МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра дискретной математики и алгоритмики

ВУСИК Дарья Владимировна

МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ О ПЛАНИРОВАНИИ РАСПИСАНИЯ УЧЕБНЫХ ЗАНЯТИЙ

Дипломная работа

Научный руководитель:
старший преподаватель
И.П. Молочко

Допущена к защите
«____»_____20__г.
Заведующий кафедрой дискретной математики и алгоритмики
доктор физико-математических наук, профессор В.М. Котов

Минск, 2025

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Дипломная работа, 34 страницы, 1 иллюстрация, 10 таблиц, 4 источника, 4 приложения.

Ключевые слова: СОСТАВЛЕНИЕ РАСПИСАНИЙ, ЦЕЛОЧИСЛЕННОЕ ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ, ЭВРИСТИКИ ЛОКАЛЬНОГО ПОИСКА, МОДЕЛЬ ЦЛП, ИМИТАЦИЯ ОТЖИГА, АЛГОРИТМ LOCAL-MIP, ОПТИМИЗАЦИЯ.

Объект исследования: алгоритмы и методы для автоматизированного составления учебных расписаний.

Предмет исследования: разработка и анализ эффективности решателя HiGHS в комбинации с эвристиками локального поиска.

Цель работы: исследование возможностей решателя HiGHS и модифицированных эвристик для решения задач составления расписаний, а также разработка ЦЛП модели , учитывающей реальные требования образовательного процесса.

Методы исследования: формализация задачи в виде целочисленной линейной модели, интеграция эвристик (Local-MIP, имитация отжига) в решатель HiGHS, проведение вычислительных экспериментов с различными конфигурациями алгоритмов, сравнительный анализ времени решения, качества и допустимости расписаний.

Результаты: разработана ЦЛП модель для составления учебных расписаний, учитывающая ограничения по аудиториям, преподавателям, группам и временным слотам; Реализованы две дополнительные версии алгоритма Local-MIP: Local-MIP с приоритезацией бинарных переменных и гибридная версия с имитацией отжига; Все вариации эвристик были интегрированы в решатель HIGHS; Проведено тестирование на задачах различных масштабов и проведен сравнительный анализ результатов тестирования.

Области применения: разработанные методы могут быть использованы в системах автоматизации составления расписаний для учебных заведений, корпоративных тренингов и других сфер, где требуется баланс между качеством решений и временными ограничениями.

РЭФЕРАТ

Дыпломная праца, 34 старонкі, 1 ілюстрацый, 10 табліц, 4 крыніцы, 4 дадаткі.

Ключавыя словы: СКЛАДАННЕ РАСПІСАННЯЎ, ЦЭЛАЛІКАВАЕ ЛІНЕЙНАЕ ПРАГРАМАВАННЕ, ЭЎРЫСТЫКІ ЛАКАЛЬНАГА ПОШУКУ, МОДЭЛЬ ЦЛП, ІМІТАЦЫЯ АДЖЫГУ, АЛГАРЫТМ LOCAL-MIP, АПТЫМІЗАЦЫЯ.

Аб'ект даследавання: алгарытмы і метады для аўтаматызаванага складання навучальных распісанняў.

Прадмет даследавання: распрацоўка і аналіз эфектыўнасці вырашальніка HiGHS у камбінацыі з эўрыстыкамі лакальнага пошуку.

Мэта працы: даследаванне магчымасцей вырашальніка HiGHS і мадыфікаваных эўрыстык для вырашэння задач складання распісанняў, а таксама распрацоўка мадэлі ЦЛП, якая ўлічвае рэальныя патрабаванні адукацыйнага працэсу.

Метады даследавання: фармалізацыя задачы ў выглядзе цэлалікавай лінейнай мадэлі, інтэграцыя эўрыстык (Local-MIP, імітацыя аджыгу) у вырашальнік HiGHS, правядзенне вылічальных эксперыментаў з рознымі канфігурацыямі алгарытмаў, параўнальны аналіз часу рашэння, якасці і дапушчальнасці распісанняў.

Вынікі: распрацавана мадэль ЦЛП для складання навучальных распісанняў, якая ўлічвае абмежаванні па аўдыторыях, выкладчыках, групах і часавых слотах; рэалізаваны дзве дадатковыя версіі алгарытму Local-MIP: Local-MIP з прыярытэзацыяй двайковых зменных і гібрыдная версія з імітацыяй аджыгу; усе варыяцыі эўрыстык былі інтэграваны ў вырашальнік HiGHS; праведзена тэставанне на задачах розных маштабаў і параўнальны аналіз вынікаў тэставання.

Вобласці прымянення: распрацаваныя метады могуць быць выкарыстаны ў сістэмах аўтаматызацыі складання распісанняў для навучальных устаноў, карпаратыўных трэнінгаў і іншых сфер, дзе патрабуецца баланс паміж якасцю рашэнняў і часавымі абмежаваннямі.

ABSTRACT

Thesis, 34 pages, 1 illustration, 10 tables, 4 references, 4 appendices.

Keywords: TIMETABLE SCHEDULING, INTEGER LINEAR PROGRAMMING, LOCAL SEARCH HEURISTICS, ILP MODEL, SIMULATED ANNEALING, LOCAL-MIP ALGORITHM, OPTIMIZATION.

Research object: algorithms and methods for automated educational timetable scheduling.

Research subject: development and efficiency analysis of HiGHS solver combined with local search heuristics.

Research goal: investigation of HiGHS solver capabilities and modified heuristics for timetable scheduling problems, along with development of an ILP model accounting for real-world educational requirements.

Research methods: problem formalization as an integer linear programming model, integration of heuristics (Local-MIP, simulated annealing) into HiGHS solver, computational experiments with different algorithm configurations, comparative analysis of solution time, quality and schedule feasibility

Results: developed ILP model for educational timetable scheduling considering classroom, teacher, group and timeslot constraints; implemented two additional algorithm versions of Local-MIP: Local-MIP with binary variable prioritization and hybrid version with simulated annealing; all heuristic variations were integrated into HiGHS solver; testing performed on problems of various scales with comparative analysis of results

Application areas: the developed methods can be applied in automated scheduling systems for educational institutions, corporate training programs and other domains requiring balance between solution quality and time constraints.

ВВЕДЕНИЕ

Составление учебного расписания — сложная оптимизационная задача, требующая учета множества взаимозависимых факторов. В современных образовательных учреждениях важно не только рационально распределить ресурсы, но и избежать конфликтов, а также соблюсти все нормативные требования. Для этого система должна обрабатывать данные о занятиях, преподавателях, аудиториях и студенческих группах, формируя согласованное расписание для всех участников процесса.

Ручное составление расписания часто приводит к ошибкам и требует значительных временных затрат. Например, при распределении аудиторий необходимо постоянно сверяться с их техническими характеристиками, а проверка квалификации преподавателей предполагает анализ большого объема данных. Даже опытные координаторы тратят 10–15 часов в неделю на корректировки, тогда как менее подготовленные сотрудники могут допускать серьезные недочеты — например, назначать занятия в аудитории, превышая их вместимость.

В данной работе рассматривается подход на основе целочисленного линейного программирования (ЦЛП): входные данные преобразуются в формализованную задачу, которая решается с помощью специализированного солвера. Для улучшения результатов исследуется возможность применения эвристик — как в виде отдельного предварительного этапа обработки данных, так и через интеграцию в сам солвер. Предложенный метод не гарантирует оптимальности для всех случаев, но позволяет сократить время составления расписания и уменьшить число ошибок по сравнению с ручным подходом.

Разрабатываемый инструмент направлен на объединение математической строгости ЦЛП с гибкостью эвристических методов, что соответствует общему запросу учебных заведений на частичную автоматизацию административных процессов.

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ И ПОСТРОЕНИЕ МОДЕЛИ

1.1 Цель проводимого исследования

Основной целью данной работы является разработка комплексного подхода к автоматизации процесса составления учебного расписания, сочетающего методы математического моделирования и эвристической оптимизации. В рамках работы предполагается достижение следующих ключевых задач:

1. Сравнительный анализ существующих методов решения задач составления расписания
2. Формализация задачи в терминах целочисленного линейного программирования (ЦЛП) с учетом жестких и дополнительных ограничений (распределение аудиторий, квалификация преподавателей, логические связи между занятиями). Интеграция в модель параметров, отражающих специфику образовательного процесса (нагрузка преподавателей, требования к оборудованию, временные предпочтения).
3. Разработка гибридного алгоритма оптимизации. Интеграция эвристики, направленной на сокращение пространства поиска решений и ускорение работы солвера. Комбинирование точных методов (на основе решателя) и эмпирических правил для баланса между оптимальностью и скоростью вычислений.
4. Экспериментальная верификация подходов: сравнение результатов, полученных с использованием решателя, эвристики и их гибридной комбинации; тестирование модели на близких к реальным данных учебных учреждений с оценкой сложности формирования расписания.

1.2 Описание сущностей и зависимостей между ними

Для формирования задачи ЦЛП сначала необходимо определится, что именно нужно моделировать, какие сущности будут входить в модель, какие зависимости будут между ними и какие ограничения должны на них накладываться.

Сущности

- Множество занятий (C): Учебные дисциплины, требующие распределения по временным слотам, аудиториям и преподавателям.
- Множество временных слотов (T): 6 рабочих дней в неделю, 9 интервалов ежедневно (с 8:15 до 22:00) с понедельника по пятницу (то есть первые пять дней), в 6-м дне 5 слотов; Всего слотов 50, это число остается неизменным.
- Множество аудиторий (R): Помещения с фиксированной вместимостью и оборудованием (проекторы, лабораторные установки и т.д.).
- Множество преподавателей (P): Сотрудники, обладающие квалификацией для проведения конкретных занятий. Для каждого преподавателя задаются: доступное время, нагрузка (макс. часов), список занятий, которые они могут провести, предпочтения по занятиям и времени их проведения.
- Множество групп (G): Студенческие коллективы с фиксированной численностью. Занятия уже распределены по группам, но требуют привязки к расписанию.

Зависимости и ограничения

Каждое занятие $c \in C$ должно быть однозначно связано с временным слотом $t \in T$; аудиторией $r \in R$; преподавателем $p \in P$, обладающим необходимой квалификацией.

Ключевые зависимости:

- Преподаватели могут задать допустимые временные интервалы и аудитории для проведения занятий.
- Нагрузка преподавателей не должна превышать установленный лимит часов.
- Аудитории должны соответствовать требованиям занятий по вместимости и оборудованию.
- Занятия могут иметь ограничения на последовательность (например, лекция должна предшествовать семинару) или распределение по дням.

Жесткие ограничения

Выполнение данных условий является обязательным для формирования допустимого расписания. Эти ограничения присутствуют в любой модели задачи составления расписания.

