

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М.В. ЛОМОНОСОВА

ФИЛИАЛ МГУ В ГОРОДЕ СЕВАСТОПОЛЕ

Направление подготовки «Прикладная математика и информатика»
01.03.02 (бакалавр)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**АВТОМАТИЗИРОВАННОЕ ПЛАНИРОВАНИЕ
НА ОСНОВЕ АДАПТИВНОГО АЛГОРИТМА**

Выполнила: Заричная Екатерина Викторовна
студентка учебной группы ПМ-401

Научный руководитель: ст. преподаватель
кафедры алгоритмических языков
Баева Наталия Валерьевна

Севастополь – 2024

Содержание

Аннотация	3
Введение.....	4
Раздел I Постановка задачи.....	9
Раздел II Обзор существующих решений рассматриваемой задачи.....	11
2.1 LeaderTask.....	11
2.2 Битрикс24.....	13
2.3 Google Tasks.....	14
2.4 Типовые календарные планировщики	16
2.5 Анализ рассмотренных решений.....	17
Раздел III Исследование и построение решения задачи.....	19
3.1 Формализация задачи планирования	19
3.1.1 Исходные данные	19
3.1.2 Ограничения	20
3.1.3 Целевая функция	21
3.2 Критерии выбора порядка планирования событий на основе вариативности и приоритета	22
3.2.1 Вариативность	22
3.2.2 Оценка сложности планирования.....	23
3.3 Теоретические основы для адаптивного алгоритма.....	24
3.4 Адаптивный алгоритм планирования	25
3.5 Анализ разрешимости задачи планирования	27
3.5.1 Конфликт ресурсов	27
3.5.2 Критерии разрешимости.....	28
3.5.3 Эвристический характер алгоритма	28
Раздел IV Описание практической части.....	29
4.1 Вычислительный модуль.....	29
4.1.1 Глобальные переменные окружения.....	29
4.1.2 Динамическая матрица временной доступности	31
4.1.3 Координатор временных ресурсов	35
4.2 Модуль взаимодействия с пользователем	42
Заключение	47
Список источников	50
Приложение А Результат проверки на уникальность	1
Приложение Б Архитектура приложения.....	4

Аннотация

В выпускной квалификационной работе разработан программный инструмент для автоматизированного планирования на основе адаптивного алгоритма, способного формировать расписание на основе указанных пользователем ограничений и приоритетов.

Метод основан на математической модели, описывающей расписание через матрицу доступности и условия корректности расписания. Адаптивный алгоритм преобразования матрицы доступности сочетает эвристические стратегии с динамической обратной связью, обеспечивающей адаптацию к изменениям во входных данных.

Разработка программного обеспечения для решения задачи автоматизации планирования и составления расписания производилась на языке Python с использованием стандартных библиотек Numpy и средства графического отображения окон Tkinter.

Ключевые слова: адаптивный алгоритм, автоматизированное планирование, управление временными ресурсами, метрика вариативности, матрица доступности.

Введение

Планирование — это процесс разработки стратегии и тактики достижения целей, а также программы их реализации, в том числе определение задач, расстановка приоритетов, распределение затрат, объема и графика работ. При планировании происходит распределение ресурсов, которые необходимо затратить на достижение поставленных целей.

Планирование времени, в свою очередь, представляет собой дисциплину, являющуюся подразделением логистики и специализированную на установлении соответствия между этапами реализации стратегии и оптимальными для них хронологическими рамками. В этой области производится определение последовательности и приоритетности действий для повышения общей эффективности и успешности реализации поставленных целей. Для этого необходимо рассматривать её как обширную междисциплинарную сферу, которая включает в себя элементы прикладной математики, информатики, теории управления, психологии и социологии.

Исследования и разработки в этом направлении нацелены на повышение эффективности использования времени, требующегося для выполнения задач и достижения целей; минимизацию конфликтов и оптимизацию процессов в различных контекстах: от управления личным временем до крупных системных задач, таких как планирование производства, распределение ресурсов в больницах или составление учебных расписаний.

Современный ритм жизни и стремительное развитие технологий в целом предъявляют новые требования к организации временных ресурсов [3]. В условиях глобального увеличения объема обрабатываемой информации, возрастающего числа задач и усиленной конкуренции как на профессиональной, так и на личной арене, способность эффективно управлять временем становится неотъемлемым элементом жизнедеятельности.

Эффективное управление временем предполагает балансировку между стратегическими и оперативными целями, минимизацию затрат ресурсов на

рутинные задачи и быструю адаптацию к изменениям. Однако традиционные методы планирования, такие как статические расписания или ручное распределение задач, могут оказываться недостаточно гибкими в условиях изменчивой внешней среды и индивидуальных потребностей пользователя.

При постоянном нарастании сложности социальных, экономических и технологических процессов высокую значимость приобретают инструменты автоматизации, которые могли бы предоставить возможность перераспределения временных ресурсов на основе анализа данных и динамической оптимизации без необходимости ручной обработки информации пользователем. Таким образом, разработка инструментов, способных автоматизировать процессы планирования и оптимизации времени, приобретает выраженную актуальность.

Основной задачей может стать разработка таких решений, способных учитывать сложную совокупность факторов, включая изменяющиеся условия, индивидуальные предпочтения и внешние ограничения. Это позволит снизить когнитивную нагрузку на пользователя, минимизировать риски ошибок при планировании и повысить общую эффективность использования времени. Такие технологии в дальнейшем могут быть интегрированы в системы персональных помощников или корпоративные платформы управления проектами, что в свою очередь может способствовать освобождению ресурсов для улучшения производительности и повышения качества жизни.

Автоматизация планирования и составления расписаний представляет собой объединение математических, алгоритмических и программных методологий, разработанных для решения задач оптимального распределения ресурсов, времени и задач [17]. Эти решения применимы в различных контекстах, включая образовательные учреждения, логистику, медицинские учреждения, авиакомпании, промышленность и другие области. В условиях усложнения социальных, экономических и технологических процессов особое значение приобретают средства автоматизации, позволяющие перераспределять время и ресурсы на основе анализа доступных данных.

Достаточно часто встречаются научные работы, рассматривающие задачу автоматизированного распределения времени именно в контексте формирования расписания занятий или экзаменов в учебном заведении [4, 8, 13]. По сложности эта задача является NP-полной, то есть для её решения необходимо привлекать эвристические методы [4], такие как: поиск с запретами, муравьиные алгоритмы, генетические алгоритмы, раскраска графа, расширенная модификация метода «великого» потопа [19] и прочие. Можно заметить, что для такого случая математическая модель является детерминированной и статической, поскольку все данные, на основе которых необходимо произвести формирование расписания, известны заранее и не изменяются в процессе его составления. Ещё одной отличительной чертой таких реализаций автоматизации составления расписания является тот факт, что каждое из событий, которое должно быть запланировано, имеет всегда одинаковую и совпадающую со всеми остальными событиями длительность. Так, в случае составления расписания школьных занятий, каждый урок всегда длится ровно 45 минут. Фиксированная длительность планируемых событий серьёзным образом ограничивает возможность обобщения реализаций автоматизации для задач, являющихся более контингентными.

Теоретическая значимость данной работы заключается в использовании не применявшегося ранее адаптивного алгоритма для эффективного распределения временных ресурсов в условиях, когда входные и выходные требования к процессу могут изменяться, — это позволяет рассматривать проблему управления ресурсами в более сложных и динамических сценариях, характерных для современных систем. Применение таких алгоритмов способствует развитию методов моделирования и оптимизации процессов, основанных на изменяющихся данных, и вносит вклад в расширение научных представлений о работе систем с высокой степенью неопределенности. Кроме того, адаптивные подходы позволяют глубже изучить механизмы принятия решений в ситуациях, когда условия изменяются в реальном времени, что

имеет важное значение для теории системного анализа, математической оптимизации и искусственного интеллекта.

Несмотря на относительно беспрепятственную доступность вычислительных и программных ресурсов, автоматизированные подходы к составлению расписания на настоящий день не получили повсеместного распространения или даже выраженной тенденции к внедрению. Локальные, часто некоммерческие решения обычно используются в ограниченных областях и оказываются привязаны не только к проекту или предприятию, в рамках которого они были разработаны, но и к конкретным специалистам, работавшим над их созданием. Это нередко приводит к тому, что процесс перепрофилирования такого решения оказывается нецелесообразно затруднён, и его воспроизводимость оказывается бесперспективной. Так, например, система полуавтоматизированного составления расписания для учебного заведения «Аэрокосмический колледж имени М. Ф. Решетнева» [8] не может быть добровольно или на коммерческой основе передана и переназначена для нужд образовательного учреждения «Московский государственный университет имени М. В. Ломоносова».

С другой стороны, по ряду объективных и субъективных факторов пользователи зачастую принимают осознанное решение воздержаться от использования указанных систем. Несмотря на наличие в некотором количестве программных решений для управления проектной деятельностью функционала, предназначенного для планирования временных ресурсов, на практике сотрудники предприятий нередко отдают предпочтение ручному формированию индивидуальных графиков, обходясь без использования доступных им программных средств.

Практическая значимость исследования заключается в том, что разработка и внедрение предложенного адаптивного алгоритма создают предпосылки для значительного улучшения процессов планирования личного времени пользователя: алгоритм позволяет формировать быстродействующий

подход к управлению индивидуальными задачами, минимизировать временные потери и повысить общую продуктивность.

В перспективе алгоритм сможет анализировать и предлагать оптимальные временные слоты для выполнения задач, учитывая предпочтения, усталость и даже прогнозируемую эффективность пользователя в разное время суток.

Раздел I

Постановка задачи

Цель работы — проектирование и реализация программного инструмента с использованием адаптивного алгоритма, способного производить автоматизированное составление расписания личного времени с учётом расставленных пользователем приоритетов и ограничений.

