pii/S0166218X11004872 ; Combinatorial Optimization.

Список рисунков

1.1 Пример диаграммы Ганта для задачи календарного планирования проекта. 15
1.2 Основные свойства задачи T_{ji}. 20
1.3 Функциональная модель составления адаптивных расписаний в системе агрегации. 29
1.4 Модель сущность-связь для системы агрегации. 33
1.5 Классификация ограничений дополнительных ресурсов в задачах теории расписаний, расширенная за счёт введения категории качественной доступности ресурса. 41
1.6 Пример графа зависимостей для обслуживания задач с заданными ограничениями предшествования. 43

2.1 Декомпозиция работ на задачи. 52
2.2 Схема работы СА. 54
2.3 Сведение случая $r_1 < r_2$ к $r'_1 = r'_2 = 0$. 60

3.1 Расписание обработки задач, составленное с использованием АДД-НП. 89
3.2 Расписание обработки задач, составленное с привязкой ресурсов к определённому прибору с обслужсиванием в порядке следования работ. 90
3.3 Результирующее расписание для примера 3.2. Исполльзование ресурсов задачами обозначено следующим образом: R_1-желтый, R_2 - серый, R_3 - голубой, R_4 зелёный, R_5 - фиолетовый. 93

4.1 Блок-схема основной программы для проведения вычислительного эксперимента. 98
4.2 Блок-схема генератора примеров. 99
4.3 Результаты моделирования времён-критериев оптимальности расписания для различных правил определения приоритета, количества ресурсов и средних частот запросов. Для маркировки ПД использованы следующие обозначения: FIFO, SPT, STPT, LIFO, LPT, LTPT. 101
4.4 Диаграмма потока данных при взаимодействии модулей. 109

4.5 Запущенная программа на Python, вывод информации в командную строку. 110
4.6 Запущенная программа на Python, вывод информации в виде графика (показан один из графиков для случая 8 хранилищ. 111
4.7 Модель «Чёрный ящик». 112
4.8 Представление системы имитационного моделирования в виде «Белого ящика». 113
4.9 Блок-схема моделирующего алгоритма для проведения численного эксперимента. 114
4.10 Блок-схема алгоритма генерации задач составления расписаний. 115
4.11 Времена работы алгоритма двухуровневой диспетчеризации для различного количества работ при количестве приборов равном 3, количестве ресурсов и максимальном числе задач в работе равном 4. Аппроксимированы сверху и снизу функцией $C\,nS\log nS$, $S = 4$, $C_u = 8 \times 10^{-5}$ для ограничения сверху и $C_l = 2 \times 10^{-6}$ для ограничения снизу. 118
4.12 Отношения критериев оптимальности для различных правил диспетчеризации из класса приоритето-порождающих функционалов 1-го порядка к критериям оптимальности для алгоритма двухуровневой диспетчеризации, для различного количества работ при разных параметрах модели. 125
4.13 Диаграмма потока данных при взаимодействии модулей системы имитационного моделирования. 127
4.14 Файловая структура СИМ. 128
4.15 Запуск СИМ в командной строке. 129
4.16 Открытие .csv файла со статистикой по временам работы алгоритмов в программе Libre Office. 130
4.17 Открытие .csv файла с подробной статистикой по критериям оптимальности в программе Libre Office. 131
4.18 Пример диаграммы Ганта, полученной программой для случая решения примера с параметрами $n = 23, m = 3, S = 4$ АДП. 132
4.19 Пример диаграммы Ганта, полученной программой для случая решения примера с параметрами $n = 20, m = 7, S = 15$ АДП. 132
4.20 Файловая структура программного комплекса. 133
4.21 Взаимодействие ключевых компонентов программного комплекса. . . 134

4.22 Схема реляционной базы метаданных. 136
4.23 Интерфейс создания нового запроса на выгрузку данных в веб-интерфейсе пользователя GRADCLI. 138
4.24 Управление запросами в пользовательском интерфейсе программного комплекса GRADCLI. 138
4.25 Асинхронная параллельная обработка данных на стороне сервера. . . . 139
4.26 Агрегирование данных из распределенных хранилищ. 140