- Уникальность ресурсов: Никакие два занятия не могут занимать одну аудиторию в одно и то же время.
- Преподаватель не может одновременно вести более одного занятия.
- Соответствие аудиторий: Вместимость аудитории должна быть не меньше численности посещающих занятие студентов.
- Оборудование аудитории должно полностью покрывать требования занятия (например, наличие проектора для лекций).
- Квалификационная привязка: Преподаватель назначается на занятие только при наличии соответствующей квалификации.

Дополнительные ограничения

Расписание останется допустимым и без этих ограничений, однако с их помощью можно задать более конкретные требования к моделируемой задаче, такие как:

- Временные предпочтения: Ограничение на проведение занятий в конкретные дни или временные интервалы (например, избегать вечерних слотов для первых курсов).
- Требование проводить связанные занятия в один день или, наоборот, распределять их по разным дням.
- Персональные настройки: Преподаватели могут фиксировать предпочтительные аудитории или временные слоты для конкретных занятий.
- Требования на последовательность занятий: Занятие А должно предшествовать занятию Б в рамках учебного плана.
- Группировка занятий одной дисциплины в рамках одного дня для минимизации перемещений студентов и преподавателей.

1.3 Формулировка модели в терминах ЦЛП

При построении модели были использованы некоторые выкладки и принципы, изложенные в статье [1].

В этом разделе будут описаны переменные и возможный способ задания ограничений для модели в терминах ЦЛП.

1.3.1 Переменные:

$a_{c,t,r} = 1$, если занятие $c \in C$ проходит во временной слот $t \in T$ в аудитории $r \in R$, и $a_{c,t,r} = 0$, в противном случае.

$z_{p,t,c} = 1$, если преподаватель $p \in P$ проводит занятие $c \in C$ во время $t \in T$ и $z_{p,t,c} = 0$ в противном случае.

1.3.2 Основные ограничения:

- Каждому занятию нужно предоставить одну аудиторию и один временной слот:

$$\forall c \in C: \quad \sum_{t \in T} \sum_{r \in R} a_{c,t,r} = 1 \tag{1.1}$$

- Каждому занятию необходимо назначить только одного преподавателя:

$$\forall c \in C: \quad \sum_{p \in P} \sum_{t \in T} z_{ptc} = 1 \tag{1.2}$$

- Никакие два занятия не должны проходить в одной и той же аудитории в одно и то же время:

$$\forall r \in R, \ \forall t \in T : \sum_{c \in C} a_{c,t,r} \leq 1 \tag{1.3}$$

- Преподаватель не может вести две пары одновременно:

$$\forall p \in P \ \forall t \in T: \sum_{c \in C} z_{ptc} \leq 1 \tag{1.4}$$

- Аудитории должны вмещать всех студентов, у которых здесь проходит занятие:

$$\forall t \in T, \ \forall r \in R: \sum_{c \in C} g_c \cdot a_{c,t,r} \leq v_r \tag{1.5}$$

Помимо этих ограничений, нужно так же добится согласованности распределения занятий по аудиториям и преподавателям. Это достигается с помощью следующих ограничений:

- Синхронизация распределения занятий по аудиториям и по преподавателям:

$$\forall c \in C, \forall t \in T: \sum_{r \in R} a_{trc} - \sum_{p \in P} z_{tcp} = 0 \tag{1.6}$$

1.3.3 Дополнительные ограничения:

- Ограничение на последовательность занятий:
  Занятие $c_1 \in C$ идет после занятия $c_2 \in C$:

$$\sum_{r \in R} \sum_{t \in T} t * a_{c_2 tr} \leq \sum_{r \in R} \sum_{t \in T} t * a_{c_1 tr} + 1 \tag{1.7}$$

Здесь и далее t – порядковый номер временного слота.

- Ограничение на проведение занятий $c_1,\ c_2 \in C$ в один день:

$$\sum_{r \in R} \sum_{t \in T} d(t) * a_{c_2 t r} = \sum_{r \in R} \sum_{t \in T} d(t) * a_{c_1 t r} \quad (1.8)$$

где $d(t) = [(t - 1) / num_{days} + 1]$ – номер дня,

num_{days} – количество рабочих дней.

- Ограничение на проведение занятий $c_1,\ c_2 \in C$ в разные дни:

$$\left| \sum_{r \in R} \sum_{t \in T} d(t) * a_{c_2 t r} - \sum_{r \in R} \sum_{t \in T} d(t) * a_{c_1 t r} \right| \geq 1 \quad (1.9)$$

- Ограничение на определенное время $t_1 \in T$ проведения занятия $c_1 \in C$

$$\sum_{r \in R} a_{c_1 t_1 r} = 1 \quad (1.10)$$

- Ограничение на недоступное время $t_1 \in T$ для преподавателя $p_1 \in P$:

$$\sum_{c \in C} z_{c\ p_1 t_1} = 0 \quad (1.11.1)$$

или, что то же самое

$$\forall c \in C: z_{c, p_1, t_1} = 0 \quad (1.11.2)$$

- Ограничение на невозможность проведения преподавателем $p_1 \in P$ занятия $c_1 \in C$:

$$\sum_{t \in T} z_{c_1 p_1 t} = 0 \quad (1.12.1)$$

или

$$\forall t \in T: z_{c_1 p_1 t} = 0 \quad (1.12.2)$$

- Ограничение на проведение занятий $c_1,\ c_2 \in C$ одним и тем же преподавателем $p \in P$:

$$\sum_{t \in T} \sum_{p \in P} z_{c_1 p t} = 1 \quad (1.13)$$

$$\sum_{t \in T} \sum_{p \in P} z_{c_2 pt} = 1$$

- Ограничение на недоступность аудиторий в некоторое время:

$$\sum_{c \in C} a_{ct_1 r_1} = 0 \quad (1.14)$$

- Ограничение по нагрузке преподавателей:

$$\sum_{c \in C} \sum_{t \in T} z_{c,p,t} \leq Z_p, \forall p \in P \quad (1.15)$$

где Z_p – максимальная нагрузка преподавателя p.

- Ограничение недоступности аудиторий для занятия $c \in C$ при отсутствии необходимого оборудования:

$$\forall c \in C, \forall r \in R_{\bar{c}}, \forall t \in T: a_{c,t,r} = 0 \quad (1.16)$$

где $R_{\bar{c}}$ – множество аудиторий, которые не удовлетворяют требованиям по оборудованию для занятия c.

- Ограничение на проведение занятий в один день сразу друг за другом можно задать как комбинацию из ограничений (1.7) и (1.8):

$$\sum_{r \in R} \sum_{t \in T} t * a_{c_2 tr} \leq \sum_{r \in R} \sum_{t \in T} t * a_{c_1 tr} + 1$$

$$\sum_{r \in R} \sum_{t \in T} d(t) * a_{c_2 tr} = \sum_{r \in R} \sum_{t \in T} d(t) * a_{c_1 tr} \quad (1.17)$$

Представленные ограничения могут быть классифицированы как жесткие или мягкие (lazy constraints), при этом последние включаются в модель исключительно при обнаружении допустимого решения, нарушающего указанные условия. Базовые ограничения, такие как условия типов 5–8, рекомендуется формулировать в качестве стандартных ограничений. Остальные условия в зависимости от их количества могут быть определены как мягкие либо также преобразованы в стандартные.

Альтернативным методом интеграции дополнительных ограничений является их имплементация в целевую функцию посредством штрафных коэффициентов. Данный подход позволяет учитывать нарушения условий без их явного включения в систему ограничений модели.

1.3.4 Целевая функция:

В этой работе в целевой функции учитывается штраф за пробелы в расписании занятий для групп, и штраф за количество занятий в день в расписании групп в том случае, если это количество превышает 4.

Штраф за пробелы можно задать следующим образом:

Пусть $b(d,\ t,\ g\ ) = \sum_{c \epsilon C_g} \sum_{r \epsilon R} a_{c,r,t}$, где t – временной слот дня d, $g\ \in G$ – группа студентов.

$b(d,\ t,\ g)$ принимает значений 1, если у группы g есть занятие во временной слот t дня d, и 0 в противном случае.

Тогда количество пробелов будет равно

$$B(g,\ d) = \sum_{t\ =\ 1}^{|T_d| - 1} |b(d,\ t,\ g)\ - b(d, t\ +\ 1,\ g)\ | \qquad (1.18)$$

где T_d – множество временных слотов в дне d.

Штраф для количества занятий в день d для студентов группы g:

$$f(g,\ d)\ = (\ - \sum_{c\ \epsilon\ C_g} \sum_{r\ \epsilon\ R} \sum_{t\ \epsilon\ T_d} a_{c,\ r,\ t}\ +\ 4) \qquad (1.19)$$

Тогда целевая функцию можно задать как:

$$\sum_{i\ =\ 1}^{6} \sum_{g\ \in\ G} (f(g,\ d)\ -\ B(g,\ d)\ ) \rightarrow max \qquad (1.20)$$

1.4 Разработка и программная реализация

Для перевода сформулированной целочисленной линейной модели (ЦЛП) в формализованный машинно-читаемый вид, который может быть обработан решателем, был выбран стандартизированный формат представления оптимизационных задач .mps (Mathematical Programming System). Данный текстовый формат, разработанный IBM в 1970-х годах, обеспечивает унифицированное описание линейных, целочисленных и смешанно-целочисленных моделей, включая информацию о переменных, ограничениях и целевой функции. Его структура, основанная на секциях (ROWS, COLUMNS, RHS, BOUNDS), позволяет точно транслировать математическую модель в машинно-читаемый вид, что обеспечивает совместимость с большинством современных решателей (например, CPLEX, Gurobi, HiGHS). Выбор .mps обусловлен его прозрачностью, поддержкой целочисленных переменных и возможностью последующего анализа модели вне зависимости от используемого инструментария.

Реализация модели выполнена с использованием библиотеки PuLP (Python Linear Programming), предоставляющей интерфейс для описания и решения задач линейной оптимизации. PuLP позволяет декларативно задавать переменные (включая целочисленные и бинарные), линейные ограничения и целевую функцию, автоматически генерируя модель в форматах .lp или .mps. Ограничения в PuLP задаются через объекты LpConstraint, где левая часть выражения формируется как линейная комбинация переменных, а правая — фиксированное значение.

Хотя PuLP поддерживает интеграцию с внешними решателями (CBC, GLPK, CPLEX и др.), в рамках данной работы использовался исключительно солвер HiGHS (High-performance Parallel Search for LP/IP) [2]. Этот выбор обусловлен его открытым исходным кодом, высокой производительностью на задачах смешанно-целочисленного программирования и наличием оптимизированных алгоритмов для работы с разреженными матрицами.