Требования к программному инструменту:

автоматизация планирования — инструмент должен автоматически формировать расписание на основе введённых пользователем данных и минимизировать ручное вмешательство при разрешении конфликтов;

гибкость — алгоритм должен учитывать динамически изменяющиеся данные, таких как добавление новых событий или изменение приоритетов;

учёт приоритетов — инструмент должен учитывать приоритеты событий, заданные пользователем, и обеспечивать их включение в расписание в первую очередь;

разрешение конфликтов — алгоритм должен автоматически выявлять и разрешать конфликты (пересечение временных интервалов);

Объект исследования: процесс составления расписания, планирование.

Предмет исследования: автоматизация процессов составления расписания с учетом динамически изменяющихся данных и пользовательских приоритетов.

Достижение указанной цели выпускной квалификационной работы предполагало решение следующих задач:

- изучить и проанализировать существующие программы и приложения, которые могли бы решать задачу автоматизации планирования личного времени;

- разработать и описать формализованную математическую модель задачи планирования, в рамках которой выделен набор варьируемых параметров, используемых для поиска оптимального решения;
- установить целевую функцию, характеризующую полезность и оптимальность решения;
- определить ряд ограничений, которым обязано удовлетворять искомое решение;
- разработать адаптивный алгоритм расстановки событий в рамках математической модели задачи планирования, который обеспечит минимизацию целевой функции при учёте заданных ограничений;
- разработать и реализовать структуру данных для хранения и управления элементами математической модели задачи планирования в памяти вычислительной машины;
- произвести программную реализацию разработанного алгоритма, обеспечивающую выполнение расчётов и формирование расписания;
- реализовать пользовательский интерфейс для ввода исходных данных, отображения результатов и управления параметрами;
- протестировать созданный программный инструмент и оценить его на соответствие техническим требованиям, быстродействие, стабильность, выявить возможные уязвимости;
- оценить применимость полученного решения к другим задачам, связанным с предметной областью.

Раздел II

Обзор существующих решений рассматриваемой задачи

На сегодняшний день существует множество приложений и программ, которые предлагаются в качестве решений для планирования личного времени. Это разнообразие обусловлено стремлением разработчиков учитывать широкий спектр потребностей пользователей, от профессиональных задач до организации повседневной рутины. Проведение анализа таких приложений представляет собой важный этап в исследовании их эффективности, так как позволяет выявить степень их соответствия ожиданиям различных категорий пользователей, определить сильные и слабые стороны, а также оценить их влияние на повышение продуктивности и оптимизацию временных ресурсов. Изучение этих инструментов нуждается в том, чтобы учесть некоторые ключевые аспекты: функциональные возможности, удобство и логичность интерфейса, адаптивность к различным сценариям использования, а также их гибкость и универсальность [3, 4].

2.1 LeaderTask

Программное обеспечение для планирования и ведения проектной деятельности как отдельным пользователем, так и коллективом предприятия.

Оно предоставляется и поддерживается на различных платформах, в частности: Windows и MacOS — для компьютерных систем, а также Android и IOS — для мобильных устройств. Может быть установлено в качестве приложения, но имеет урезанную версию для работы с помощью веб-браузера. С применением серверных технологий и ведения учётных записей организована синхронизация расписания между устройствами пользователя.

Интерфейс программы LeaderTask реализован в виде комбинации календаря и менеджера проектов и целей.

Основная целевая аудитория этого программного обеспечения — это учреждения, которым необходимо организовывать работу своих сотрудников,

устанавливать для них единые сроки исполнения и формулировки задач. Для совместных проектов назначается ответственное лицо — так называемый лидер проекта — имеющий полномочия создавать, редактировать и модифицировать содержимое задач, их даты и другие параметры.

Благодаря особенностям целевой аудитории и специфике решаемой проблемы, данное программное обеспечение предлагает в основном функции, наиболее подходящие для работы небольших фирм или предприятий. По своему желанию пользователь может использовать это приложение для планирования личных целей или проектов, при этом сама задача распределения времени полностью возлагается на человека.

Это позволяет пользователю эффективно распределять время, назначать задачи и контролировать выполнение. Система разработки задач и проектов с возможностью назначения ответственного лица (лидера проекта) и возможности редактирования и модификации задач делает приложение удобным для работы в команде, особенно в малых организациях.

Есть инструменты отслеживания соблюдения сроков, но никаким образом не контролируются конфликты в построенных планах.

Так, LeaderTask подходит для планирования и организации работы, если проектная деятельность не слишком сложна. Оно помогает координировать работу команды, контролировать сроки, но в то же время не предоставляет глубокой аналитики или инструментов для управления более сложными проектами, а инструментов для автоматизации не предоставляется.

С точки зрения соответствия поставленным требованиям к программному инструменту: отсутствует автоматизация планирования (пользователь распределяет задачи вручную) — следовательно, ограничена гибкость и конфликты требуют ручного устранения. LedaerTask не предусматривает механизмов автоматической перестройки расписания при изменении исходных данных. Есть возможность выставить приоритет задачи.

2.2 Битрикс24

Битрикс24 — это интерактивная информационная система на базе веб-интерфейса, в которую включено множество различных инструментов для осуществления деятельности компании и управления бизнес-процессами. Она включает в себя набор различных модулей, одним из которых является модуль планирования времени и организации совместной работы.

В значительной степени его функционал ориентирован на решение типовых задач, аналогичных таким, которые обычно вынуждена решать целевая аудитория упомянутого выше приложения LeaderTask. Информационная система Битрикс24, в большей степени адаптирована для работы с крупными компаниями и не имеет бесплатной лицензии.

Планировщик событий в Битрикс24 — одно из немногих приложений, которое обладает критерием, позволяющим ему выявлять конфликтующие события. Такой критерий является единственным и применяется исключительно к типу события «собрание», что ограничивает сферу его действия и делает функционал системы частичным в решении проблем с конфликтами событий в более общем контексте.

Для планирования собраний характерно выделение переговорной комнаты. Алгоритмы приложения исключают возможность бронирования одной и той же переговорной комнаты на пересекающиеся временные интервалы. В случае возникновения такого конфликта пользователь получает уведомление в виде всплывающего окна с ошибкой и предложением выбрать другое время вручную. Таким образом, система на этом уровне предлагает решение проблемы конфликтов, однако она ограничена по своей эффективности, так как всю ответственность за поиск нового времени и принятие решения по организации работы возлагает на пользователя.

Этот подход позволяет обеспечить функционал для предотвращения наиболее очевидных конфликтов, однако оставляет за пользователем важную роль в принятии решения о разрешении проблемы. Это означает, что Битрикс24 не предоставляет полноценно автоматизированной системы для

управления всем расписанием, и большая часть задач по разрешению конфликтов остается на усмотрение пользователя. Такой подход имеет свои плюсы и минусы. С одной стороны, это позволяет сохранить гибкость в управлении временем, так как пользователь может самостоятельно принимать решения, ориентируясь на конкретные условия. С другой стороны, это требует от пользователя дополнительных усилий и навыков, чтобы эффективно использовать систему и избежать ошибок, связанных с перепланированием.

Анализ с точки зрения соответствия требованиям к программному инструменту показал, что автоматизация планирования в данной системе не предусмотрена. Битрикс24 способен выявлять коллизии для узкого класса событий, однако производить их устранение необходимо вручную. Изменения в расписании не приводят к автоматическому пересчету всего плана, то есть гибкость ограничена и требует участия пользователя. Не предоставлен функционал для учёта приоритетов.

Таким образом, функционал модуля планирования информационной системы Битрикс24 может рассматриваться как частично эффективный инструмент для организации рабочего времени.

2.3 Google Tasks

Google Tasks является интеграцией в интерактивную систему Google Calendar, которая в свою очередь является подразделением рабочего интернет-пространства Google Workspace. В противоположность вышеупомянутым решениям, этот сервис ориентирован на индивидуальное пользование.

Одной из главных отличительных черт сервиса Google Tasks является его надёжная совместимость с другими продуктами компании Google. В частности, на мобильных устройствах, на которых в качестве стандартных системных инструментов используется соответствующее программное обеспечение, обуславливается бесшовная согласованность и синхронизация с экосистемой Google. Для задач планирования это означает, что пользователь может в режиме реального времени получать обновления, отслеживать

изменения и управлять своими событиями и задачами на своём мобильном устройстве. Такая синхронизация способствует повышению производительности, так как оказывается устранена необходимость в дублировании информации и обеспечивается доступ к актуальным данным на всех устройствах пользователя. Так, встреча, добавленная на смартфоне, автоматически синхронизируется с другими устройствами, подключёнными к личному аккаунту Google. Использование облачных технологий гарантирует защиту данных от потери, поскольку они сохраняются в распределённых центрах обработки данных, что позволяет восстановить информацию в случае утраты устройства. Однако такой подход требует постоянного доступа в интернет, расходует заряд аккумулятора и может привести к нежелательному дублированию данных при недостаточном опыте использования.

Ещё одной характерной особенностью данного сервиса планирования времени является возможность создания «слотов встреч» в личном календаре, которые могут быть забронированы другими пользователями средствами системы. «Слоты встреч» определяются промежутком времени от его начала до окончания и длительностью одной встречи. При этом промежуток автоматически разбивается на такое количество встреч, которое можно разместить в пределах заданного интервала.

Анализ интеграции на соответствие требованиям к программному инструменту продемонстрировал, что Google Tasks демонстрирует низкий уровень автоматизации, выраженный в формировании некоторых повторяемых элементов расписания (бронирование встреч) без участия пользователя и предоставляет инструмент предотвращения наложения встреч друг на друга, при этом другие события могут накладываться на существующие встречи и это не отслеживается. Такой функционал ограничен рамками единственного типа событий и не предоставляет возможностей для автоматизированного составления расписания и разрешения конфликтов и обладает ограниченной гибкостью. Нет учёта приоритетов.