Б.1 Свидетельство о регистрации программы «Многопроцессорное клиент-серверное приложение с механизмом адаптивного составления расписаний для обработки информационных запросов в распределенных хранилищах данных». 170
Б.2 Свидетельство о регистрации программы «Программный комплекс для моделирования процесса составления адаптивных расписаний для многолинейной системы с ограниченными восполнимыми ресурсами». 171
Б.3 Свидетельство о регистрации программы «Система имитационного моделирования для исследования проблем составления расписаний в распределённых системах с ограниченными ресурсами». 172

В.1 Акт о внедрении результатов исследования в учебный процессфакультета прикладной математики, информатики и механики Воронежского государственного университета, стр. 1. 173
В.2 Акт о внедрении результатов исследования в учебный процессфакультета прикладной математики, информатики и механики Воронежского государственного университета, стр. 2. 174
В.3 Акт о внедрении результатов исследования в международном проекте GRADLCI. 175

Список таблиц

1 Связи между сущностями логической EDR модели системы-агрегатора 35
2 Известные результаты для задач составления расписаний для приборов с ограниченными ресурсами 38
3 Список переменных математической модели 54
4 Список переменных математической модели (2.10) 58
5 Асимптотический анализ худшего времени выполнения для алгоритмов, основанных на приоритето-порождающих функционалах, в случае выделенных приборов (АППФ1-ВП) 79
6 Асимптотический анализ худшего времени выполнения для АДД-ВП . 82
7 Асимптотический анализ худшего времени выполнения для алгоритмов, основанных на приоритето-порождающих функционалах, в случае индивидуальных независимых приборов и $m < S$ (АППФ1-НП) 86
8 Ожидаемые времена выполнения работ $J_1, \ldots, J_{10}$ 91
9 Назначение задач на приборы P_1, P_2P_3 согласно АДД-НП 91
10 Изменения значений загрузки приборов по мере назначения задач на приборы 92
11 Асимптотический анализ худшего времени выполнения для АДД-НП . 95
12 Асимптотический анализ времени выполнения для алгоритмов, разработанных в рамках диссертационного исследования 96
13 Результаты моделирования времен-критериев оптимальности расписания для различных правил определения приоритета, количества хранилищ и средних частот запросов 103
14 Результаты моделирования критериев оптимальности расписания для различных правил определения приоритета. Здесь АДД — алгоритм двухуровненевой диспетчеризации 118
15 Допустимые опции запуска программного обеспечения 129
16 Наименования аспектов данных различных экспериментальных установок, по которым возможна сортировка 135

17 Наименования и принимаемые значения аспектов данных имитационного моделирования (симуляций), по которым возможна сортировка 136

18 Нотация записи информации о приборах, их отношении друг к другу и к заданиям в задачах теории расписаний 165

19 Нотация записи ограничений при обслуживании заданий в задачах теории расписаний 166

20 Возможные значения целевого функционала в задачах теории расписаний 167

21 Символы и способы нотации диаграмм «сущность-связь» (ERD) 168

22 Нотация «вороньи лапки» кардинальности и ординальности связей диаграмм «сущность-связь» (ERD) 169

Приложение А

Используемые в работе обозначения

А.1 Нотация записи задач теории расписаний

Таблица 18 — Нотация записи информации о приборах, их отношении друг к другу и к заданиям в задачах теории расписаний

Нотация	Величина	Значение
α_1	Параллельные приборы	
	$\emptyset$	Один прибор
	P	Параллельные идентичные приборы
	Q	Приборы с различной производительностью
	R	Независимые приборы
	Выделенные приборы	
	G	Задача цеха в обобщённой формулировке
	J	Задача планирования в рабочем цехе
	F	Задача планирования потоковой линии
	O	Задача планирования открытой линии
	X	Задача смешанного цеха (Mixed-shop problem)
	Многоцелевые машины [159]	
	PMPM	Индентичные многоцелевые машины
	QMPM	Многоцелевые машины с различной производительностью
α_2	Число приборов	
	$\emptyset$	Произвольное число приборов
	$n_j \leq m$	Система с m приборами
α_3	Топология сети	
	$\emptyset$	Незначительная связанность
	conn	Произвольные каналы связи
	linear array	Линейный массив (одномерная сетка)
	ring	Кольцо (замкнутый массив)
	mesh	d-мерная сетка