Модель, сгенерированная в PuLP, экспортировалась в .mps и передавалась в HiGHS[2] через через аргументы командной строки так же, как и основные настройки решателя. В Приложении Г приведен код формирования модели в задачу в формате .mps.

1.4.1 Адаптация исходной модели к требованиям формата .mps

Исходная целочисленная линейная модель содержала нелинейные компоненты — знаки модуля в ограничениях (9) и в штрафе (16), которые не поддерживаются стандартом .mps и большинством MIP-солверов, ориентированных на линейные зависимости. Для устранения этой проблемы была применена методология линеаризации, суть которой заключается в замене переменных следующего вида

$y = |x_1 - x_2|$ для двух переменных x_1, x_2 с $1 \leq x_i \leq U$. Введем бинарные переменные d_1, d_2 обозначающие

d_1: 1 если $x_1 - x_2$ имеет положительный значение

d_2: 1 если $x_2 - x_1$ имеет положительное значение

MIP формулировка будет иметь следующий вид:

$$0 \leq x_i \leq U, i = \overline{1, 2} \tag{1.21}$$

$$0 \leq y - (x_1 - x_2) \leq 2 \cdot U \cdot d_2 \tag{1.22}$$

$$0 \leq y - (x_2 - x_1) \leq 2 \cdot U \cdot d_1 \tag{1.23}$$

$$d_1 + d_2 = 1 \tag{1.24}$$

Данный подход позволил сохранить линейность модели, обеспечив её совместимость с форматом .mps и решателем HiGHS[2].

1.4.2 Вычислительная сложность модели

После линеаризации итоговое количество переменных и ограничений определяется параметрами исходной задачи.

Пусть модель строится для C занятий, P преподавателей, T временных слотов, R аудиторий и G групп.

Тогда основных переменных будет $C \cdot R \cdot T + C \cdot T \cdot P$. (переменные a и z).

Проведем оценку количества ограничений и количества задействованных ограничений.

(1.1) Вводится по 1 для каждого занятия и задействует $T \cdot R$ переменных. То есть, добавляет $C \cdot (T \cdot R)$ненулевых элементов в матрицу A задачи.

(1.2) Так же вводится по 1 для каждого занятия и задействует $P \cdot T$ переменных. Добавляет $C \cdot (P \cdot T)$ ненулевых элементов в матрицу A.

(1.3) – $R \cdot T$ ограничений, задействующие C переменных. Добавляет $C \cdot (T \cdot R)$ ненулевых элементов в матрицу A задачи.

(1.4) – $P \cdot T$ ограничений задействующие C переменных. Добавляет $C \cdot (P \cdot T)$ ненулевых элементов в матрицу A.

(1.5) – $R \cdot T$ ограничений, задействующие C переменных. Добавляет $C \cdot (T \cdot R)$ ненулевых элементов в матрицу A задачи.

(1.6) – $C \cdot T$ ограничений, задействующие $R + P$ переменных.

Для дополнительных ограничений:

(1.7) – По одному ограничению на каждую пару занятий, для которых задана последовательность, использует $2 \cdot R \cdot T$ переменных.

(1.8) – По одному ограничению на каждую пару занятий, для которых известно, что они должны быть в один день, задействующих $2 \cdot R \cdot T$ переменных.

(1.9) - Для каждый пары занятий, для котрых известно, что они должны проходить в разные дни, вводится одно неравенство, использующего один знак модуля, для которого вводится:

$$y = |\sum_{r \in R} \sum_{t \in T} d(t) * a_{c_2 tr} - \sum_{r \in R} \sum_{t \in T} d(t) * a_{c_1 tr}|$$

И две бинарные переменные d_1, d_2

Помимо этого добавляются ограничения вида (1.22), (1.23) задействующие $2 \cdot R \cdot T + 1$ переменных каждое, и ограничение (1.24) задействующее 2 переменные.

Значит, в итоговую модель каждое ограничение вида (1.9) добавляет дополнительно 3 переменные, 3 ограничения и $4 \cdot R \cdot T + 4$ ненулевых элемента в матрицу A.

(1.10) – Для каждого занятия с фиксированным временем вводится ограничение, задействующее R переменных.

(1.11) – Для каждой пары преподавателя и недоступного для него времени вводится ограничение, задействующее C переменных или C ограничений с одной переменной.

(1.12) – Для каждой пары преподавателя и недоступного для него занятия вводится ограничение, задействующее T переменных, или T ограничений с одной переменной.

(1.13) – По два ограничения для каждой пары занятий, задействующее $T \cdot P$ переменных каждое.

(1.14) – По ограничению для каждой пары аудитория и недоступное время, задействующее C переменных.

(1.15) – P ограничений, задействующих $C \cdot T$ переменных. Добавляет $C \cdot (P \cdot T)$ ненулевых элементов в матрицу A.

(1.16) – Число ограничений оценивается сверху числом $C \cdot R$, каждое использует T переменных.

(1.17) – По два ограничения на каждую пару занятий, для которых задано это ограничение, использует $2 \cdot R \cdot T$ переменных.

Целевая функция в штрафе (1.19) задействует $C \cdot R \cdot T \cdot G$ переменных в худшем случае (в том случае, когда занятие назначено нескольким группам, это занятие учитывается столько раз, скольким группам оно было назначено).

Штраф (1.18) задействует $\sum_{d=1}^{6} (|T_d| - 1)$ выражений с модулем для одной группы g. Каждое добавляет по 3 переменные и по три ограничения, которые в свою очередь задействуют $4 \cdot C \cdot R + 4$ переменных.

Итого $\sum_{d=1}^{6} (|T_d| - 1) \cdot (4 \cdot C \cdot R + 4) \cdot G + C \cdot R \cdot T \cdot G$ задействуется в целевой функции. Помимо этого, штраф (1.18)

дополнительно добавляет $3 \cdot G \cdot \sum_{d=1}^{6} (|T_d| - 1)$ переменных и ограничений.

Разработанная модель демонстрирует полиномиальную вычислительную сложность, определяемую параметрами C, R, G, T и количеством дополнительных ограничений. Ключевым фактором, ограничивающим масштабируемость, являются парные условия (1.7–1.9, 1.13), приводящие к с большому росту числа переменных и ненулевых элементов матрицы. Штрафные функции, учитывающие требования групп, дополнительно увеличивают размерность задачи.

1.4.3 Характеристики сгенерированных задач

В результате различных комбинаций входных данных и разных используемых дополнительных ограничений, было сгенерировано 20 тестовых задач разных размеров. Входные данные для них, такие как квалификация преподавателей, недоступное для них время, требование по оборудованию у занятий и имеющееся оборудование у аудиторий и т.д. генерировались случайным образом. При этом занятия количество занятий задавалось в пределах 30-250, количество преподавателей 15-75, количество аудиторий 15-50, количество временных слотов во всех задачах было 50, количество групп задавалось в пределах 3-20. В таблицах А.1, А.2 приложения А приведены упомянутые задачи и их характеристики, а также типы дополнительных ограничений и их количество.

ГЛАВА 2. ЭВРИСТИКА ЛОКАЛЬНОГО ПОИСКА

В этом пункте будет рассмотрена эвристика локального поиска – Local-MIP[3].

Эта эвристика[3] решает задачи вида:

$$min\{c^T x | \, Ax \leq b, \, l \leq x \leq u, \, x \in R^n, \, x_j \in Z \text{ для всех } j \in I\} \qquad (2.1)$$

Выражение $c^T x$ - целевая функция от переменных x со стоимостями c. l, u – векторы нижних и верхних границ переменных соответственно. Ограничение $A_i x \leq b_i$ обозначается как con_i, оно содержит переменные x_j, $A_{ij} \neq 0$.

Решением s обозначается вектор текущих значений переменных x, $obj(s)$ - значение целевой функции на этом решении, т.е. $obj(s) = c^T s$, s_j обозначает значение переменной x_j.

Решение называется допустимым, если оно удовлетворяет всем ограничениям и границам переменных, а также все целочисленные переменные имеют целые значения в этом решении.

Лучшее найденное решение на текущий момент обозначается как s^*, s^{cur} - текущее решение.

2.1 Введение в локальный поиск

Локальный поиск – это метод оптимизации, который стремится найти наилучшее решение в пространстве возможных решений, начиная с некоторого начального решения и итеративно улучшая его. Алгоритм локального поиска исследует соседние решения, производя небольшие изменения в текущем решении, и выбирает то из них, которое ведет к улучшению целевой функции. Процесс продолжается до тех пор, пока не будет достигнуто локальное оптимальное решение, при котором дальнейшие изменения не приводят к улучшению. Локальный поиск широко используется в задачах комбинаторной оптимизации и может быть эффективно применен в различных областях, включая операционное исследование и искусственный интеллект.

2.2 Эвристика локального поиска Local-MIP

Алгоритм эвристики[3] можно описать следующим образом: на каждой итерации из множества доступных операций, представленных как изменения значений отдельных переменных, выбирается операция с наибольшим

положительным значением оценки (score). В случае отсутствия операций с положительным значением оценки применяется наилучшая из случайно выбранных операций.

Local-MIP[3] использует следующие операции:

Операция "Прорыва" (breakthrough move или btm)

Эта операция изменяет текущее решение чтобы улучшить значение целевой функции.

Пусть есть переменная x_j с ненулевым коэффициентом в целевой функции c_j и некоторое решение s, такое, что $obj(s^*) \leq obj(s)$, оператор прорыва (обозначается как $bm(x_j, s)$) назначает переменной x_j пороговое значение достигая максимального возможного улучшения целевой функции по сравнению с obj(s*), не нарушая границ переменной.

Пусть ε - положительное малое число, обеспечивающее строгое улучшение целевой функции. Пусть

$$\Delta_j = (obs(s^*) - obj(s) - \varepsilon)/c_j \quad (2.2)$$

Оператор прорыва $bm(x_j, s)$ назначает переменной x_j новое значение s_j^{new}, равное $min(s_j + \Delta_j, u_j)$, если коэффициент c_j отрицательный, и $max(s_j + \Delta_j, l_j)$ в противном случае для действительных переменных, для целочисленных: $min(\lceil s_j + \Delta_j \rceil, \lfloor u_j \rfloor)$, если коэффициент c_j отрицательный, и $max(\lfloor s_j + \Delta_j \rfloor, \lceil l_j \rceil)$ в противном случае.

Операция "Строгого перемещения" (tight move или tm)

Эта операция делает удовлетворенные ограничение строже и делает разрыв между левой частью и правой частью неудовлетворенных ограничений меньше.