2.4 Типовые календарные планировщики

Типовые календарные планировщики представляют собой простые инструменты для управления временем, которые фокусируются на функции отображении календаря и записи событий с привязкой к времени. Эти приложения не имеют алгоритмов для автоматизации процессов, интеграции с другими сервисами или дополнительных функций для более сложного планирования. Они служат в первую очередь для отображения календарных событий и обеспечения их записи, что делает их подходящими для пользователей, которым не требуются дополнительные функциональности.

Основная особенность таких планировщиков — минимализм и простота функционала, как в решениях, подобных описанным в работе [7]. В отличие от более сложных решений, которые включают в себя интеграцию с электронной почтой, задачами, командной работой или аналитикой, типовые календарные планировщики обеспечивают только базовый интерфейс для ввода и отображения событий. Пользователи могут просто записывать встречи, задания или другие важные события на конкретные даты и время, при этом не ожидается какого-либо глубокого взаимодействия с другими инструментами или сервисами. Эти приложения обычно не предоставляют дополнительных функций для оптимизации работы пользователя, таких как напоминания, автоматическое обновление или управление задачами.

Такой подход имеет свои преимущества. Простота использования и отсутствие лишних настроек позволяют пользователям сосредоточиться исключительно на записи событий и их отображении, без лишних сложностей. Планировщик может быть полезен для тех, кому не нужно автоматизировать процессы планирования, а достаточно просто иметь календарь для записи основных событий. Например, такие приложения могут быть полезны для индивидуальных пользователей, которым достаточно видеть, какие события запланированы на определённые дни.

С точки зрения соответствия поставленным требованиям, такие планировщики не удовлетворяют ни одному из перечисленных требований:

нет автоматизации планирования, нет выявления конфликтов, полностью отсутствует гибкость и учёт приоритетов.

2.5 Анализ рассмотренных решений

Проведённый анализ современных систем планирования выявил ряд фундаментальных ограничений, препятствующих их эффективному применению в условиях динамически изменяющихся данных.

Ключевой проблемой является отсутствие механизмов автоматизированного разрешения конфликтов, что полностью возлагает ответственность за оптимизацию расписания на пользователя.

Например, в Битрикс24 предотвращение пересечений событий ограничено сценариями бронирования переговорных комнат. Кроме того, рассмотренные решения опираются на статические модели данных.

Эти ограничения обусловлены ориентацией существующих систем на сценарии с фиксированными параметрами, что противоречит требованиям современных приложений, где длительность, доступность и значимость событий могут варьироваться. Например, в образовательных учреждениях или логистических системах необходимость частой корректировки расписания делает ручное управление неэффективным. Таким образом, возникает потребность в алгоритмах, способных не только автоматизировать формирование расписания, но и перестраивать его с учётом новых условий.

Выявленные проблемы стали основой для разработки адаптивного подхода, представленного в данной работе. В отличие от аналогов, предложенный алгоритм интегрирует два ключевых критерия — вариативность и приоритет событий, — что позволяет ранжировать задачи по степени гибкости их размещения и важности. Это обеспечивает превентивное разрешение конфликтов за счёт приоритетного планирования событий с низкой вариативностью, минимизируя риски их исключения из расписания. Кроме того, введение динамической обратной связи (обновление матрицы

доступности после каждой итерации) устраняет жёсткую привязку к исходным данным, характерную для статических моделей.

Перечисленные выводы создают теоретический фундамент для перехода к формализации задачи планирования, которая рассмотрена в разделе III. Предложенные метрики и условия корректности расписания позволяют преобразовать качественные требования (устранение конфликтов, учёт приоритетов, гибкость) в математическую модель, пригодную для алгоритмической обработки. Это обеспечивает преемственность между критическим анализом существующих решений и разработкой нового метода, объединяющего эвристические стратегии с адаптивными механизмами.

Раздел III

Исследование и построение решения задачи

В данном разделе представлена новая математическая модель, разработанная специально для решения задачи автоматизированного планирования событий с учетом временных ограничений, приоритетов и конфликтов ресурсов. Модель базируется на дискретном представлении времени, множестве событий с заданными характеристиками и матрице доступности, отражающей допустимые временные интервалы для каждого события. Разработан оригинальный адаптивный алгоритм, основанный на новых метриках вариативности и оценки сложности планирования для формирования расписания с разрешением конфликтов, стремящийся обеспечить включение в расписание как можно большего числа событий.

3.1 Формализация задачи планирования

Задача автоматизированного планирования событий формализуется как проблема оптимального распределения ограниченных временных ресурсов с учётом множества условий и ограничений [4, 8]. В данном разделе представлены исходные данные и условия корректности, необходимые для построения математической модели, которая обеспечивает формирование расписания, удовлетворяющего заданным требованиям.

3.1.1 Исходные данные

Основой модели служит концепция дискретного временного горизонта, который представляет собой упорядоченное множество $T = \{t_0, t_1, \dots, t_{n-1}\}$ из n временных промежутков фиксированной длительности. Каждый промежуток t_j соответствует определенному интервалу времени в календарном плане.

Множество событий $E = \{e_1, e_2, \dots, e_m\}$ включает в себя m различных событий, каждое из которых характеризуется:

- Длительностью $d_i \in \mathbb{N}$, выраженной в количестве временных промежутков;
- Приоритетом $p_i \in \{1, 2, 3, 4, 5\}$, который характеризует степень важности включения события e_i в расписание;
- Вектором доступности $a_i = (a_{i1}, a_{i2}, \dots, a_{in})$, где $a_{ij} \in \{0, 1\}$;

Матрица доступности M размерности $m \times n$ объединяет векторы доступности всех событий и служит основным инструментом для анализа возможных вариантов планирования. Значение элемента матрицы $M_{ij} = 1$ указывает на то, что событие e_i в принципе может происходить во временной промежуток t_j , а значение $M_{ij} = 0$ — что событие e_i не может произойти во временной промежуток t_j .

3.1.2 Ограничения

Корректное расписание должно удовлетворять определённому набору ограничений, которые обеспечивают логическую целостность расписания.

Можно заметить, что временной промежуток t_j соответствует только одному событию тогда и только тогда, когда выполняется условие (3.1):

$$\sum_{i=1}^m a_{ij} = 1, \quad (3.1)$$

Для составления корректного расписания необходимо, чтобы в каждый момент времени t_j выполнялось не более одного события. Отсюда формулируется первое условие для составления расписания (3.2):

$$\forall j (0 \leq j < n): \sum_{i=1}^m a_{ij} \leq 1, \quad (3.2)$$

То есть, если $\sum_{i=1}^m a_{ij} = 0$, то промежуток t_j остаётся незанятым, а если $\sum_{i=1}^m a_{ij} = 1$, то на промежуток t_j назначено ровно одно событие.

В терминах матрицы фиксация события e_i в расписании означает выбор такой последовательности подряд идущих единиц, что длина этой последовательности равна длительности d_i , то есть должно выполняться условие (3.3):

$$\forall i (1 \leq i \leq m): \sum_{j=0}^{n-1} a_{ij} = d_i, \quad (3.3)$$

При этом единицы в строке a_i должны образовывать непрерывный блок. Формально для события e_i это требование можно сформулировать в виде набора условий (3.4):

$$\begin{cases} \forall j \in [j_{start}, j_{end}]: a_{ij} = 1 \\ \forall j (j < j_{start} \text{ или } j > j_{end}): a_{ij} = 0, \\ j_{end} - j_{start} + 1 = d_i \end{cases} \quad (3.4)$$

где j_{start}, j_{end} — границы выбранной последовательности единиц.

Для получения расписания необходимо исходную матрицу M преобразовать в матрицу S , для которой выполнены условия (3.2), (3.3) и (3.4).

3.1.3 Целевая функция

В рамках математической модели задачи автоматизированного планирования целевая функция формализует критерий оптимальности, направленный на минимизацию количества событий, исключённых из

расписания в ходе преобразования матрицы доступности. Данная функция отражает ключевое требование к составляемому расписанию: обеспечить включение как можно большего числа событий в расписание при соблюдении заданных ограничений.

Событие оказывается утерянным в ходе расстановки тогда, когда его вектор доступности полностью состоит из нулей, то есть в матрице доступности для события e_i выполняется условие (3.5):

$$\text{для } e_i: \sum_{j=0}^n a_{ij} = 0 \quad (3.5)$$

Целевая функция определяется как сумма всех индикаторных переменных для событий по формуле (3.6):

$$F = \sum x_i, \text{ где } x_i = f(x) = \begin{cases} 1, & \text{если для } e_i \text{ выполнено (3.5)} \\ 0, & \text{в противном случае} \end{cases} \quad (3.6)$$

3.2 Критерии выбора порядка планирования событий на основе вариативности и приоритета

При автоматизированном планировании событий с учётом временных ограничений одной из подзадач является определение оптимального порядка их размещения в расписании. В данном разделе вводятся количественные метрики, которые позволяют формализовать процесс принятия решения об очередности расстановки событий.

3.2.1 Вариативность

Для формализации гибкости размещения события в расписании вводится метрика вариативности V_i , определяемая по формуле (3.7)

$$V_i = \frac{\sum_{j=0}^{n-1} M_{ij}}{d_i}, \quad (3.7)$$

Где $\sum_{j=0}^{n-1} M_{ij}$ — суммарная длина всех доступных временных промежутков для события e_i , а d_i — это его длительность.

Данная величина характеризует, сколькими способами можно выбрать подходящий временной интервал для события: чем выше значение V_i , тем больше существует вариантов его размещения.

Поскольку в процессе планирования доступные наборы временных слотов могут перекрывать друг друга, а некоторые события могут оказаться исключены из расписания из-за конфликтов, целесообразно в первую очередь размещать события с наименьшей вариативностью и наибольшим приоритетом.

Это обусловлено следующими причинами: события с низкой вариативностью (V_i) имеют ограниченное количество вариантов размещения, поэтому их раннее включение в расписание снижает риск невозможности их планирования на более поздних этапах. События с высоким приоритетом (p_i) должны быть гарантированно включены в расписание, даже если это потребует исключения менее значимых событий.