Продолжение на следующей странице

Нотация	Величина	Значение
	hypercube	d-мерный гиперкуб
	tree	Сеть с древовидной структурой
α_4	Одновременность расчёта и коммуникации	
	$\emptyset$	Одновременный расчёт и связь возможны
	no-overlap	Отсутствие возможности одновременного расчёта и коммуникации

Таблица 19 — Нотация записи ограничений при обслуживании заданий в задачах теории расписаний

Нотация	Величина	Значение
β_1	Прерываемость	
	0	Прерывания не допускаются
	pmtn	Прерывания разрешены
	div	Разделяемые задачи (прерывания разрешены)
β_2	Дополнительные ресурсы	
	0	Дополнительные ресурсы отсутствуют
	res	Определены дополнительные ресурсы
β_3	Отношение предшествования	
	0	Независимые задания
	prec	Произвольное отношение предшествования (DAG)
	uan	Несвязанные сети активности
	intree, outtree	intree или outtree
	tree	Дерево (in- либо outtree)
	chains	Набор цепочек
	sp-graph	Последовательно-параллельный граф
β_4	Время поступления в систему	
	0	Все задания поступают в начале ($t = 0$)
	r_j	Произвольное время поступления в систему
β_5	Время обслуживания	
	0	Произвольное время обслуживания
	$p_j = p$	Постоянное время обслуживания
	$p_{\text{lb}} \le p_j \le p_{\text{ub}}$	Ограниченное время обслуживания

Продолжение на следующей странице

Нотация	Величина	Значение
β_6	Предельные сроки завершения обслуживания	
	0	Сроки отсутствуют
	d	Указаны директивные сроки завершения обслуживания
	D	Указаны предельные сроки завершения обслуживания
β_7	Максимальное количество операций в задачах цеха	
	0	Без ограничений или не является задачей цеха
	$n_j \leq k$	Количество операций ограничено k
β_8	Ожидание на выделенных приборах	
	0	Буферые бсконечно большой ёмкости
	no-wait	Без буферизации, задания должны немедленно продолжать обработку на последующих приборах

Таблица 20 — Возможные значения целевого функционала в задачах теории расписаний

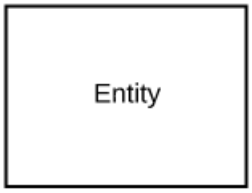
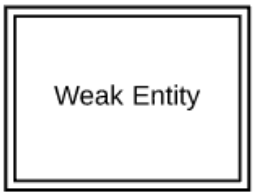
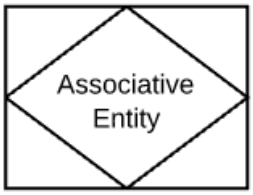
Обозначение	Формула	Название	Англ. название
$C_{\max}$	$\max_j\{C_j\}$	Длина расписания	Schedule length (makespan)
$\overline{F}$	$\frac{1}{n}\sum_{j=1}^{n} F_j$	Время нахождения требования в системе	Mean flow time
$\overline{F}_w$	$\sum_{j=1}^{n} w_j F_j / \sum_{j=1}^{n} w_j$	Взвешенное время нахождения требования в системе	Mean weighted flow time
$L_{\max}$	$\max\{L_j\}$	Максимального временное смещение	Maximum lateness
$\overline{D}$	$\frac{1}{n}\sum_{j=1}^{n} D_j$	Суммарное запаздывание требований	Mean tardiness
$\overline{D}_w$	$\sum_{j=1}^{n} w_j D_j / \sum_{j=1}^{n} w_j$	Взвешенное суммарное запаздывание требований	Mean weighted tardiness

Продолжение на следующей странице

Обозначение	Формула	Название	Англ. название
$\overline{E}$	$\frac{1}{n}\sum_{j=1}^{n} E_j$	Суммарное опережение требований	Mean earliness
$\overline{E}_w$	$\sum_{j=1}^{n} w_j E_j / \sum_{j=1}^{n} w_j$	Взвешенное суммарное опережение требований	Mean weighted earliness
$\overline{U}$	$\frac{1}{n}\sum_{j=1}^{n} U_j$	Количество запаздывающих требований	Number of tardy tasks
$\overline{U}_w$	$\sum_{j=1}^{n} w_j EUj$	Взвешенное количества запаздывающих требований	Weighted number of tardy tasks

А.2 Нотация диаграмм Сущность - Связь

Таблица 21 — Символы и способы нотации диаграмм «сущность-связь» (ERD)