Пусть есть переменная x_j, ограничение con_i в котором содержится переменная x_j ($A_{ij} \neq 0$) и некоторое решение s. Пусть

$$\Delta_{ij} = (b_i - A_i s) \setminus A_{ij} \quad (2.3)$$

Оператор смешанного строгого перемещения $mtm(x_j, con_i)$ назначает переменной x_j новое значение s_j^{new}

$$s_j^{\text{new}} = \begin{cases} \min(s_j + \Delta ij, u_j) & \text{если } \Delta ij > 0, \\ \max(s_j + \Delta ij, l_j) & \text{если } \Delta ij < 0, \\ s_j & \text{в противном случае.} \end{cases} \quad (2.4)$$

Если переменная действительная.

Если переменная целочисленная:

$$s_j^{\text{new}} = \begin{cases} \min(\lceil s_j + \Delta_{ij} \rceil, \lfloor u_j \rfloor) & \text{если } \Delta_{ij} > 0 \text{ и } A_{ij} < 0, \\ \max(\lfloor s_j + \Delta_{ij} \rfloor, \lceil l_j \rceil) & \text{если } \Delta_{ij} < 0 \text{ и } A_{ij} > 0, \\ \max(\lceil s_j + \Delta_{ij} \rceil, \lceil l_j \rceil) & \text{если } \Delta_{ij} < 0 \text{ и } A_{ij} < 0, \\ \min(\lceil s_j + \Delta_{ij} \rceil, \lfloor u_j \rfloor) & \text{если } \Delta_{ij} > 0 \text{ и } A_{ij} > 0, \\ s_j & \text{в противном случае.} \end{cases} \quad (2.5)$$

Операция "Переворота" (flip move или fp)

Эта операция меняет значение бинарной переменной на противоположное, то есть $s_j^{new} = 1$ если $x_j = 0$ и $s_j^{new} = 0$ если $x_j = 1$.

Операция "Подъема" (lift move)

Эта операция улучшает значение целевой функции не нарушая допустимости решения.

Пусть у нас есть некоторое допустимое решение s. Локальным допустимым доменом $lfd(x_i, s)$ переменной будем называть интервал значений для переменной x_j такой, что при зафиксированных переменных x_k, $k \neq j$ при присваивании любого значения из этого интервала все ограничения и границы переменной будут удовлетворены.

Пусть у нас есть ограничение con_i. Локальным допустимым доменом в ограничении con_i для переменной x_j обозначается как $ldc(x_i, con_i, s)$.

Пусть $\Delta = b_i - A_i s$. Тогда, если переменная x_j – целочисленная:

$$\text{ldc}(x_j, con_i, s) = \begin{cases} [s_j + \left\lceil \frac{\Delta}{A_{ij}} \right\rceil, +\infty) & \text{если } A_{ij} < 0, \\ (-\infty, s_j + \left\lfloor \frac{\Delta}{A_{ij}} \right\rfloor] & \text{иначе.} \end{cases} \quad (2.6)$$

Если переменная x_j – действительная:

$$\text{ldc}(x_j, c_i, s) = \begin{cases} [s_j + \frac{\Delta}{A_{ij}}, +\infty) & \text{если } A_{ij} < 0, \\ (-\infty, s_j + \frac{\Delta}{A_{ij}}] & \text{иначе.} \end{cases} \quad (2.7)$$

Тогда локальный допустимый домен можно определить как:

$$lfd(x_j, s) = \bigcap_{i=1}^{m} ldc(x_j, con_i, s) \cap [l_j, u_j] \quad (2.8)$$

Операция подъема $lm(x_j, s)$ назначает переменной x_j значение на верхней грации $lfd(x_j, s)$, если $c_j < 0$, и на нижней границе в противном случае.

Случайная операция

Случайная операция выбирается из операций прорыва и о операций строгости в неудовлетворенных ограничений.

Для выбора операций Local-MIP[3] использует следующую оценочную схему:

Сначала операции оцениваются по оценке прогресса (progress score):

Пусть у нас имеется операция op и текущее решение s^{cur}. Пусть s^{op} - кандидат на новое решение, полученной с помощью операции op из текущего решения. Оценка прогресса для операции ор за улучшения значения целевой функции определяется как

$$\text{score}^{\text{obj}}_{\text{progress}}(op) = \begin{cases} w(\text{obj}) & \text{если } \text{obj}(s^{op}) < \text{obj}(s^{cur}), \\ -w(\text{obj}) & \text{если } \text{obj}(s^{op}) > \text{obj}(s^{cur}), \\ 0 & \text{иначе.} \end{cases} \quad (2.9)$$

Для ограничения con_i оценка прогресса вычисляется следующим образом:

$$\text{score}^{\text{con}_i}_{\text{progress}} = \begin{cases} w(\text{coni}) & \text{если } A_i s^{op} \le b_i < A_i s^{cur}, \\ -w(\text{coni}) & \text{если } A_i s^{cur} \le b_i < A_i s^{op}, \\ \frac{w(\text{coni})}{2} & \text{если } b_i < A_i s^{op} < A_i s^{cur}, \\ -\frac{w(\text{coni})}{2} & \text{если } b_i < A_i s^{cur} < A_i s^{op}, \\ 0 & \text{в противном случае.} \end{cases} \quad (2.10)$$

Тогда оценку прогресса для операции op можно определить как:

$$score_{progress}(op) = score^{obj}{}_{progress}(op) + \sum_{i=1}^{m} score^{con_i}{}_{progress}(op) \quad (2.11)$$

Если же есть операции с одинаковой оценкой прогресса, применяется дополнительная оценка:

Имея операцию op, текущее лучшее решение s^*, решение s^{op}, полученное при применении op на текущем решении, оценку прорыва $bonus_{break}(op)$ будем считать как:

$$\text{bonus}_{\text{break}}(op) = \begin{cases} w(\text{obj}) & \text{если } \text{obj}(s^{op}) < \text{obj}(s^*), \\ 0 & \text{в противном случае.} \end{cases} \quad (2.12)$$

Часто, при применении операций для некоторых ограничений можно получить $A_i s_{cur} = b_i$. Такие ограничения, хотя и удовлетворены, могут быть очень легко нарушены. Поэтому важно учитывать устойчивость ограничений к изменениям при выборе операций.

Имея операцию op, ограничение con_i, порожденной применением операции op га текущее решение s^{cur} решение s^{op} оценку устойчивости будем считать как

$$\text{bonus}^{\text{con}_i}_{\text{robust}}(op) = \begin{cases} w(\text{con}_i) & \text{если } A_{ij} s^{op} < b_i, \\ 0 & \text{в противном случае.} \end{cases} \quad (2.13)$$

Тогда дополнительную оценку $score_{bonus}(op)$ вычисляется как:

$$score_{bonus}(op) = bonus_{break}(op) + \sum_{i=1}^{m} bonus^{con_i}{}_{robust}(op) \quad (2.14)$$

Таким образом, операции оцениваются с помощью двухуровневой оценки. При таком подходе учитываются и локальное, и глобальное улучшения, балансируется оптимизация целевой функции и удовлетворение ограничений.

Помимо этого, Local-MIP[3] использует следующий механизм обновления весов ограничений и целевой функции:

Изначально веса всех ограничений и целевой функции (обозначаются, $w(con_i)$ и $w(obj)$ соответственно) равны 1.

Когда алгоритм не находит операцию с положительной оценкой, активируется схема обновления весов:

С вероятностью $1 - sp$, если текущее решение допустимо, $w(obj)$ увеличивается на 1; в противном случае, для каждого нарушенного ограничения con_i $w(con_i)$ увеличивается на 1.

С вероятностью sp, если $obj(s^{cur}) < obj(s^*)$ и $w(obj) > 0$, вес целевой функции уменьшается на 1; для каждого удовлетворенного ограничения con_i такого, что $w(con_i) > 0$, $w(con_i)$ уменьшается на 1.

Local-MIP[3] использует эвристику случайного выбора Best from Multiple Selections (BMS). Данная эвристика заменяет полный перебор всех возможных операций на выбор оптимального действия из случайно сформированного ограниченного подмножества. Такой подход позволяет сократить вычислительные затраты на каждой итерации, сохраняя при этом высокую вероятность выбора эффективных операций, что способствует ускорению сходимости алгоритма без существенного ухудшения качества решений.

Для предотвращения цикличности и повышения эффективности поиска в алгоритм Local-MIP[3] внедрена стратегия запрета (Tabu Strategy). Её ключевой механизм заключается во временном блокировании обратных операций, которые могли бы вернуть решение в ранее посещенные состояния. После выполнения любой операции, соответствующая ей обратная операция (например, возврат переменной к предыдущему значению) добавляется в «список запретов» (Tabu List) и остается там в течение tt итераций.

После того, как были приведены определения всех операций и оценочной схемы, можно сформулировать более подробный алгоритм работы Local-MIP[3]:

Алгоритм 1. **Алгоритм эвристики локального поиска Local-MIP.**

1. Сначала значения все переменные принимают наиболее близкие к 0 значения в пределах их границ.
2. s^* - пустое, $obj(s^*)$равно infinity
3. Пока не достигнут лимит итераций:
    a. если s^{cur} - допустимое применить операцию *lf*, если есть с такие операции с положительной оценкой. Обновить s^* если было получено значение лучше.

b. получить операцию-кандидата с наилучшей оценкой эффективности.
c. применить операцию и обновить текущее решение, активность ограничений.

4. Вернуть лучшее полученное решение s^*.

Алгоритм 2. Получение операции-кандидата

1. Если не найдено допустимое решение, то
   a. Если существует положительная *mtm* операция в нарушенных ограничениях, то
   b. candOP ← положительные *mtm* операции в нарушенных ограничениях;
2. Иначе
   a. Если существует положительная *bm* операция или *mtm* операция в нарушенных ограничениях, то
   b. candOP ← объединение положительных *bm* операций и *mtm* операций в нарушенных ограничениях;
3. Иначе, если существует положительная *mtm* операция в удовлетворенных ограничениях, то
   a. candOP ← положительные *mtm* операции в удовлетворенных ограничениях;
4. Если op == ∅, то
   a. Если существует положительная операция *fp*, то
      i. candOP ← положительные операции *fp*;
   b. Иначе
   c. Активировать схему обновления весов;
5. candOP ← *bm* операции и случайно выбранные *mtm* операции;
6. Вернуть операцию с наилучшей оценкой из candOP;

2.3 Вариации алгоритма Local-MIP

Помимо оригинальной эвристики, в работе тестировались две другие её вариации, подробнее о которых будет написано далее.

2.3.1 Local-MIP с приоритетом на бинарные переменные

В исходной эвристике[3] приоритет операций сначала отдавался операциям “строгого перемещения” *tight move*. То есть, все типы переменных, т. е. целые, дробные и бинарные, имели одинаковый приоритет. Однако, в тестовых данных для задач составления оптимального расписания более 90% переменных являлись бинарными, что определило их

критическую роль в формировании «ядра» допустимого решения. Кроме того, изменение бинарных переменных (например, выбор аудитории или временного слота) оказывает наибольшее влияние на целевую функцию.