3.2.2 Оценка сложности планирования

Для формализации данного подхода вводится функция метрики D_i , объединяющая оба критерия и выражаемая формулой (3.8):

$$D_i = V_i * w_V - p_i * w_p, \quad (3.8)$$

Где w_V — вес вариативности, w_p — вес приоритета.

Данная функция построена таким образом, что при увеличении V_i метрика D_i возрастает (что снижает приоритетность события), а при увеличении p_i метрика D_i убывает (что повышает приоритетность события).

Для обеспечения корректного баланса между вариативностью и приоритетом необходимо выбрать веса w_v и w_p таким образом, чтобы: события с низкой вариативностью, но низким приоритетом не получали преимущества перед событиями с высокой вариативностью и высоким приоритетом.

Эмпирически установлено, что при $w_v = 1, w_p = n + 1$, где n — общее количество временных слотов, достигается требуемый баланс.

3.3 Теоретические основы для адаптивного алгоритма.

Адаптивный алгоритм — это тип алгоритма, который динамически изменяет своё поведение по ходу своего исполнения, основываясь на доступной информации и входных данных. Такой алгоритм не полагается на заранее заданные настройки, а постоянно корректирует свой ход решения, чтобы справляться с поставленной задачей. Некоторые алгоритмы могут учитывать ошибки предыдущих вычислений и применять предварительно определённую систему вознаграждений. Адаптивный алгоритм использует не только статические фиксированные параметры, но и задействует механизм обратной связи для оптимизации работы во время исполнения. Примеры таких алгоритмов можно встретить в системах управления, машинном обучении, цифровой обработке сигналов и других областях [2, 12, 14].

Составляющие алгоритма, из которых обычно следует его адаптивность — это обратная связь и динамическое обновление параметров [18]. Обратная связь предоставляет алгоритму возможность регулярно получать информацию о качестве своих текущих решений или о характеристиках входных данных. Эта информация служит основой для внесения корректировок в работу алгоритма. Динамическое обновление параметров позволяет учитывать текущие изменения в условиях работы и подстраивать стратегию решения. Такая архитектура обеспечивает устойчивость алгоритма к нестационарным данным и изменяющимся требованиям. Кроме того, сочетание этих

механизмов способствует балансу между стабильностью и пластичностью реализации, предотвращая как избыточную консервативность, так и неоправданную изменчивость в её поведении.

В задачах планирования входные параметры часто зависят от действий пользователя, чьи предпочтения, ограничения или доступные ресурсы априорно неизвестны и могут произвольно изменяться. Например, пользователь может корректировать цели, вводить новые условия или пересматривать приоритеты. Адаптивный алгоритм хорошо подходит для автоматизации решения задач планирования, потому что он может подстраиваться под изменения ограничений и доступных ресурсов. Его гибкость позволяет обновлять параметры, основываясь на предыдущих решениях. Он устойчив к неопределенности и эффективно использует обратную связь для оптимизации использования ресурсов.

3.4 Адаптивный алгоритм планирования

Запланировать событие означает установить однозначное соответствие между этим событием и набором подряд идущих временных промежутков, в течение которого оно должно начаться, продолжаться и завершиться, а составить расписание — это найти такое размещение событий, что каждому временному промежутку соответствует не более одного события [8].

Адаптивность алгоритма обеспечивается обратной связью: после каждой итерации матрица M обновляется, что позволяет корректировать поведение по ходу исполнения и динамическим выбором стратегии: в зависимости от текущих значений метрики D_i и конфликтов, алгоритм перераспределяет ресурсы [9].

Механизм разрешения конфликтов

Предлагаемый алгоритм разрешения конфликтов реализует стратегию последовательного планирования с обратной связью, основанную на динамическом анализе состояния матрицы доступности M . Алгоритм работает

итеративно, на каждом шаге принимая решение о фиксации конкретного события в расписании на основе текущего состояния матрицы. Процесс выполнения алгоритма включает следующие этапы:

Этап 1: предварительная валидация матрицы

Перед началом каждой итерации выполняется проверка корректности всех последовательностей единиц в матрице. Последовательности, длина которых меньше длительности соответствующего события d_i , подлежат обнулению. Данная операция гарантирует, что в матрице остаются только потенциально возможные варианты размещения событий, удовлетворяющие условию непрерывности (3.4).

Этап 2: фиксация однозначно определённых событий.

Необходимо произвести поиск событий, у которых в строке оказалась последовательность единиц такая, что для неё уже выполнены условия (3.2) и (3.4). Выполнение условия (3.2) означает, что эта последовательность не имеет конфликтов с другими событиями, а значит это событие является единственным претендующим на этот набор слотов. Следовательно, при обнаружении такого события оптимальным решением является немедленная фиксация этого события в расписании. Для этого все элементы соответствующей строки, кроме выбранной последовательности, обнуляются, после чего алгоритм переходит к следующей итерации.

Нахождение события с минимальным значением метрики D_i в нефиксированной части матрицы.

Для всех событий в нефиксированной части матрицы по формуле ... рассчитывается нынешнее значение метрики и выбирается событие с наименьшим её значением. Это событие выносится в верхнюю нефиксированную строку.

Этап 3: фиксация выбранного события

Для выбранного события e_i . производится выбор произвольной допустимой последовательности временных слотов. Выбранная

последовательность временных слотов фиксируется в верхней доступной позиции матрицы. Это означает, что событие e_i закрепляется в расписании, а его временные интервалы исключаются из дальнейшего рассмотрения для других событий. Все элементы матрицы, находящиеся под выбранной последовательностью, также обнуляются. Таким образом, события, планируемые позднее, не могут занимать те же временные интервалы, что и e_i . После фиксации события алгоритм переходит к следующей итерации.

Этап 4: обработка тупиковых ситуаций

Если на какой-либо итерации для события e_i не остаётся ни одной допустимой последовательности ($\forall j M_{ij} = 0$), такой исход считается неудовлетворительным. Необходимо выполнить откат к самой первой точке принятия решения (“развилке”), выбрать альтернативный вариант размещения и продолжать работу с обновлённой матрицей.

3.5 Анализ разрешимости задачи планирования

Существование допустимого расписания существенно зависит от структуры исходной матрицы доступности M . В общем случае, при произвольных входных данных невозможно гарантировать существование расстановки, сохраняющей все события. Это обусловлено следующими фундаментальными ограничениями.

3.5.1 Конфликт ресурсов

В случае, когда несколько событий требуют одних и тех же временных слотов, сохранить все события невозможно. Произвольный пример такой ситуации представлен в матрице (3.9):

$$M = \begin{matrix} & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{matrix} \quad (3.9)$$

При $d_i = 3$ для всех событий:

Событие 1 требует слотов 0-2, событие 2 требует слотов 1-3, событие 3 требует слотов 2-4. При этом слот 2 необходим каждому из трёх событий. Так, одновременное включение более одного из них в расписание невозможно.

3.5.2 Критерии разрешимости

Для существования допустимого расписания необходимо выполнение следующих условий:

$$\forall i \exists j: \sum_{\{k=j\}}^{\{j+d_i-1\}} M[i][k] = d_i, \quad (3.10)$$

Условие (3.10) означает, что каждое событие имеет хотя бы одну допустимую последовательность.

$$\sum_{\{i=1\}}^m d_i \leq n, \quad (3.11)$$

Условие (3.11) означает, что общая длительность событий не превышает размерности временного горизонта.

3.5.3 Эвристический характер алгоритма

В условиях, когда полное сохранение всех событий невозможно, предложенный алгоритм старается максимизировать количество включённых событий, обеспечивает приоритетное включение событий с высоким значением p_i и минимизирует количество потерянных событий за счёт оптимального выбора порядка планирования.

Раздел IV

Описание практической части

В данном разделе рассматривается программная реализация предложенной математической модели с использованием алгоритма расстановки событий. Целью реализации является практическая проверка корректности модели, а также получение результатов, иллюстрирующих её поведение в различных сценариях. Для построения программы использован язык Python версии 3.11. Применялись стандартные средства библиотеки Python: NumPy для упрощения и оптимизации вычислений, random для генерации случайных данных и Tkinter для реализации оконного интерфейса взаимодействия с пользователем. Для реализации алгоритма использовались методы, описанные в работах по моделированию распределения задач [10, 16].

4.1 Вычислительный модуль

Вычислительный модуль спроектирован в соответствии с принципами объектно-ориентированного программирования (ООП). Основу архитектуры составляют два пользовательских класса, которые реализуют методы взаимодействия с элементами математической модели, и функциональная реализация логики адаптивного алгоритма.

4.1.1 Глобальные переменные окружения

Система использует набор глобальных переменных для управления процессом временного планирования. Все параметры инициализируются при запуске системы, а временные характеристики остаются постоянными в течение одного сеанса. Это позволяет гибко настраивать параметры работы программы при необходимости без модификации алгоритмической части, а также обеспечивает поддержку адекватного уровня абстракции.

Базовые временные параметры

timeslotLength — длительность одного временного слота, выраженная в минутах. По умолчанию имеет значение 30 минут.

numberOfDays — Количество суток в горизонте планирования N , по умолчанию 3 суток.

Производные временные характеристики

timeslotsInDay — расчётное количество временных слотов в сутках n_d , вычисляемое по формуле (4)

$$n_d = \frac{24 \text{ часа} * 60 \text{ минут}}{\tau}, \quad (4)$$

где τ — длительность одного временного слота (значение *timeslotLength*).

planningHorizon — общая длина горизонта планирования в слотах n , выраженная формулой (4.1).

$$n = N * n_d, \quad (4.1)$$

где N — количество суток в горизонте планирования, n_d — количество временных слотов в одних сутках.

Управление идентификацией событий

availableID — текущий доступный уникальный идентификационный номер ID .

Каждому инициализируемому событию присваивается уникальный идентификационный номер. В системе хранится счётчик для генерации номеров, автоматически инкрементируемый при добавлении новых событий в матрицу доступности.