Символ	Название	Описание
Символы ERD-сущностей		
	Независимая (сильная) сущность	Не зависит от других сущностей и часто называется «родительской», так как у нее в подчинении обычно находятся слабые сущности. Независимые сущности сопровождаются первичным ключом, который позволяет идентифицировать каждый экземпляр сущности.
	Зависимая (слабая) сущность	Сущность, которая зависит от сущности другого типа. Не сопровождается первичным ключом и не имеет значения в схеме без своей родительской сущности.
	Ассоциативная сущность	Соединяет экземпляры сущностей разных типов. Также содержит атрибуты, характерные для связей между этими сущностями.
Символы ERD-связей		

Продолжение на следующей странице

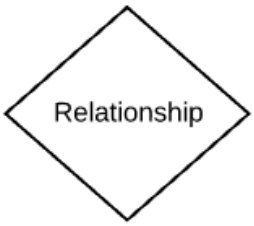
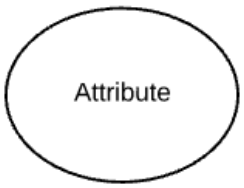
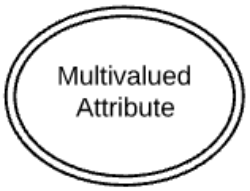
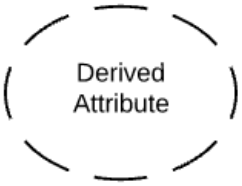
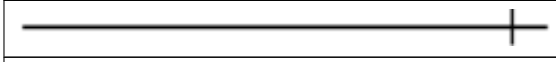
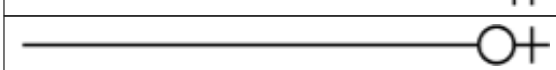
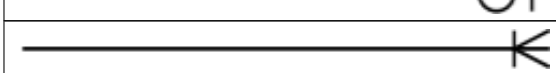
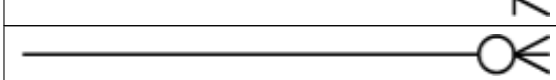
Символ	Название	Описание
	Связь	Отношение между сущностями.
	Слабая связь	Связь между зависимой сущностью и ее «хозяином».
Символы ERD-атрибутов		
	Атрибут	Характеризует сущность, а также отношения между двумя или более элементами.
	Много-значный атрибут	Атрибут, которому может быть присвоено несколько значений.
	Производ-ный атрибут	Атрибут, чье значение можно вычислить, опираясь на значения связанных с ним атрибутов.

Таблица 22 — Нотация «вороньи лапки» кардинальности и ординальности связей диаграмм «сущность-связь» (ERD)

Символ	Описание
	Один к одному.
	Один ко многим.
	Один к одному (и только одному).
	Один к нулю или одному.
	Один к одному или многим.
	Один к нулю или многим.

Приложение Б

Свидетельства о регистрации ПО ЭВМ

Б.1 Свидетельство о регистрации ПО ЭВМ №2021680467

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2021680467

«Многопроцессорное клиент-серверное приложение с механизмом адаптивного составления расписаний для обработки информационных запросов в распределенных хранилищах данных»

Правообладатель: *федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет» (ФГБОУ ВО «ВГУ») (RU)*

Авторы: *Бондаренко Юлия Валентиновна (RU), Токарева Виктория Андреевна (RU)*

Заявка № 2021680026
Дата поступления 06 декабря 2021 г.
Дата государственной регистрации
в Реестре программ для ЭВМ *10 декабря 2021 г.*

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

Рисунок Б.1 — Свидетельство о регистрации программы «Многопроцессорное клиент-серверное приложение с механизмом адаптивного составления расписаний для обработки информационных запросов в распределенных хранилищах данных».

Б.2 Свидетельство о регистрации ПО ЭВМ №2021681570

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2021681570

Программный комплекс для моделирования процесса составления адаптивных расписаний для многолинейной системы с ограниченными восполнимыми ресурсами

Правообладатель: ***Токарева Виктория Андреевна (RU)***

Автор(ы): ***Токарева Виктория Андреевна (RU)***

Заявка № **2021668937**
Дата поступления **23 ноября 2021 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***23 декабря 2021 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

Рисунок Б.2 — Свидетельство о регистрации программы «Программный комплекс для моделирования процесса составления адаптивных расписаний для многолинейной системы с ограниченными восполнимыми ресурсами».