Поэтому, новая вариация эвристики отдает приоритет операциям с бинарными переменными, то есть операциям *flip move*.

Для этой вариации эвристики алгоритм выбора операции с наивысшей оценкой выглядит следующим образом:

Алгоритм 3. Получение операции-кандидата для вариации Local-MIP, отдающей приоритет операциям с бинарными переменными

1. Если существует положительная операция *fp*, то
    a. candOP ← положительные операции булевого переключения;
2. Если op == ∅
    a. не найдено допустимое решение, то
        i. Если существует положительная *mtm* операция в нарушенных ограничениях, то
        ii. candOP ← положительные *mtm* операции в нарушенных ограничениях;
    b. Иначе
        i. Если существует положительная *bm* операция или *mtm* операция в нарушенных ограничениях, то
        ii. candOP ← объединение положительных *bm* операций и *mtm* операций в нарушенных ограничениях;
    c. Иначе, если существует положительная *mtm* операция в удовлетворенных ограничениях, то
        i. candOP ← положительные *mtm* операции в удовлетворенных ограничениях;
3. Если op == ∅, то
    a. Активировать схему обновления весов;
    b. candOP ← bm операции и случайно выбранные *mtm* операции;
4. Вернуть операцию с наилучшей оценкой из candOP.

2.3.2 Комбинация с алгоритмом имитации отжига

Несмотря на то, что у оригинальной эвристики есть механизм преодоления локального оптимума (применение случайной операции) на практике такой подход не всегда является эффективным: как видно из результатов тестирования (см. Приложение В), эвристика часто находит хорошие решения на начальных этапах, но на дальнейших этапах решения улучшаются не так быстро, как на начальных этапах.

Для повышения устойчивости алгоритма к локальным оптимумам и расширения пространства поиска решений была разработана гибридная версия, сочетающая эвристику локального поиска (Local-MIP[3]) с метаэвристикой имитации отжига.

Алгоритм имитации отжига

Алгоритм имитации отжига (Simulated Annealing) – это метаэвристический метод оптимизации, основанный на аналогии с процессом термообработки металлов, где материал постепенно охлаждается для достижения минимальной энергии. Алгоритм стартует с выбора начального решения и задания высокой начальной температуры, определяющей вероятность принятия решений с ухудшением целевой функции. На каждом шаге генерируется новое решение через модификацию текущего, после чего вычисляется изменение целевой функции. Если новое решение улучшает результат, оно автоматически принимается. В случае ухудшения решение может быть принято с вероятностью, зависящей от текущей температуры и степени ухудшения: чем выше температура, тем выше вероятность допустить временное ухудшение для выхода из локального оптимума. Температура систематически снижается по заданному закону, что постепенно уменьшает роль случайности и фокусирует поиск на локальных улучшениях. Алгоритм завершает работу при достижении минимальной температуры или достижения лимита итераций. Ключевое преимущество метода — способность балансировать между исследованием пространства решений и эксплуатацией найденных оптимумов, что делает его эффективным для задач с высокой комбинаторной сложностью, включая задачи составления расписания.

Механизм работы гибридного алгоритма

На каждой итерации поддерживается переменная температуры, инициализируемая начальным значением и монотонно уменьшающаяся в процессе работы по формуле (2.14)

$$T_{k+1} = \alpha \cdot T_k \tag{2.14}$$

где $\alpha \in (0, 1)$ – коэффициент охлаждения.

Вероятностный выбор операций осуществлялся следующим образом.

С вероятностью

$$p = 1 - e^{-T} \tag{2.15}$$

выполняется случайная операция.

С вероятностью $1 - p$ применяется жадный выбор операции с наивысшей оценкой улучшения (алгоритм 3), сохраняя приоритет бинарных переменных.

Алгоритм 4. Алгоритм комбинации эвристики локального поиска Local-MIP с приоритетом на бинарные операции и алгоритма имитации отжига

1. Сначала значения все переменные принимают наиболее близкие к 0 значения в пределах их границ.
2. s^* – пустое, $obj(s^*)$ равно infinity.
3. Пока не достигнут лимит итераций:
    a. если s^{cur} - допустимое применить операцию *lf*, если есть с такие операции с положительной оценкой. Обновить s^* если было получено значение лучше.
    b. вычислить вероятность p по формуле (2.15).
    c. получить операцию-кандидата с наилучшей оценкой эффективности с вероятностью $1 - p$ с помощью алгоритма 3, или получить случайную операцию *random tight move*.
    d. применить операцию и обновить текущее решение, активность ограничений.
    e. обновить текущую температуру T по формуле (2.14).
4. Вернуть лучшее полученное решение s^*.

2.4 Сравнительный анализ эффективности вариаций эвристики Local-MIP

В рамках экспериментального исследования проведена оценка эффективности трех реализаций алгоритма: базовой версии, модификации с приоритезацией бинарных переменных и гибридного подхода, интегрированного с методом имитации отжига.

Набор данных: Benchmark set из MIPLIB2017[4];

Оборудование, на котором проводилось тестирование: ноутбук Dell G5 5590, процессор Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GH,16,0 GB (15,8 GB usable), система 64-bit operating system, x64-based processor.

Тестирование проводилось на эвристиках с заранее вычиленным лимитом количество итераций, которые вычислялись по формуле (3.2), приведенной в следующей главе.

В результате тестирования были получены следующие результаты:

Таблица 2.1 – Сравнение результатов работы оригинальной эвристики Local-MIP и эвристики с приоритетом на бинарные переменные

Критерий	Local-MIP	bin-Local-MIP
Найдено допустимых решений	123	133
Найдено оптимальных решений	9	7
Итоговое решение лучше, чем у конкурента	53	72

Колонка Local-MIP показывает результаты оригинальной эвристики[3]. bin-Local-MIP – эвристика с измененным приоритетом в пользу операций с бинарными переменными.

Как видно из таблицы, изменения приоритета операций имело положительный эффект на эвристику: несмотря на то, что количество задач, в которых эвристика нашла оптимальное решение, упало, количество задач, в которых было найдено допустимое решение, так же как и качество этих допустимых решений, возросло.

Гибрид эвристики Local-MIP[3] с алгоритмом имитации отжига был протестирован на различных начальных температурах и степенях охлаждения. Результаты тестирования были следующими:

Таблица 2.2 – Количество задач, в которых гибридная эвристика нашла допустимые решения в зависимости от начальной температуры T и коэффициента охлаждения α

T, α	0.93	0.95	0.98
100	128 8	127 6	126 8
200	127 6	127 7	125 6
300	126 7	130 7	125 5

Таблица 2.3 – Количество задач, в которых гибридная эвристика нашла оптимальные решения, в зависимости от начальной температуры T и коэффициента охлаждения α

T, α	0.93	0.95	0.98
100	128 8	127 6	126 8
200	127 6	127 7	125 6
300	126 7	130 7	125 5

На основе полученных результатов можно сделать следующие выводы:

- Высокая начальная температура (300) увеличивает вероятность нахождения допустимых решений.
- Медленное охлаждение (коэффициент 0.98) улучшает поиск оптимальных решений, но требует больше итераций.
- Умеренное охлаждение (коэффициент 0.95) обеспечивает баланс между скоростью и качеством.

Учитывая тот факт, что в дальнейшем эвристика будет использоваться для получения прямой оценки целевой функции на начальных этапах решения, для дальнейшего использования гибридной эвристики были выбраны параметры, на которых эвристика находила большее количество допустимых решений, то есть, температура $T = 300$, коэффициент охлаждения $\alpha = 0.95$.

Проведя сравнительный анализ результатов гибридной эвристики н упомянутых выше парамерах, с результатами оригинальной эвристики и

эвристики с измененным приоритетом операций, были получены результаты, описанные в таблицах 2.4 и 2.5.

Таблица 2.4 – Сравнение результатов работы оригинальной эвристики Local-MIP и эвристики-гибрида Local-MIP с имитацией отжига

Критерий	Local-MIP	гибрид Local-MIP с имитаций отжига
Найдено допустимых решений	123	130
Найдено оптимальных решений	9	7
Итоговое решение лучше, чем у конкурента	84	40

Таблица 2.5 – Сравнение результатов работы эвристики Local-MIP с измененным приоритетом в пользу бинарных переменных и эвристики-гибрида Local-MIP с имитацией отжига

Критерий	bin-Local-MIP	гибрид Local-MIP с имитаций отжига
Найдено допустимых решений	133	130
Найдено оптимальных решений	7	7
Итоговое решение лучше, чем у конкурента	78	42

Проведенный вычислительный эксперимент продемонстрировал, что модификация Local-MIP[3] с приоритезацией бинарных операций (bin-Local-MIP) является наиболее эффективной версией алгоритма для задач составления расписания. Она обеспечивает максимальное количество допустимых решений (133), существенно превосходя оригинальный алгоритм (123) и гибридный подход (130). Это подтверждает, что фокус на бинарных переменных, играющих ключевую роль в структурной организации расписания, позволяет значительно повысить эффективность поиска.

Гибридный алгоритм, интегрирующий имитацию отжига, хотя и уступает в качестве решений (7 оптимальных против 9 у оригинала), демонстрирует улучшенную устойчивость к локальным оптимумам и большую гибкость в сложных сценариях. Его способность находить

допустимые решения в большем числе случаев (130 против 123 у оригинала) делает его ценным инструментом для задач с жёсткими ограничениями, где допустимость критически важна.

ГЛАВА 3. ИНТЕГРАЦИЯ ЭВРИСТИКИ ЛОКАЛЬНОГО ПОИСКА В РЕШАТЕЛЬ. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

В данной главе рассматривается процесс интеграции эвристики локального поиска Local-MIP[3] в солвер HiGHS[2], который был разработан с учетом современных методов оптимизации.

3.1 Процесс интеграции

HiGHS[2] и Local-MIP[3] используют различные модели представления задач. В HiGHS[2] параметры задачи хранятся в виде векторов: коэффициенты целевой функции (c), смещение (d), границы переменных (l, u) и ограничений (L, U). Матрица коэффициентов ограничений A представлена в упакованном векторном формате, где индексы и значения ненулевых элементов хранятся в массивах a_{index} и a_{value}, а позиции первых элементов строк/столбцов — в a_{start}.

Local-MIP[3] оперирует ограничениями вида $Ax \leq b$, используя специализированные структуры для хранения метаданных (имена, типы, границы переменных и ограничений) и текущих состояний (значения переменных, веса ограничений). Различия в архитектурах потребовали разработки класса-адаптера, преобразующего модель из формата HiGHS[2] в формат Local-MIP[3]. Это обеспечило совместимость систем.

Loca-MIP был интегрирован после пресолва в процесс перед решением корневого узла с расчетом на то, что хорошее допустимое решение на начальных этапах поможет солверу решить задачу быстрее.