4.1.2 Динамическая матрица временной доступности

Класс *timeMatrix* содержит единственное поле и хранит в себе бинарную матрицу доступности, а также предоставляет методы для взаимодействия с ней. Объект этого класса является основным хранилищем информации о системе и обеспечивает синхронизацию между алгоритмической логикой планирования и актуальным состоянием данных о времени.

Матрица доступности организована в виде специализированного двумерного массива данных, структура которого унаследована от базового класса `np.ndarray`, содержащегося в стандартной библиотеке Numpy языка Python. Класс `np.ndarray` оптимизирован для работы с многомерными структурами данных и позволяет минимизировать затраты памяти и вычислительных ресурсов. Он предоставляет базовые функции для работы с массивами, в том числе индексированный доступ к элементам (для обращения к временному слоту с номером j в строке с номером i).

Каждая строка двумерного массива соответствует вектору доступности одного события, а столбец — временному промежутку, причём длина строки фиксирована и завязана на значении параметра n — общей длины горизонта планирования, выраженной в слотах. Количество столбцов в матрице, в свою очередь, может изменяться, когда события оказываются добавлены или удалены пользователем.

В процессе работы алгоритма планирования осуществляется постоянная реорганизация матрицы доступности, что подразумевает, что строки могут произвольным образом менять своё положение в соответствии с приоритетами событий, текущими значениями метрики вариативности и результатами разрешения конфликтов. Следовательно, возникает необходимость в поддержании способа идентификации, который не зависит от положения строки в матрице. Для этого каждой строке матрицы статически присваивается уникальный идентификатор (ID) в момент инициализации события. С точки зрения программирования это означает присоединение к матрице слева дополнительного столбца, строго зарезервированного для уникальных

идентификаторов. Идентификационный столбец жестко привязан к соответствующим строкам и сохраняется при всех операциях. Также он не подвергается модификациям в ходе алгоритмических преобразований. Это гарантирует однозначную идентификацию события независимо от его текущей позиции в матрице.

Таким образом, длина строки матрицы становится $(n + 1)$, где n — общая длина горизонта планирования в слотах.

Для работы с матрицей был реализован набор специализированных методов. Для удобства и обеспечения прозрачного доступа к данным, все методы были разработаны таким образом, что внешние методы работают с матрицей так, как если бы она состояла только из временных слотов, тогда как внутренняя обработка учитывает служебный столбец с идентификационными номерами, то есть с соблюдением индексации от 0 до $n-1$.

Базовые операции модификации структуры

add_row(list) — добавление строки: вставка нового вектора доступности с автоматической генерацией уникального ID и расширением матрицы. Принимает в качестве аргумента список или вектор длины n . Возвращает новое состояние матрицы. Новые строки всегда добавляются в конец матрицы.

pop_row(i) — извлечение строки: удаление указанной строки с возвратом соответствующего ей идентификационного номера (ID). Принимает в качестве аргумента номер удаляемой строки. Возвращает новое состояние матрицы и идентификационный номер удалённого события. По умолчанию (при вызове метода без аргументов) извлекается последняя строка матрицы.

surface(i_{from}, i_{to}) — позиционное перемещение строки: перестановка строки на заданную позицию с сохранением целостности данных. Принимает в качестве аргумента номер строки, которую необходимо переместить и номер позиции, на которую необходимо поместить строку. Возвращает новое состояние матрицы. По умолчанию (при вызове метода без аргументов) строка переносится в верхнюю позицию матрицы.

Выборочное обнуление элементов

$nullify(i, j_1, j_2)$ — строковое обнуление: замена нулями элементов в заданном диапазоне конкретной строки. Принимает в качестве аргумента номер строки и пару индексов: индексы начала и окончания последовательности, которую необходимо обнулить в заданной строке. Возвращает новое состояние матрицы.

$nullifyRow(i, j_1, j_2)$ — обнуление под последовательностью: очистка указанных столбцов ниже определённой строки. Принимает в качестве аргумента номер строки и пару индексов: начала и окончания последовательности, под которой необходимо произвести обнуление элементов. Возвращает новое состояние матрицы.

$nullifyExcept(i, j_1, j_2)$ — инверсное обнуление: сохранение ненулевых значений только в указанных позициях строки. Принимает в качестве аргумента номер строки и пару индексов: начала и окончания последовательности, которую необходимо сохранить. Возвращает обновлённое состояние матрицы.

Аналитические операции

$sumRow(i)$ — подсчёт заполненности по строке: вычисление количества доступных слотов для конкретного события. Другими словами, метод возвращает значение выражения (4.2), что есть левая часть в формуле (3.3) и числитель в формуле (3.7).

$$\sum_{j=0}^{n-1} M_{ij}, \quad (4.2)$$

где M_{ij} — элемент матрицы в столбце j строки i .

В качестве аргумента метод принимает номер строки i . Возвращает вычисленное значение выражения (4.2).

sumCol(j) — подсчёт заполненности по столбцу: определение числа событий, претендующих на определённый временной слот. Другими словами, метод возвращает значение выражения (4.3), что есть левая часть в формулах (3.1) и (3.2).

$$\sum_{i=0}^{m-1} M_{ij}, \quad (4.3)$$

где M_{ij} — элемент матрицы в столбце j строки i .

В качестве аргумента метод принимает номер столбца j . Возвращает вычисленное значение выражения (4.3).

cleanUp(i, d) — фильтрация последовательностей: удаление всех последовательностей в указанной строке, длина которых меньше заданной. В качестве аргумента принимает номер строки i и длину последовательности d . Возвращает новое состояние матрицы.

longestStreak(i) — определение максимальной непрерывной последовательности доступных временных слотов в строке. В качестве аргумента принимает номер строки i . Возвращает длину максимальной непрерывной последовательности ненулевых элементов в этой строке.

Операции поиска

nearestNonZero(i, j) — нахождение ближайшего ненулевого элемента в строке: последовательно производится проверка на равенство нулю каждого элемента указанной строки начиная с заданного до тех пор, пока не будет встречен ненулевой элемент. Принимает в качестве аргумента номер строки и номер элемента, начиная с которого следует начинать поиск. Возвращает номер столбца, в котором был найден ненулевой элемент.

traceAFlow(j) — нахождение ближайшего ненулевого элемента в столбце: последовательно производится проверка на равенство нулю каждого элемента указанного столбца начиная с заданного до тех пор, пока не будет

встречен ненулевой элемент. Принимает в качестве аргумента номер столбца и номер строки, начиная с которой следует начинать поиск. Возвращает номер строки, в которой был найден ненулевой элемент.

scanForChronoHarmony() — выявление всех однозначно детерминированных временных интервалов в матрице. Для каждого столбца с применением метода подсчёта заполненности по столбцу проверяется соблюдение условия (3.1). В случае, когда столбец удовлетворяет этому условию, с помощью метода нахождения ближайшего (в данном случае — единственного) ненулевого элемента в столбце определяется номер строки, в котором он содержится. Выявленная пара (строка, столбец) добавляется в список в виде кортежа. Метод не принимает аргументов. Возвращает список всех найденных пар, позволяющих обратиться к элементам, для которых выполнено условие (3.1).

4.1.3 Координатор временных ресурсов

Класс *TimeVortex* представляет собой центральный управляющий компонент системы, ответственный за согласование работы матрицы доступности с алгоритмом планирования. Его архитектура сочетает методы управления ресурсами и адаптивную логику принятия решений.

Структура данных

Класс содержит следующие поля:

- *matrix* — экземпляр класса *timeMatrix* динамических матриц временной доступности.

Хранит актуальное состояние матрицы доступности.

По умолчанию инициализируется пустым (не содержащим ни одного события) объектом.

- *data* — вложенный словарь данных событий.

Ключом в словаре является уникальный идентификационный номер события. Значением является внутренний словарь унифицированной

структуры, содержащий поля с информацией о событии. Структура внутреннего словаря (ключ: значение):

- *duration*: длительность d_i события e_i , выраженная целым числом временных слотов и для которого выполняются ограничения $1 \leq d_i \leq n$, где n — общая длина горизонта планирования в слотах;
- *priority*: приоритет p_i события e_i , выражен целым числом от 1 до 5;
- *name*: строковый идентификатор или наименование события, используемое в интерфейсе и текстовом выводе.

Обеспечивает механизм доступа к данным о событии по его уникальному идентификационному номеру.

- *rowTimeLocked* — индекс последней зафиксированной строки.

Позволяет отслеживать границу между запланированными и ещё не распределёнными событиями. По умолчанию инициализирован значением -1, что означает, что ни одно событие ещё не было зафиксировано в расписании.

- *isVortexSettled* — флаг завершения планирования.

Принимает значения True или False, по умолчанию инициализирован значением False. Служит индикатором того, что все строки матрицы доступности, которые можно было зафиксировать, уже зафиксированы и дальнейшие вычисления над матрицей не требуются. Устанавливается в True в случаях, когда индекс последней зафиксированной строки достигает последней строки матрицы (все события спланированы) или когда не осталось ни одного допустимого варианта расстановки событий среди нераспределённых, что равносильно тому, что под последней зафиксированной строкой остались только полностью обнулившиеся вектора доступности.

Методы класса

`cleanUp()`: Процедура очистки матрицы от непригодных для размещения событий временных интервалов. Для каждой строки производится анализ непрерывных последовательностей ненулевых элементов в векторе доступности и обнуление тех последовательностей, длина которых оказалась меньше длительности соответствующего события. Длительность извлекается из словаря данных событий по уникальному идентификационному номеру, хранимому в служебном столбце матрицы. Метод не принимает аргументов и возвращает новое состояние координатора временных ресурсов.

`pop(i)`: Метод удаления события из системы. Выполняет удаление строки с заданным индексом из матрицы доступности с применением метода извлечения строки матрицы доступности и удаление соответствующей записи из словаря данных событий по уникальному номеру события. Если удаляемое событие находится в пределах уже зафиксированного диапазона, то значение индекса последней зафиксированной строки корректируется. В качестве аргумента метод принимает i — номер удаляемой строки. При вызове метода без аргументов удаляется событие в последней строке матрицы доступности. Возвращает новое состояние координатора временных ресурсов.