Б.3 Свидетельство о регистрации ПО ЭВМ №2022610178

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2022610178

Система имитационного моделирования для исследования проблем составления расписаний в распределенных системах с ограниченными ресурсами

Правообладатель: *Токарева Виктория Андреевна (RU)*

Автор(ы): *Токарева Виктория Андреевна (RU)*

Заявка № 2021681892
Дата поступления 24 декабря 2021 г.
Дата государственной регистрации
в Реестре программ для ЭВМ *10 января 2022 г.*

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

Рисунок Б.3 — Свидетельство о регистрации программы «Система имитационного моделирования для исследования проблем составления расписаний в распределённых системах с ограниченными ресурсами».

Приложение В

Акты о внедрении

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Воронежский государственный университет»
(ФГБОУ ВО «ВГУ»)

Университетская пл., 1, Воронеж, 394018.
Тел. (473) 220-75-21. Факс (473) 220-87-55.
E-mail: office@main.vsu.ru
http://www.vsu.ru
ОКПО 02068120. ОГРН 1023601560510
ИНН/КПП 3666029505/366601001

23.03.2022 № 0600-041
На № _____ от ___.___.20___

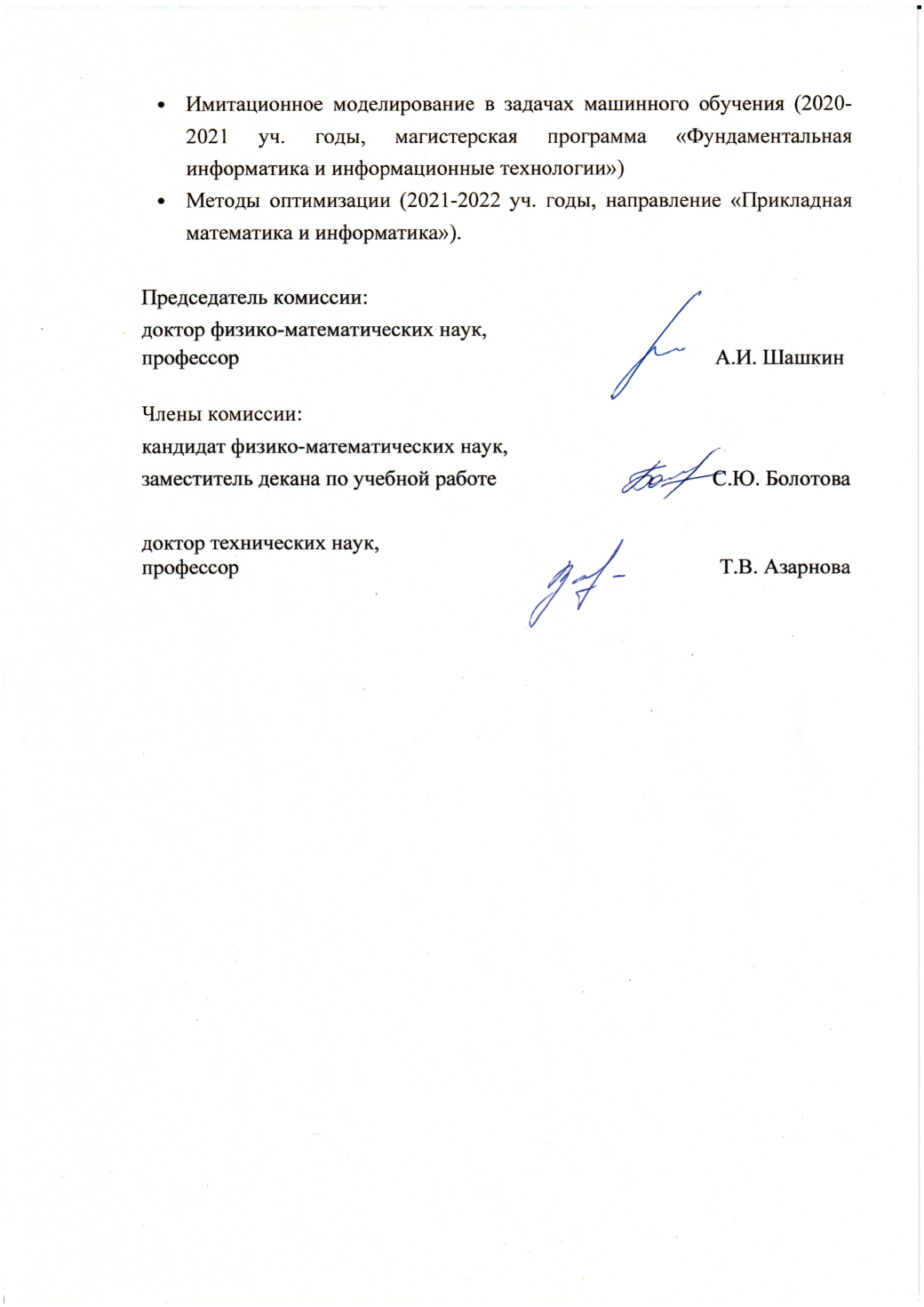
УТВЕРЖАЮ
Проректор по науке, инновациям и цифровизации
ФГБОУ ВПО «ВГУ»,
д-р хим. наук, доцент
Козадеров О. А.
«____»_____2022 г.