Оптимизация количества операций

В отличие от стандартного подхода Local-MIP[3], где время выполнения ограничивается заранее заданным ограничением времени, в рамках интеграции применено ограничение числа итераций. Это повысило предсказуемость работы алгоритма.

Для оценки зависимости длительности итерации от сложности задачи проведены эксперименты с использованием набора MIPLIB2017[4].

На графике представлены данные, иллюстрирующие среднее время выполнения одной итерации эвристики в зависимости от количества ненулевых коэффициентов в матрице A. Каждая точка на графике соответствует конкретной задаче, что позволяет визуализировать взаимосвязь между сложностью задачи и временем выполнения итерации.

Красная линия представляет собой функцию, полученную в результате работы линейной регрессии, демонстрируя общую тенденцию изменения времени выполнения с увеличением числа ненулевых коэффициентов.

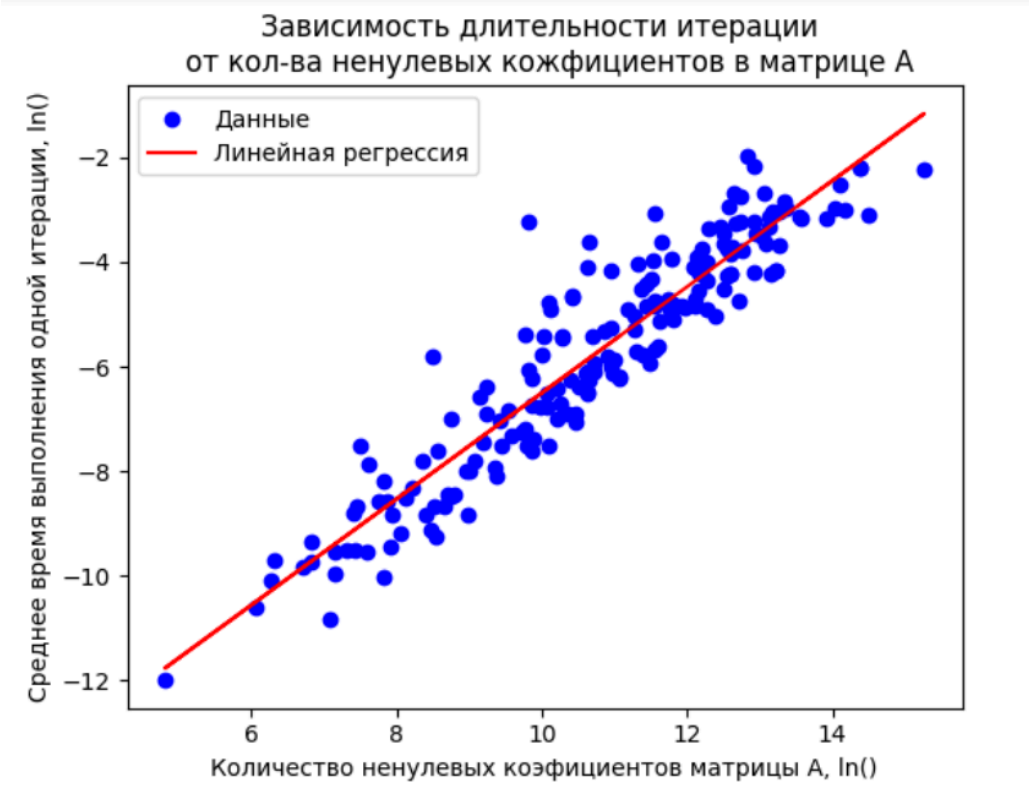

Рисунок 3.1 – Оценка длительности итерации в зависимости от количества ненулевых коэффициентов в матрице A модели, полученная с помощью линейной регрессии

Была получена функция, оценивающая нужное количество итераций для того, чтобы эвристика работала время t:

$$N(t,\ Nonz)\ =\ t\ /\ exp(C\ ln(Nonz)\ +\ Sh) \qquad (3.1)$$

где

$C\ =\ 1.01434917,$

$Sh\ =-\ 16.645264280936566$

– коэффициенты полученные с помощью регрессии;

$Nonz$ – количество ненулевых коэффициентов матрицы A.

В итоговой версии алгоритма был использован следующий вариант этой функции:

$$N\ (Nonz)\ =\ exp(-\ C\ ln(max(Nonz\ ,\ 500))\ -\ Sh)\ *\ 10 \qquad (3.2)$$

$max(Nonz\ ,\ 500)$ было добавлено во избежание слишком большого количества итераций для эвристики в тех случаях, тогда количество ненулевых элементов в матрице A достаточно мало.

3.2 Вычислительный эксперимент

Набор данных: Benchmark set из MIPLIB2017[4];

Оборудование, на котором проводилось тестирование: ноутбук Dell G5 5590, процессор Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GH,16,0 GB (15,8 GB usable), система 64-bit operating system, x64-based processor.

Тестирование проводилось на модифицированной версии солвера HiGHS[2] с интегрированной эвристикой Local-MIP[3]. Для оценки влияния алгоритма на производительность использовались четыре конфигурации:

Оригинальный HiGHS – эвристика отключена;

HiGHS с Local-MIP – эвристика активна, используется оригинальная версия алгоритма.

HiGHS с bin-Local-MIP – эвристика активна, испоьзуется версия эвристики с измененным приоритетом операций

HiGHS с sima-Local-MIP (N_{mid}) — эвристика активна, используется гибрид с алгоритмом имитации отжига.

Во всех тестах установлены одинаковые параметры: ограничение времени выполнения – 600 секунд, ограничение количества поток равно 1, для остальных параметров были использованы настройки по умолчанию.

В результате тестирования были получены следующие результаты:

Таблица 3.1 – Сравнение эффективности разных версий HiGHS

Критерий сравнения	HiGHS	HiGHS с оригинальным Local-MIP	Highs с bin-Local-MIP	HiGHS с sima-Local-MIP
Количество решенных задач	15	15	15	15
Среднее время решения	188.71	194.16	192.25	188.47
Среднее время решения (только решенные)	51.61	58.88	56.34	51.30
Количество задач, в которых Local-MIP нашел допустимое решение	-	8	8	7

Для всех конфигураций решателя, для которых проводилось в тестирование в этом вычислительном эксперименте, находились оптимальные решение до истечения времени, кроме следующих: usched-big-sd, usched-big, usched-big-dd, usched-big-csun, usched-hard-pos-inf. Эти задачи характеризуются большим количеством переменных и ограничений и малым количеством заранее зафиксированных переменных.

Ввиду полученных результатов, можно сделать следующие выводы:

- Решатель HiGHS[2], независимо от конфигурации, способен решать задачи составления расписания малых и средних масштабов ($\leq$2 млн. коэффициентов в матрице ограничений) достаточно быстро (менее, чем за 10 минут), что делает его эффективным инструментом для решения задач такого рода.
- Интеграция эвристики в солвер в среднем не привела к значительному

увеличению эффективности поиска оптимального решения, однако почти в половине случаев все версии эвристики находили допустимые решения. Как видно из Приложения Б эвристика сильно улучшает время решения для 4-5 задач независимо от конфигурации, из чего можно сделать вывод, что интеграция эвристик имеет положительный результат для решателя.

- При этом лучшие показатели были достигнуты для HiGHS[2] с интегрированным гибридом алгоритма имитации отжига и Local-MIP[3].

ЗАКЛЮЧЕНИЕ

Проведённое исследование позволило разработать и апробировать комплексный подход к решению задачи составления учебных расписаний с использованием современных методов оптимизации. Решатель HiGHS [2] подтвердил свою эффективность в обработке задач малого и среднего масштаба, демонстрируя стабильное время решения в пределах 10 минут, что делает его практичным инструментом для решения задач составления расписания учебных занятий. Интеграция эвристических методов, включая модифицированный алгоритм Local-MIP[3] и гибридный подход с имитацией отжига, позволила повысить устойчивость решателя к локальным оптимумам и расширить спектр находимых допустимых решений.

Ключевые результаты работы:

1. Была разработана математическая модель, учитывающая реальные требования к учебным расписаниям, включая ограничения по аудиториям, преподавателям и группам.

2. Комбинация HiGHS с гибридной эвристикой продемонстрировала наилучшие показатели, сокращая время решения для 15% задач и улучшая качество расписаний.

3. Несмотря на ограниченное влияние на поиск оптимальных решений, эвристики обеспечили нахождение допустимых решений в 48% случаев, что критически важно для задач с жесткими временными ограничениями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A mixed-integer programming approach for solving university course timetabling problems. / E. Rappos [et al]. // J. Sched. – 2022. – № 25. – P. 391–404.
2. Huangfu, Q. Parallelizing the dual revised simplex method. / Q. Huangfu, J.A.J. Hall // Math. Prog. Comp. – № 10. – 2018. – P. 119–142.
3. Lin, P. An Efficient Local Search Solver for Mixed Integer Programming. / P. Lin, M. Zou, S. Cai // In: 30th International Conference on Principles and Practice of Constraint Programming (CP 2024). – Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl – Leibniz-Zentrum für Informatik – 2024. – Vol. 307. – Pp. 19:1-19:19.
4. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. / A. Gleixner [et al]. // Math. Prog. Comp. – № 13. – 2021. – P. 443–490.