`add(vector, priority, duration, name)`: Метод создания события и добавления его в систему. Выполняет инициализацию события, присвоение ему уникального идентификационного номера и присоединение его к вектору доступности слева. Полученная строка вносится в конец матрицы доступности. В словаре данных событий создаётся новая запись, в которую вносится информация о событии: его длительность, приоритет и название. После инициализации вызывается процедура очистки `cleanUp()` для удаления непригодных для размещения событий временных интервалов. Метод возвращает новое состояние координатора временных ресурсов.

`density(i)`: Метод вычисления оценки D_i сложности планирования для события в заданной строке i , реализующий формулу (3.8). Для вычисления

значения метрики используется метод подсчёта заполненности по строке $\text{sumRow}(i)$ матрицы доступности и значения p_i и d_i из словаря данных событий, извлекаемые по уникальному идентификационному номеру события. В случае, когда значение $\text{sumRow}(i)$ равно нулю, что означает, что для данного события не осталось допустимых расстановок, значение метрики принимается заведомо большим для опускания таких строк в низ матрицы доступности. Принимает в качестве аргумента номер строки i . Возвращает значение метрики вариативности события, вектор доступности которого хранится в заданной строке.

$\text{sortByDensityBelowRow}(i)$: Метод сортировки строк матрицы доступности по убыванию значений оценки сложности планирования событий под заданной строкой. Для каждой строки, начиная со строки с индексом $i+1$ вычисляется значение (density), и выполняется упорядочивание строк по возрастанию значения оценок для того, чтобы события с наименьшим значением оценки оказались вверху свободной части матрицы. Метод принимает в качестве аргумента номер строки i и возвращает новое состояние матрицы доступности.

$\text{formVortexTrail}(i)$: Метод формирования списка возможных вариантов размещения события во временных слотах для заданной строки. По уникальному идентификационному номеру события извлекает значение его длительности и производит анализ состояния вектора доступности для выявления допустимых последовательностей, длина которых соответствует необходимой длительности. Каждая такая последовательность определяется номером её начального элемента. Метод принимает в качестве аргумента номер строки в матрице доступности. Метод возвращает список индексов начальных элементов последовательностей, которые могут быть присвоены событию в этой строке.

$\text{timeLock}(i,j)$: Метод фиксации события во временной матрице. Закрепляет событие в расписании и обновляет параметры координатора временных ресурсов в соответствии с произведёнными изменениями.

Закрепление события производится следующей последовательностью действий: обновление значения поля `rowTimeLocked`; перенос строки, соответствующей фиксируемому событию, с помощью метода `surface(...)` на позицию `rowTimeLocked`; обнуление всех элементов в строке за пределами фиксируемой последовательности с помощью метода `nullifyExcept(...)`; обнуление всех элементов в матрице под выбранной последовательностью с помощью метода `nullifyRow(...)`; очистка матрицы доступности от потерявших корректность последовательностей с помощью метода `cleanUp()`; упорядочивание строк матрицы под зафиксированной строкой с помощью метода `sortByDensityBelowRow(...)`; обращение в `True` значения поля *`isVortexSettled`* в случае, если был достигнут конец матрицы доступности.

`seekContinuumBalance(list)`: Метод поиска ближайшей однозначно детерминированной непрерывной последовательности среди существующих, которая может быть зафиксирована в расписании без конфликтов. В качестве аргумента принимает список кортежей вида (i, j) , где i — индекс строки матрицы доступности, соответствующей событию, j — индекс временного слота, где событие может быть размещено. Метод группирует входные данные по строкам матрицы доступности, для каждой строки проверяется наличие непрерывной последовательности, длина которой равна длительности события. Другими словами, проверяется выполнение условия (3.4). Метод возвращает первую найденную пару (i, j) , где i — номер строки, j — номер начала подходящей однозначно детерминированной последовательности.

`applyContinuumBalance()`: Метод фиксации всех однозначно детерминированных последовательностей, подходящих своим событиям, в матрице доступности. С помощью метода `scanForChronoHarmony()` осуществляется формирование списка кортежей, определяющих положение единственных в столбце ненулевых элементов для передачи в качестве аргумента методу `seekContinuumBalance()`. С его помощью последовательно выявляются подходящие для фиксации без конфликтов последовательности и с помощью метода `timeLock()` они фиксируются в расписании. В случае

отсутствия допустимых последовательностей, метод завершает выполнение и возвращает обновлённое состояние координатора временных ресурсов. Метод не принимает аргументов и возвращает обновлённое состояние координатора временных ресурсов.

`doomedThreads()`: Метод для определения количества событий, которые не попали в расписание в ходе расстановки событий. Производит подсчёт строк, полностью состоящих из нулей, в матрице доступности. Метод не принимает аргументов и возвращает число событий, которые не удалось включить в расписание.

`wobbleFit()`: Метод выбора произвольной подходящей последовательности для события в верхней строке свободной части матрицы доступности. Для строки с индексом `rowTimeLocked+1` вызывается метод `formVortexTrail(...)` для формирования списка возможных расстановок. Производится выбор произвольной расстановки и фиксация её в расписании с помощью метода `timeLock(...)`. Если все строки матрицы доступности обработаны, значение поля `isVortexSettled` обращается в `True`. Если список, сформированный методом `formVortexTrail(...)`, пуст — значит вектор доступности в строке пуст и дальнейшая расстановка событий невозможна. В таком случае `isVortexSettled` также обращается в `True`. Метод не принимает аргументов и возвращает обновлённое состояние координатора временных ресурсов.

`exploreTimeline()`: метод последовательной расстановки событий. С помощью методов `applyContinuumBalance()` и `wobbleFit()` фиксирует события в матрице, пока это возможно. Когда `isVortexSettled` обращается в `True`, метод завершает работу и возвращает обновлённое состояние координатора временных ресурсов, а также число не попавших в расписание событий, возвращённое методом `doomedThreads` для этого состояния.

Реализация адаптивного алгоритма планирования

Производится предварительная обработка: матрица доступности очищается от непригодных для размещения последовательностей с помощью метода `cleanUp()`

Алгоритм автоматизированной расстановки событий реализован функцией `TimeyWimeyPathfinder(vortex, depth)`. В качестве аргументов функция принимает экземпляр класса `TimeVortex` `vortex` — текущее состояние координатора временных ресурсов, и целое число `depth` — текущий уровень рекурсии, по умолчанию имеет значение 0. Алгоритм выполняет рекурсивный поиск оптимального расписания, его цель — предложить расстановку, содержащую все события, либо минимизировать количество незапланированных событий путём исследования различных способов фиксации событий в матрице доступности.

На первом этапе работы алгоритма производится проверка глубины рекурсии. Если глубина рекурсии превысила ограничения

На первом этапе работы алгоритма производится фиксация всех однозначно детерминированных событий, которые возможно расставить, с помощью метода `applyContinuumBalance()`.

Если дальнейшая расстановка невозможна, то есть поле `isVortexSettled` имеет значение `True`, функция возвращает текущее состояние координатора временных ресурсов. Иначе, расстановка событий продолжается.

Для следующего незафиксированного события, вектор доступности которого расположен в строке `rowTimeLocked+1`, формируется список допустимых расстановок. Список перемешивается случайным образом и варианты расстановки последовательно извлекаются из списка для исследования до тех пор, пока не будет найдено оптимальное расписание или пока список не опустеет.

Для рассматриваемой на данном шаге расстановки создаётся копия текущего состояния. Событие фиксируется в выбранной позиции, после чего с помощью метода `exploreTimeline()` производится расстановка событий до

конца матрицы доступности. Если все события были успешно расставлены и число строк, полностью состоящих из нулей, равно нулю, поиск прекращается, и функция возвращает состояние координатора временных ресурсов. Если есть события, которые не удалось включить в расписание, то производится сравнение с последним сохранённым кандидатом на расписание с наименьшим количеством утеранных событий. Если в новом расписании меньше утеранных событий, чем в последнем сохранённом, то оно становится новым кандидатом. Также на этом этапе сохраняется состояние с $\text{rowTimeLocked}+1$ фиксированных событий для отката к этой развилке.

Если список опустел до того, как было найдено оптимальное расписание, производится рекурсивный вызов функции `TimeyWimeyPathfinder` с кандидатом на расписание с наименьшим количеством утеранных событий в качестве аргумента и глубиной рекурсии $\text{depth}+1$.

Если глубина рекурсии вышла за пределы ограничения рекурсии, поиск прекращается и возвращается состояние координатора временных ресурсов, содержащее расписание с наименьшим количеством утеранных событий.

4.2 Модуль взаимодействия с пользователем

Модуль реализован с использованием библиотеки `Tkinter`, предназначенной для визуализации элементов системы и организации взаимодействия с пользователем. Графический интерфейс представлен в виде набора окон, каждое из которых отвечает за отдельный функциональный аспект работы программного инструмента. Такая архитектура обеспечивает интуитивно понятное управление, а также способствует логическому разделению операций ввода исходных данных, отображения промежуточных и итоговых результатов, и управления параметрами алгоритма.

Панель управления

Основное окно программы представлено на рисунке 1. Панель управления предоставляет базовые функции взаимодействия пользователя с

планировщиком: создание, просмотр и удаление событий. События визуализируются в виде так называемых карточек событий — цветных прямоугольников, содержащих ключевые характеристики: наименование, длительность и приоритет. Цвет карточки соответствует приоритету события, что обеспечивает визуальную дифференциацию элементов. Карточки располагаются в интерфейсе вертикально, в порядке их создания.