АКТ
о внедрении результатов диссертационного исследования
Токаревой Виктории Андреевны

Комиссия в составе: председателя комиссии – декана факультета прикладной математики, информатики и механики, доктора физико-математических наук, профессора Шашкина А.И. и членов комиссии: кандидата физико-математических наук, доцента кафедры математического обеспечения ЭВМ, заместителя декана по учебной работе Болотовой С.Ю.; заведующей кафедрой математических методов исследования операций, доктора технических наук, профессора Азарновой Т.В. составила настоящий Акт о том, что результаты диссертационного исследования Токаревой В.А. внедрены в учебный процесс факультета прикладной математики, информатики и механики в программах следующих учебных дисциплин:

- Теория игр и исследование операций (2020-2022 уч. годы, направление «Прикладная математика и информатика»)
- Имитационное моделирование (2020-2022 уч. годы, направление «Бизнес информатика»)

Рисунок В.1 — Акт о внедрении результатов исследования в учебный процессфакультета прикладной математики, информатики и механики Воронежского государственного университета, стр. 1.



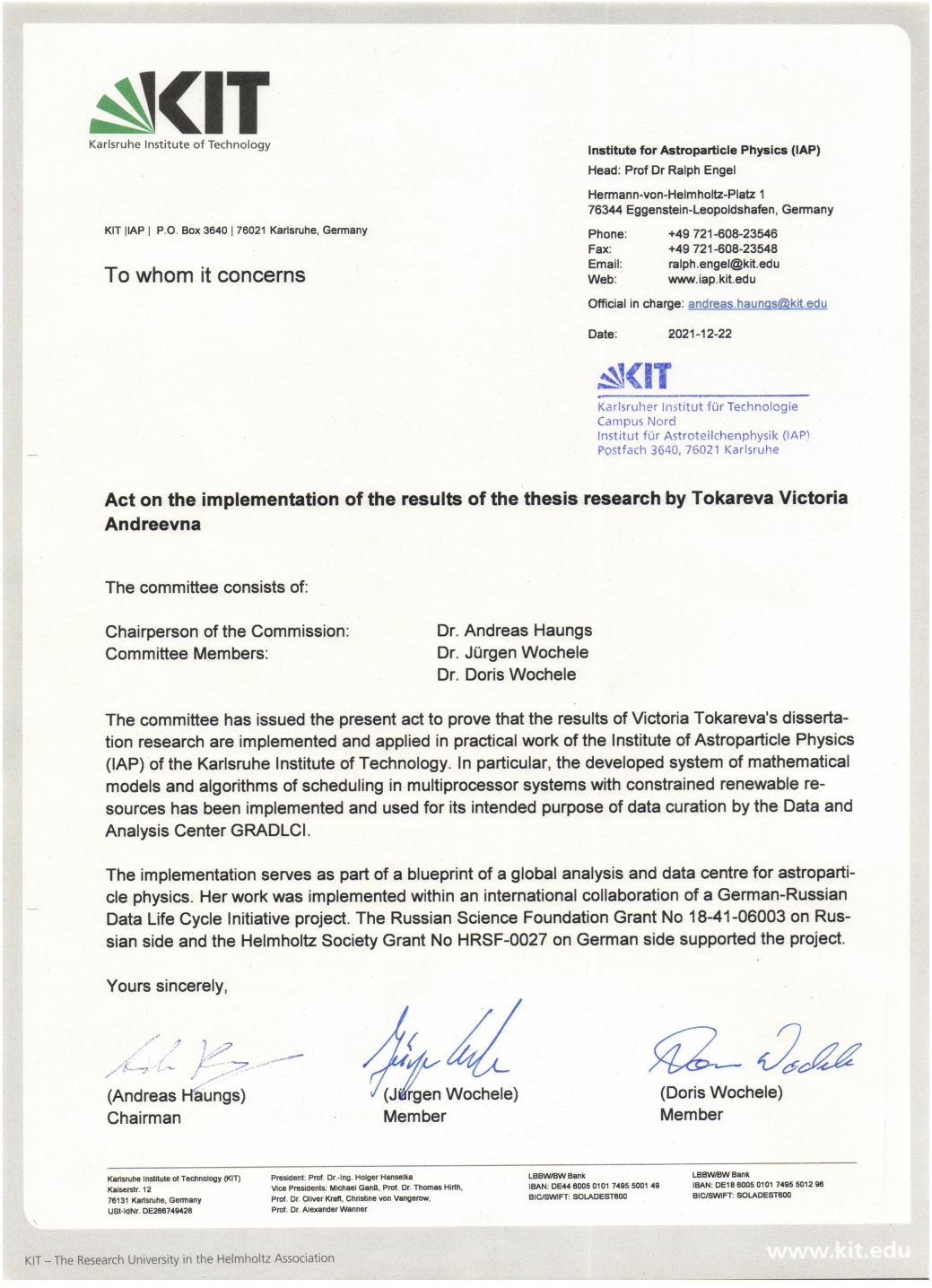
- Имитационное моделирование в задачах машинного обучения (2020-2021 уч. годы, магистерская программа «Фундаментальная информатика и информационные технологии»)
- Методы оптимизации (2021-2022 уч. годы, направление «Прикладная математика и информатика»).