ПРИЛОЖЕНИЕ А

ИНФОРМАЦИЯ О СГЕНЕРИРОВАННЫХ ЗАДАЧАХ СОСТАВЛЕНИЯ РАСПИСАНИЯ

Таблица А.1 – Информация о переменных и ограничениях исходных сгенерированных задач составления расписания

Название	Кол-во переменных	Кол-во бинарных переменных	Кол-во ограничений	Кол-во ненулевых коэффициентов матрицы ограничений
usched-big	1151584	1151056	665790	5597274
usched-big-csun	1151584	1151056	565300	6046724
usched-big-dd	1151644	1151096	585350	6516854
usched-big-sd	1151584	1151056	569230	6650574
usched-big-sd	1151584	1151056	569230	6650574
usched-big-sqftst	1151584	1151056	618345	6025049
usched-easy	100660	100440	54700	399760
usched-easy-all	52905	52770	31269	273918
usched-easy-sdddftst	100546	100364	49989	505172
usched-easy-sqsdun	120360	120140	68816	594352
uschef-hard-pos-inf	1252715	1251810	686985	7169315
usched-inter-all	158049	157866	87072	938251
usched-inter-ftun	207728	207552	112671	933794
usched-inter-sqsdcsddftun	140945	140630	82555	748907
usched-inter-sqsdddftst	129378	129152	74021	658991
usched-medium	388156	387804	218202	1536412
usched-medium-sd	401056	400704	214905	2129116
usched-medium-sddd	703128	702752	346283	3884142

Продолжение таблицы А.1

usched-medium-sdun	401056	400704	212896	2082116
usched-medium-sqsdft	388156	387804	216543	1830250
usched-medium-sqst	401056	400704	204813	2119416

Таблица А.2 – Данные о входных размерах задач, типах и количествах использованных ограничений

Название	C	P	R	G	Использованные дополнительные ограничения:
usched-big	200	75	40	12	(1.11.2) - 19800 (1.12.2) - 202750 (1.16) - 420500
usched-big-csun	200	75	40	12	(1.11.2) - 22200 (1.12.2) - 205850 (1.15) - 40 (1.16) - 314450 (1.17)- 10
usched-big-dd	200	75	40	12	(1.9) - 20 (1.11.2) - 16200 (1.12.2) - 199550 (1.15) - 40 (1.16) - 346700
usched-big-sd	200	75	40	12	(1.8) - 100 (1.11.2) - 21600 (1.12.2) - 188050 (1.15) - 40 (1.16) - 336700
usched-big-sqftst	200	75	40	12	(1.7) - 10 (1.10) - 1 (1.11.2) - 18800 (1.12.2) - 203650 (1.13) - 4 (1.15) - 40 (1.16) - 373100
usched-easy	50	20	20	5	(1.11.2) - 2000 (1.12.2) - 21750 (1.16) - 24250

Продолжение таблицы А.2

usched-easy-all	30	15	20	3	(1.7) - 3 (1.8) - 3 (1.9) - 3 (1.10) - 3 (1.11.2) - 1440 (1.12.2) - 16050 (1.13) - 6 (1.15) - 20 (1.16) - 9000 (1.17) - 3
usched-easy-sdddftst	40	20	30	4	(1.8) - 6 (1.9) - 6 (1.10) - 3 (1.11.2) - 2800 (1.12.2) - 29600 (1.13) - 4 (1.15) - 30 (1.16) - 11050
usched-easy-sqsdun	42	33	24	5	(1.7) - 8 (1.8) - 8 (1.11.2) - 2142 (1.12.2) - 20250 (1.15) - 24 (1.16) - 38600
usched-hard-pos-inf	250	50	50	20	(1.8) - 25 (1.9) - 25 (1.11.2) - 28000 (1.12.2) - 310850 (1.13) - 10 (1.15) - 50 (1.16) - 322700 (1.17) - 25
usched-inter-ftun	74	34	22	4	(1.10) - 5 (1.11.2) - 4366 (1.12.2) - 43900 (1.15) - 22 (1.16) - 55150

Продолжение таблицы А.2

usched-inter-all	70	25	20	4	(1.7) - 7 (1.8) - 7 (1.9) - 7 (1.10) - 8 (1.11.2) - 3080 (1.12.2) - 36050 (1.13) - 4 (1.15) - 20 (1.16) - 39750 (1.17) - 7
usched-inter-sqsdcsddftun	70	20	20	7	(1.7) - 7 (1.8) - 7 (1.9) - 7 (1.10) - 5 (1.11.2) - 3430 (1.12.2) - 34700 (1.15) - 20 (1.16) - 36150 (1.17) - 7
usched-inter-sqsdddftst	66	21	18	5	(1.7) - 6 (1.8) - 6 (1.9) - 6 (1.10) - 3 (1.11.2) - 3564 (1.12.2) - 29550 (1.13) - 6 (1.15) - 18 (1.16) - 33300
usched-medium	98	34	45	8	(1.11.2) - 9996 (1.12.2) - 111250 (1.16) - 84450
usched-medium-sd	100	50	30	8	(1.8) - 15 (1.11.2) - 5700 (1.12.2) - 71350 (1.15) - 30 (1.16) - 124250

Продолжение таблицы А.2

usched-medium-sddd	120	62	55	8	(1.8) - 15 (1.11.2) - 5700 (1.12.2) - 71350 (1.15) - 30 (1.16) - 124250
usched-medium-sdun	100	50	30	8	(1.8) - 6 (1.11.2) - 8200 (1.12.2) - 74750 (1.15) - 30 (1.16) - 116350
usched-medium-sqsdft	98	34	45	8	(1.7) - 11 (1.8) - 11 (1.10) - 8 (1.11.2) - 11662 (1.12.2) - 112300 (1.15) - 45 (1.16) - 80000
usched-medium-sqst	100	50	30	8	(1.7) - 15 (1.11.2) - 7700 (1.12.2) - 72000 (1.13) - 8 (1.15) - 30 (1.16) - 111500

ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТАТЫ ВЫЧИСЛИТЕЛЬНОГО ЭКСПЕРИМЕНТА С РАЗНЫМИ КОНФИГУРАЦИЯМИ HIGHS

Таблица Б.1 Подробные результаты вычислительного эксперимента с разными конфигурациями HiGHS

	Default HiGHS		Default LMIP			Bin Local-MIP			sima LMIP		
Название	статус	время	feas	статус	время	feas	статус	время	feas	статус	время
usched-big	TLR	600.0		TLR	600.0		TLR	600.0		TLR	600.0
usched-big-csun	TLR	600.0		TLR	600.0		TLR	600.0		TLR	600.0
usched-big-dd	TLR	600.0		TLR	600.0		TLR	600.0		TLR	600.0
usched-big-sd	OPT	215.22		TLR	600.0		TLR	600.0		TLR	600.0
usched-big-sqftst	TLR	600.0		TLR	600.0		OPT	276.29		OPT	273.61
usched-easy	OPT	6.23	Y	OPT	7.28	Y	OPT	6.58	Y	OPT	5.7
usched-easy-all	OPT	23.28	Y	OPT	17.86	Y	OPT	16.43	Y	OPT	16.52
usched-easy-sdddftst	OPT	8.78	Y	OPT	9.81	Y	OPT	9.21	Y	OPT	8.49
usched-easy-sqsdun	OPT	29.65	Y	OPT	27.13	Y	OPT	26.07	Y	OPT	26.27
usched-hard-pos-inf	TLR	600.0		TLR	600.81		TLR	600.0		TLR	600.0
usched-inter-all	OPT	50.91	Y	OPT	31.71	Y	OPT	28.34		OPT	31.14
usched-inter-ftun	OPT	07.07	Y	OPT	8.12	Y	OPT	7.45	Y	OPT	7.43
usched-inter-sqsdcsddftun	OPT	27.71		OPT	26.09	Y	OPT	70.52	Y	OPT	50.39
usched-inter-sqsdddftst	OPT	30.53	Y	OPT	19.34	Y	OPT	27.36	Y	OPT	27.04
usched-medium	OPT	12.6	Y	OPT	13.19		OPT	12.28		OPT	13.08

Продолжение таблицы Б.1.

usched-medium-sd	OPT	40.29		OPT	34.18		OPT	37.02		OPT	35.09
usched-medium-sddd	OPT	112.62		TLR	600.0		OPT	113.94		OPT	112.84
usched-medium-sdun	OPT	42.56		OPT	38.2		OPT	31.16		OPT	22.05
usched-medium-sqsdft	OPT	79.63		OPT	71.08		OPT	87.55		OPT	65.56
usched-medium-sqst	OPT	87.13		OPT	83.55		OPT	94.9		OPT	74.35

Таблица Б.1 описывает детали результатов вычислительного эксперимента, проведенного в главе 3.

Колонка «Статус» фиксирует конечное состояние работы решателя. Значение OPT (Optimal) указывает на успешное нахождение и доказательство оптимальности решения, что является наивысшим результатом. В случаях, когда решатель не успевает завершить поиск в установленный временной лимит, статус отмечается как TLR (Time Limit Reached), что означает прерывание процесса до достижения гарантированной оптимальности.

Колонка «Время» содержит суммарное время выполнения задачи, включая этапы чтения входных данных, предобработки модели (presolve) и непосредственного поиска решения.

Для конфигураций, использующих эвристики локального поиска, добавлена колонка «Feas», которая отражает успешность их применения. Значение Y (Yes) означает, что эвристика смогла найти хотя бы одно допустимое решение в рамках выделенного времени. Пустые ячейки в этой колонке свидетельствуют о том, что эвристика не обнаружила допустимых решений.

ПРИЛОЖЕНИЕ В

СРАВНЕНИЯ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ РЕШАТЕЛЯ HIGHS И ЭВРИСТИКИ LOCAL-MIP

Таблица В.1 – Количество допустимых решений, найденных Local-MIP и HiGHS за 1, 10, 50, 100, 300 секунд.

	Время				
	1c	10c	50c	100c	300c
Local-MIP	102	152	166	171	182
HiGHS	53	99	145	161	177

Таблица В.2 – Количество задач, для которых один решатель обнаружил лучшие решения по сравнению с другим.

	Время				
	1c	10c	50c	100c	300c
Local-MIP	74	99	75	57	44
HiGHS	31	62	111	130	150

В Таблице В.1 представлены сравнительные данные о количестве задач, для которых были найдены допустимые решения эвристическим алгоритм и решателем в различные временные интервалы.

Таблица 2 иллюстрирует количество задач, для которых один решатель нашел лучшие решения по сравнению с другим в зависимости от времени, отведенного на решение.

Ресурсы, с помощью которых проводилось тестирование, результаты которого описаны в таблицах В.1 и В.2:

Набор данных: Benchmark set из MIPLIB2017[4];

Оборудование, на котором проводилось тестирование: ноутбук Dell G5 5590, процессор Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GH,16,0 GB (15,8 GB usable), система 64-bit operating system, x64-based processor.

Во всех экспериментах HiGHS[2] тестировался с параметрами по умолчанию, кроме количества используемых потоков (1), времени (5 минут).