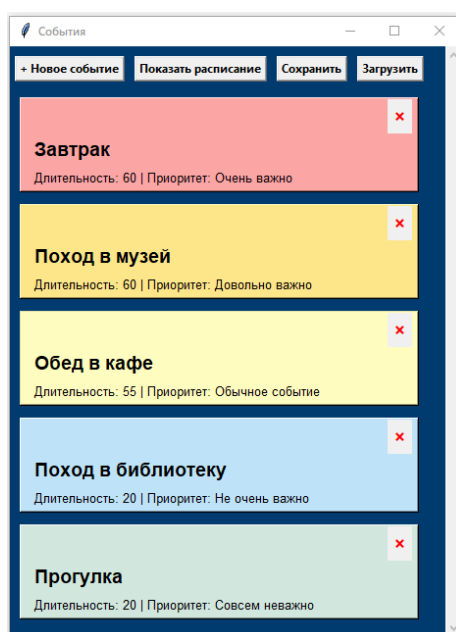


Рисунок 1 Панель управления пользовательского интерфейса

Удаление события осуществляется путём нажатия на пиктограмму с изображением красного креста, расположенную в правом верхнем углу карточки. После этого пользователю предлагается подтвердить действие; при подтверждении событие удаляется из списка.

Кнопка «+ Новое событие» инициирует создание нового элемента. Пользователю предлагается ввести параметры события в форме редактора.

Кнопка «Показать расписание» запускает реализацию адаптивного алгоритма составления расписания, результат которого отображается в соответствующем окне.

С помощью кнопок «Сохранить» и «Загрузить» пользователь может, соответственно, сохранить текущее расписание в файл и загрузить ранее сохранённое расписание. Эти функции обеспечивают удобство хранения и повторного использования результатов работы алгоритма.

Редактор события

Окно редактора события представлено на рисунке 2.

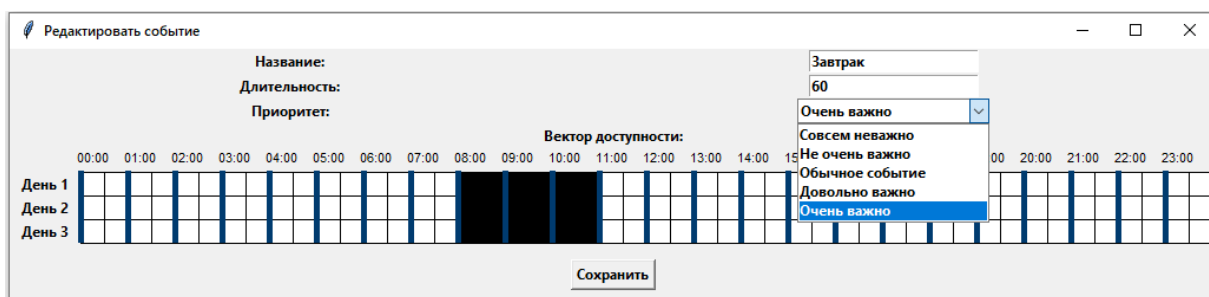


Рисунок 2 Редактор события

Редактор события предназначен для ввода и редактирования параметров, необходимых для включения события в процесс планирования. Интерфейс редактора включает следующие элементы: поле для ввода названия события, поле для указания длительности, элемент управления для выбора приоритета, а также графическое представление вектора доступности.

Название события вводится пользователем в текстовом формате.

Длительность события указывается в виде целого количества минут, необходимых для проведения события, причём если введённое число не кратно длительности временного слота, то при передаче его вычислительному модулю оно округляется в большую сторону.

Приоритет может быть присвоен событию пользователем путём выбора уровня приоритета из выпадающего списка. Уровень приоритета события при передаче данных о событии в вычислительный модуль для дальнейшей обработки преобразуются в численное значение p_i .

Соответствие между названием уровня приоритета и численным значением устанавливается следующим образом:

Очень важно — $p_i = 5$ (расставляются в первую очередь)

Довольно важно — $p_i = 4$

Обычное событие — $p_i = 3$

Не очень важно — $p_i = 2$

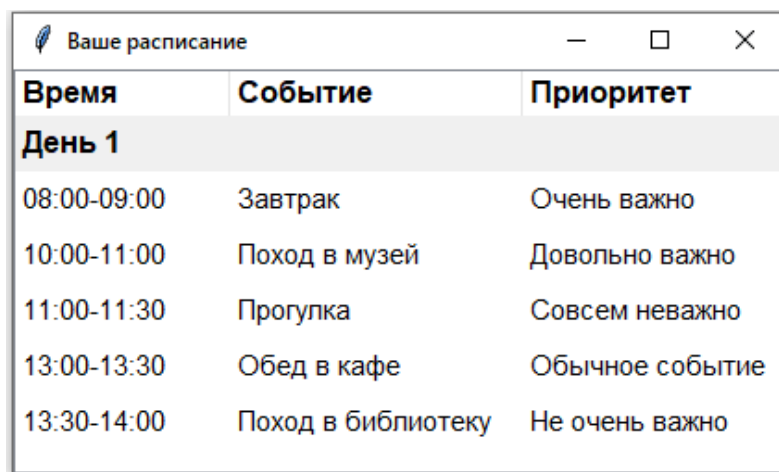
Совсем неважно — $p_i = 1$ (расставляются в последнюю очередь)

Вектор доступности представлен графически в виде временных шкал, объединённых в матрицу. По вертикальной оси указаны дни недели, по горизонтальной — временные интервалы продолжительностью 30 минут, сгруппированные по часу. Каждая ячейка соответствует одному временному слоту и может находиться в одном из двух состояний: белом (значение 0, недоступность) или чёрном (значение 1, доступность). Пользователь может редактировать доступность, используя графический способ ввода посредством нажатия левой кнопки мыши и перемещения указателя по матрице ячеек.

После внесения всех изменений, нажатием кнопки «Сохранить» происходит обновление данных о событии в вычислительном модуле.

Визуализация расписания

Визуализация расписания представлена на рисунке 3.



Время	Событие	Приоритет
День 1		
08:00-09:00	Завтрак	Очень важно
10:00-11:00	Поход в музей	Довольно важно
11:00-11:30	Прогулка	Совсем неважно
13:00-13:30	Обед в кафе	Обычное событие
13:30-14:00	Поход в библиотеку	Не очень важно

Рисунок 3 Визуализация расписания

Сформированное расписание отображается в табличной форме с группировкой по дням. Таблица представляет собой структурированный список событий, отсортированных в хронологическом порядке в пределах каждого дня. Каждая строка таблицы соответствует отдельному событию. В первой колонке указывается временной интервал, отражающий время начала и окончания события. Во второй колонке отображается наименование события, заданное пользователем. Третья колонка содержит информацию о приоритете соответствующего события. Данный формат визуализации обеспечивает наглядность и удобство восприятия структуры расписания, позволяя пользователю оперативно оценить распределение событий во времени, а также их приоритеты.

Заключение

Целью выпускной квалификационной работы являлось проектирование и реализация программного инструмента для автоматизированного составления расписания на основе адаптивного алгоритма с учётом расставленных пользователем приоритетов и ограничений.

В соответствии с поставленной целью в ходе работы были решены следующие задачи:

Были изучены и проанализированы существующие инструменты планирования. В ходе анализа было выявлено, что большинство из них не предусматривают механизмов автоматизированного выявления и разрешения конфликтов, возникающих при формировании расписания.

Была разработана формализованная математическая модель задачи планирования, предлагающая представление временных ресурсов в виде множества подряд идущих временных слотов одинаковой длительности. Слоты организованы в виде векторов доступности, объединённых в матрицу доступности. В рамках математической модели процесс составления расписания сведён к установлению соответствия между множеством событий и множеством временных слотов посредством преобразования матрицы доступности. В качестве варьируемых параметров в математической модели выступают строки матрицы доступности. Сформулированы характеристика вариативности и функция оценки сложности планирования события, которая учитывает его приоритет, установленный пользователем, и вариативность — число способов, сколькими событие может быть включено в расписание.

В рамках математической модели задачи планирования была установлена целевая функция, основанная на количестве событий, которые были потеряны в ходе составления расписания. В рамках задачи планирования необходимо производить поиск решения, минимизирующего значение целевой функции.

Был определён и сформулирован ряд ограничений, которым должно удовлетворять искомое решение для обеспечения логической целостности

расписания. Ограничения включают требования к суммам элементов матрицы доступности по строкам и столбцам, а также к непрерывному характеру размещения элементов в матрице.

Был разработан адаптивный алгоритм преобразования матрицы доступности для поиска оптимального решения и приведения матрицы к виду, удовлетворяющему сформулированным ограничениям и минимизирующему значение целевой функции путём последовательного преобразования и фиксации строк матрицы доступности в соответствии с её динамически обновляемым состоянием.

Разработана и реализована структура данных для хранения и управления элементами математической модели в памяти вычислительной машины в виде пользовательского класса, написанного на языке Python. Реализованы методы для инициализации структуры исходными данными и инструменты для её модификации в процессе работы алгоритма.

Разработанный на теоретическом уровне адаптивный алгоритм преобразования матрицы доступности был реализован в составе программного вычислительного модуля в виде библиотеки на языке Python.

Средствами библиотеки Tkinter языка Python был реализован пользовательский интерфейс в виде компьютерного приложения-планировщика, обеспечивающего взаимодействие пользователя с компонентами вычислительного модуля для автоматизированного составления расписания на основе вводимых пользователем данных. Создан графический редактор векторов доступности.

Реализация адаптивного алгоритма подтвердила свою эффективность в условиях динамически изменяющихся данных.

Результаты работы могут быть применены в различных областях, требующих оптимизации временных ресурсов и автоматизации процессов планирования. Адаптивность метода позволяет применять его в условиях высокой неопределённости, требующих оперативного перераспределения ресурсов в реальном времени. Предложенный алгоритм может быть внедрён в

уже существующие системы, рассмотренные в разделе II или предложенные в работах [6, 11, 15], включая облачные грид-системы [5], в виде опционального предложения автоматизированного разрешения конфликтов в случае обнаружения таковых при составлении расписания вручную. Также алгоритм может быть улучшен введением дополнительных критериев и интеграцией дополнительных параметров, которые учитываются при составлении расписания, таких как равномерность распределения нагрузки или учёт географической привязки событий и времени на дорогу, и стратегиями самостоятельной оптимизации, предложенных в работе [20], интеграцией искусственного интеллекта [1].