Председатель комиссии:
доктор физико-математических наук,
профессор А.И. Шашкин

Члены комиссии:
кандидат физико-математических наук,
заместитель декана по учебной работе С.Ю. Болотова

доктор технических наук,
профессор Т.В. Азарнова

Рисунок В.2 — Акт о внедрении результатов исследования в учебный процессфакультета прикладной математики, информатики и механики Воронежского государственного университета, стр. 2.



Karlsruhe Institute of Technology

Institute for Astroparticle Physics (IAP)
Head: Prof Dr Ralph Engel

Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen, Germany

Phone: +49 721-608-23546
Fax: +49 721-608-23548
Email: ralph.engel@kit.edu
Web: www.iap.kit.edu

Official in charge: andreas.haungs@kit.edu

Date: 2021-12-22

KIT |IAP | P.O. Box 3640 | 76021 Karlsruhe, Germany

To whom it concerns

Karlsruher Institut für Technologie
Campus Nord
Institut für Astroteilchenphysik (IAP)
Postfach 3640, 76021 Karlsruhe

Act on the implementation of the results of the thesis research by Tokareva Victoria Andreevna

The committee consists of:

Chairperson of the Commission:	Dr. Andreas Haungs
Committee Members:	Dr. Jürgen Wochele
	Dr. Doris Wochele

The committee has issued the present act to prove that the results of Victoria Tokareva's dissertation research are implemented and applied in practical work of the Institute of Astroparticle Physics (IAP) of the Karlsruhe Institute of Technology. In particular, the developed system of mathematical models and algorithms of scheduling in multiprocessor systems with constrained renewable resources has been implemented and used for its intended purpose of data curation by the Data and Analysis Center GRADLCI.

The implementation serves as part of a blueprint of a global analysis and data centre for astroparticle physics. Her work was implemented within an international collaboration of a German-Russian Data Life Cycle Initiative project. The Russian Science Foundation Grant No 18-41-06003 on Russian side and the Helmholtz Society Grant No HRSF-0027 on German side supported the project.

Yours sincerely,

(Andreas Haungs)	(Jürgen Wochele)	(Doris Wochele)
Chairman	Member	Member

Рисунок В.3 — Акт о внедрении результатов исследования в международном проекте GRADLCI.