Список источников


1. Беседин О. А., Пюра П. К. Проектирование планировщика задач на основе искусственного интеллекта // Вестник науки. – 2024. – Т. 3. – №. 6 (75). – С. 1324-1329.
2. Данилов М. В. Методы планирования выполнения задач в системах реального времени // Программные продукты и системы. — 2001. — № 4. — С. 28–35.
3. Жеребин В. М., Вершинская О. Н., Махрова О. Н. Современное восприятие времени и ускорение темпа жизни // Народонаселение. — 2014. — № 2 (64). — URL: <https://cyberleninka.ru/article/n/sovremennoe-voSPIriatie-vremeni-i-uskorenie-tempa-zhizni> (дата обращения: 10.12.2024).
4. Клеванский Н. Н., Мавзовин В. С. Математическая модель формирования расписания экзаменов ВУЗа // Современные наукоемкие технологии. — 2018. — № 5. — С. 97–102. — URL: <https://top-technologies.ru/ru/article/view?id=36998> (дата обращения: 20.12.2024).
5. Кошкарева Ю. И., Третьякова А. А. Моделирование подходов к распределению ресурсов и оценка времени распределения задач в облачных грид системах // Седьмая всероссийская научно-практическая конференция "Имитационное моделирование. Теория и практика" (ИММОД-2015). — 2015. — С. 267–274.
6. Кучумов Р. И. Реализация и анализ work-stealing планировщика задач // Стохастическая оптимизация в информатике. – 2016. – Т. 12. – №. 1. – С. 20-39.
7. Лучкив А. А. Разработка telegram бота «планировщик задач» // Редакционная коллегия кандидат технических наук, научный сотрудник Светличная ЛА (отв. ред.) доктор экономических наук, научный сотрудник Чернова ТВ Политехнический институт (филиал) ДГТУ в г. Таганроге Наука и современность: материалы всероссийской научно-практической. – 2021. – С. 107.


8. Павлычев В. А., Хлыстов Н. С. Полуавтоматизированная система составления расписания в учебном заведении // Решетневские чтения. 2017. №21-2. URL: <https://cyberleninka.ru/article/n/poluavtomatizirovannaya-sistema-sostavleniya-raspisaniya-v-uchebnom-zavedenii> (дата обращения: 10.12.2024).
9. Платов А. Ю., Гончаров М. С. Разработка планировщика задач на платформе Android //Актуальные проблемы гуманитарных, инженерных и социально-экономических наук в свете современных исследований. – 2018. – С. 114-117.
10. Прихожий А. А., Фролов О. М. Исследование планировщиков задач в GRID //Системный анализ и прикладная информатика. – 2015. – №. 1. – С. 15-23.
11. Расулова Н. Ю. Мультиагентный подход в создании адаптивных интеллектуальных обучающих систем //Экономика и социум. – 2021. – №. 2-2 (81). – С. 524-531.
12. Рябец М. А. Разработка эмулятора планировщика задач реального времени //МНСК-2018: Информационные технологии. – 2018. – С. 157-157.
13. Сиркин, Т. В. Разработка автоматизированной системы составления и оптимизации расписания занятий / Т. В. Сиркин, А. П. Чернышова, П. А. Мартынов, А. Д. Морозов. — Текст: непосредственный // Молодой ученый. — 2020. — № 27 (317). — С. 65-71. — URL: <https://moluch.ru/archive/317/72336/> (дата обращения: 19.01.2025).
14. Фадеев Р. Н., Огурцова Е. Ю. Интеллектуальный анализ технологических данных //Наука и образование в современном вузе: вектор развития. – 2022. – С. 302-304.
15. Шориков А. Ф., Буценко Е. В. Сетевое экономико-математическое моделирование оптимизации адаптивного управления процессами бизнес-планирования //Экономика и математические методы. – 2021. – Т. 57. – №. 3. – С. 110-125.

16. Ciolek D. et al. Multi-tier automated planning for adaptive behavior //Proceedings of the International Conference on Automated Planning and Scheduling. – 2020. – T. 30. – C. 66-74.
17. Geffner H., Bonet B. A concise introduction to models and methods for automated planning. – Springer Nature, 2022.
18. Gil R. Automated planning for self-adaptive systems //2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. – IEEE, 2015. – T. 2. – C. 839-842.
19. Kahar M.M., Kendall G. A great deluge algorithm for a real-world examination timetabling problem // Journal of the Operational Research Society. – 2013. – Vol.67, № 3. – P. 116–133.
20. Lesch V. et al. Self-aware optimization of adaptation planning strategies //ACM Transactions on Autonomous and Adaptive Systems. – 2023. – T. 18. – №. 3. – C. 1-35.

Приложение А

Результат проверки на уникальность



TEXT.RU


0 Р

Text.ru в Telegram

Контент о текстах, маркетинге и нейросетях

Реклама: ООО "ТЕКСТ МЕДИА" ИНН 3662229020 erid: 2VfmxVWuFE



Онлайн проверка уникальности

ПРОВЕРКА ТЕКСТА НА УНИКАЛЬНОСТЬ
>
РЕЗУЛЬТАТЫ
↓

+ **Новый текст**

Проверка уникальности

Уникальность: 97.21%

moluch.ru/archive/317/72336/
1%

colab.ws/articles/10.1057/jors.2012.169
0%

Подробнее

Подсвечено:

Неуникальные фрагменты

1 **МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М.В. ЛОМОНОСОВА ФИЛИАЛ МГУ В ГОРОДЕ СЕВАСТОПОЛЕ**

Направление подготовки «**Прикладная математика и информатика**» 01.03.02 (бакалавр) ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА АВТОМАТИЗИРОВАННОЕ ПЛАНИРОВАНИЕ НА ОСНОВЕ АДАПТИВНОГО АЛГОРИТМА Выполнила: Заричная Екатерина Викторовна студентка учебной группы ПМ-401 Научный руководитель: ст. преподаватель кафедры алгоритмических языков Баева Наталия Валерьевна Севастополь – 2024 2 Содержание Аннотация

Результат проверки уникальности на ресурсе text.ru.



Отчет предоставлен сервисом
«Антиплагиат» - [http://dep02-
msu.antiplagiat.ru](http://dep02-msu.antiplagiat.ru)



Отчет о проверке

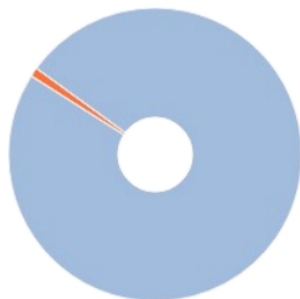
Автор: Баева Наталия Валерьевна

Название документа: ЗаричнаяЕВ

Проверяющий: Волканов Дмитрий Юрьевич

Организация: Факультет вычислительной математики и кибернетики

РЕЗУЛЬТАТЫ ПРОВЕРКИ



Совпадения:
1,27%



Оригинальность:
98,73%



Цитирования:
0%



Самоцитирования:
0%



ИНФОРМАЦИЯ О ДОКУМЕНТЕ

Номер документа: 78

Тип документа: Не указано

Дата проверки: 13.05.2025 16:59:59

Дата корректировки: Нет

Количество страниц: 54

Символов в тексте: 79062

Слов в тексте: 9284

Число предложений: 3073

Комментарий: не указано

ИСТОЧНИКИ

№	Доля в тексте	Доля в отчете	Источник	Актуален на	Модуль поиска	Комментарий
[01]	0,83%	0,83%	https://nnov.hse.ru/data/2020/06/... https://nnov.hse.ru	21 Дек 2023	Интернет Плюс	
[02]	0,66%	0%	не указано	29 Сен 2022	Шаблонные фразы	Источник исключен. Причина: Маленький процент пересечения.
[03]	0,44%	0,44%	Михненко П.А. Теория менеджме... http://ivo.garant.ru	13 Сен 2014	Перефразирования по СПС ГАРАНТ: аналитика	
[04]	0,3%	0%	Противодействие преступления... http://ivo.garant.ru	18 Мар 2023	Перефразирования по СПС ГАРАНТ: аналитика	Источник исключен. Причина: Маленький процент пересечения.
[05]	0,27%	0%	Ознакомьтесь с электронной ве... http://wwwrus.imm.uran.ru	21 Окт 2014	Переводные заимствования по коллекции Интернет в русском сегменте	Источник исключен. Причина: Маленький процент пересечения.
[06]	0,26%	0%	kurbanova_s_m_servis-dlya-veden...	26 Мая 2023	Кольцо вузов (переводы и перефразирования)	Источник исключен. Причина: Маленький процент пересечения.
[07]	0,14%	0%	не указано	13 Янв 2022	Цитирование	Источник исключен. Причина: Маленький процент пересечения.
[08]	0,12%	0%	А. В. Карпов, И. Г. Савин ; М-во о... http://dlib.rsl.ru	01 Янв 2005	Публикации РГБ	Источник исключен. Причина: Маленький процент пересечения.
[09]	0,12%	0%	Кошкина, Вероника Константино... http://dlib.rsl.ru	01 Янв 2011	Публикации РГБ	Источник исключен. Причина: Маленький процент пересечения.
[10]	0,12%	0%	[авторы-составители: О. М. Луго... http://dlib.rsl.ru	01 Янв 2017	Публикации РГБ	Источник исключен. Причина: Маленький процент пересечения.
[11]	0,12%	0%	[Аверкин Р.Г., Александров И.О., ... http://dlib.rsl.ru	раньше 2011	Публикации РГБ	Источник исключен. Причина: Маленький процент пересечения.
[12]	0,08%	0%	Мезенцев-диплом антиплаг.docx	22 Июн 2021	Кольцо вузов	Источник исключен. Причина: Маленький процент пересечения.

Результаты проверки уникальности сервисом Антиплагиат.