ПРИЛОЖЕНИЕ Г

ПРОГРАММНЫЙ КОД ГЕНЕРАЦИИ ЗАДАЧ СОСТАВЛЕНИЯ ОПТИМАЛЬНОГО РАСПИСАНИЯ

```
# %% [markdown]
# # Создание модели

# %%
from pulp import LpProblem, LpVariable, LpBinary, LpInteger, LpMaximize,
lpSum
import random
C_num = 50 #number of classes
P_num = 20 #number of professors
R_num = 20 #number of classrooms
T_num = 50 #number of time slots
G_num = 5  # number of groups

# %%
# Пример данных (замените на реальные)
C = [f"c{i}" for i in range(1, C_num + 1)]        # 100 занятий
P = [f"p{i}" for i in range(1, P_num + 1)]        # 50 преподавателей
R = [f"r{i}" for i in range(1, R_num + 1)]        # 30 аудиторий
G = [f"g{i}" for i in range(1, G_num + 1)]
T = list(range(1, T_num + 1))                     # 50 временных слотов

# %%
ready = True
mps_file = ""
# %%
slot_per_day = [9, 9, 9, 9, 9, 5]
days = 6

# %%
C_qualification = {}
for c in C:
    num_teachers = random.randint(1, len(P))  # Каждое занятие могут вести
от 1 до всех преподавателей
    C_qualification[c] = random.sample(P, num_teachers)

# %%
P_qualification = {}
for c, teachers in C_qualification.items():
    for teacher in teachers:
        if teacher not in P_qualification:
```

```
            P_qualification[teacher] = []  # Инициализируем список, если
преподаватель еще не добавлен
        P_qualification[teacher].append(c)

# %%
# Случайно задаем недоступные слоты для преподавателей
unavailable_slots = {
    p: random.sample(T, k=random.randint(0, 5))  # 0-5 случайных
недоступных слотов
    for p in P
}
# %%
use_equipment_constr = True
use_qualification_constr = True
use_anavailable_slots_constr = True
# дополнительные ограничения
use_additional_constr = False

use_sequence_constr = True #sq
use_same_day_constr = True #sd
use_consecutive_constr = False #cs
use_different_day_constr = False #dd
use_fixed_time_asgn_constr = False #ft
use_same_teacher_asignment = False #st

# %%
constraint_percentage = 0.2
def generate_all_pairs():
    return [(c1, c2) for c1 in C for c2 in C if c1 != c2]

# Генерация данных
def generate_unique_pairs(use_constraint, total_pairs, num_pairs):
    if use_constraint and num_pairs > 0:
        return random.sample(total_pairs, k=num_pairs)
    return []

# Генерация всех пар занятий
all_pairs = generate_all_pairs()

# Определяем количество пар на основе процента
num_pairs = int(constraint_percentage * len(C))

# Генерация пар для ограничений
```

```
sequence_pairs = generate_unique_pairs(use_sequence_constr, all_pairs,
num_pairs)
remaining_pairs = [pair for pair in all_pairs if pair not in
sequence_pairs]

same_day_pairs = generate_unique_pairs(use_same_day_constr,
remaining_pairs, num_pairs)
remaining_pairs = [pair for pair in remaining_pairs if pair not in
same_day_pairs]

consecutive_pairs = generate_unique_pairs(use_consecutive_constr,
remaining_pairs, num_pairs)
remaining_pairs = [pair for pair in remaining_pairs if pair not in
consecutive_pairs]

different_day_pairs = generate_unique_pairs(use_different_day_constr,
remaining_pairs, num_pairs)

if use_fixed_time_asgn_constr:
    fixed_time_assignments = [(random.choice(C), random.choice(T)) for _
in range(random.randint(1, len(C) // 5))]
else:
    fixed_time_assignments = []

if use_unavailable_constr:
    unavailable_teachers = {p: random.sample(range(1, T_num + 1),
random.randint(1, T_num // 4)) for p in random.sample(P, random.randint(1,
len(P) // 5))}
else:
    unavailable_teachers = {}

if use_same_teacher_asignment:
    # Выбираем % преподавателей и для каждого выбираем пару занятий
    num_teachers = random.randint(1, 5)  # Количество преподавателей для
выбора
    same_teacher_assignments = []
    selected_teachers = random.sample(P, num_teachers)  # Случайный выбор
преподавателей
    for teacher in selected_teachers:
        if len(P_qualification[teacher]) >= 2:  # Убедимся, что у
преподавателя достаточно занятий
            # Выбираем пару занятий из квалификаций преподавателя
            assigned_classes = random.sample(P_qualification[teacher], 2)
            same_teacher_assignments.append((teacher, assigned_classes))
```

```
else:
    same_teacher_assignments = []

Z_p = 40 # ограничение по нагрузке
# %%
from enum import Enum
class Equipment(Enum):
    PROJECTOR = 1
    COMPUTER = 2
    LABORATORY = 3

# %%
def get_day(slot_id):
    return (slot_id - 1) // 9 + 1

# %%
def get_day_slot(slot_id):
    return (slot_id - 1) % 9 + 1

# %%
def get_slot_id(day, day_slot):
    return (day - 1) * 9 + day_slot

# %% [markdown]
# ### Переменные

# %%

# Создаем модель
model = LpProblem("University_Scheduling_Large", LpMaximize)

# Генерация переменных через LpVariable.dicts()

# 1. Переменные a_{c,t,r} (занятие c в слоте t в аудитории r)
a_indices = [(c, t, r) for c in C for t in T for r in R]
a = LpVariable.dicts(
    name="a",
    indices=a_indices,
    cat=LpBinary
)

# 2. Переменные z_{p,t,c} (преподаватель p ведет занятие c в слоте t)
z_indices = [(p, t, c) for p in P for t in T for c in C]
z = LpVariable.dicts(
```

```
    name="z",
    indices=z_indices,
    cat=LpBinary
)

# %% [markdown]
# ### Вместимость аудиторий и оборудование

# %%
C_space = {c: random.randint(20, 50) for c in C}  # Студентов на занятии
R_space = {r: random.randint(30, 80) for r in R}  # Вместимость аудитории

# %%

# Оборудование занятий и аудиторий
C_eq = {}
for c in C:
    num_eq = random.randint(0, len(Equipment))
    C_eq[c] = set(random.sample(list(Equipment), num_eq))

R_eq = {}
for r in R:
    num_eq = random.randint(0, len(Equipment))
    R_eq[r] = set(random.sample(list(Equipment), num_eq))

# %%
if use_equipment_constr:
    for c in C:
        required_equipment = C_eq[c]
        for r in R:
            available_equipment = R_eq[r]
            # Если аудитория не подходит по оборудованию
            if not required_equipment.issubset(available_equipment):
                for t in T:
                    model += (
                        a[(c, t, r)] == 0,
                        f"Equipment_Mismatch_{c}_{r}_{t}"
                    )

# %%
for t in T:
    for r in R:
        model += (
            lpSum(C_space[c] * a[(c, t, r)] for c in C) <= R_space[r],
```

```
            f"Capacity_{r}_{t}"
        )

# %% [markdown]
# ### Квалификация преподавателей и недоступное для них время

# %%

if use_qualification_constr:
    for c in C:
        allowed_teachers = C_qualification[c]
        for p in P:
            if p not in allowed_teachers:
                for t in T:
                    model += (
                        z[(p, t, c)] == 0,
                        f"Qualification_Mismatch_{p}_{c}_{t}"
                    )

# %%
if use_anavailable_slots_constr:
    for p in P:
        busy_slots = unavailable_slots[p]
        for t in busy_slots:
            for c in C:
                model += (
                    z[(p, t, c)] == 0,
                    f"Unavailable_Slot_{p}_{t}_{c}"
                )

# %% [markdown]
# ### Синхронизация переменных z и a

# %%
for c in C:
    for t in T:
        model += (
            lpSum(a[(c, t, r)] for r in R) == lpSum(z[(p, t, c)] for p in
P),
            f"Time_Sync_{c}_{t}"
        )


# %% [markdown]
```

```
# ## Основные ограничения

# %% [markdown]
#  - Каждое заданиятие должно проходить в одно время
#  - Каждому занятию должна быть предоставлена аудитория и только одна
#  - Каждому занятию должен быть назначен преподаватель
#  - Никакие две пары не должны проходить в одной и той же аудитории в
одно и то же время и/или с одним и тем же преподавателем

# %%
# 1. Каждое занятие проходит в одно время
# 2. Каждое занятие в одной аудитории (одинаковые?)
for c in C:
    model += (
        lpSum(a[c, t, r] for t in T for r in R) == 1,
        f"Single_Time_Assignment_{c}"
    )

# 3. Каждому занятию назначен преподаватель
for c in C:
    model += (
        lpSum(z[p, t, c] for p in P for t in T) == 1,
        f"Single_Teacher_Assignment_{c}"
    )

# 4.1 Конфликты аудиторий
for t in T:
    for r in R:
        model += (
            lpSum(a[c, t, r] for c in C) <= 1,
            f"Room_Conflict_{t}_{r}"
        )

# 4.2 Конфликты преподавателей
for p in P:
    for t in T:
        model += (
            lpSum(z[p, t, c] for c in C) <= 1,
            f"Teacher_Conflict_{p}_{t}"
        )
```

```
# %% [markdown]
# ## Дополнительные ограничения

# %% [markdown]
#  - последовательность занятий
#     - раньше/позже
#     - в один день
#     - сразу друг за другом
#     - в разные дни
#     - определенное время
#     - исключить недоступное время
#  - недоступность преподавателя
#     - один и тот же преподаватель для занятий
#     - нагрузка преподавателей

# %%
if use_additional_constr:
    # 2.1 Раньше/Позже
    for c1, c2 in sequence_pairs:  # sequence_pairs — список пар (c1, c2),
которые нужно упорядочить
        model += (
            lpSum(t * a[c1, t, r] for t in T for r in R) <= lpSum(t *
a[c2, t, r] for t in T for r in R) - 1,
            f"Order_{c1}_before_{c2}"
        )

    slot_to_day = {}
    t = 1
    # Дни 1-5 (пн-пт) по 9 слотов
    for day in range(1, 6):
        for _ in range(9):
            slot_to_day[t] = day
            t += 1
    # День 6 (сб) — 4 слота
    for _ in range(5):
        slot_to_day[t] = 6
        t += 1

    # 2.2 В один день
    for c1, c2 in same_day_pairs:
        model += (
            lpSum(get_day(t) * a[c1, t, r] for t in T for r in R)
            == lpSum(get_day(t) * a[c2, t, r] for t in T for r in R),
```

```
                f"Same_Day_{c1}_{c2}"
            )

        # 2.3 В разные дни
        U = 1
        for c1, c2 in different_day_pairs:
            day_c1 = lpSum(slot_to_day[t] * a[c1, t, r] for t in T for r in R)
            day_c2 = lpSum(slot_to_day[t] * a[c2, t, r] for t in T for r in R)

            # Создание переменных для каждой пары
            day_dif = LpVariable(f"day_dif_{c1}_{c2}", lowBound=0,
cat=LpInteger)  # Переменная для |day_c1 - day_c2|
            day_dif_d1 = LpVariable(f"day_dif_d1_{c1}_{c2}", cat=LpBinary)  #
1, если day_c1 - day_c2 >= 0
            day_dif_d2 = LpVariable(f"day_dif_d2_{c1}_{c2}", cat=LpBinary)  #
1, если day_c2 - day_c1 >= 0

            # Ограничения для линеаризации
            model += (day_dif - (day_c1 - day_c2) <= 2 * slot_to_day[t] *
day_dif_d2, f"Pos_Day_Diff_Constraint_{c1}_{c2}_1")
            model += (day_dif - (day_c2 - day_c1) <= 2 * slot_to_day[t] *
day_dif_d1, f"Pos_Day_Diff_Constraint_{c1}_{c2}_2")
            model += (day_dif - (day_c1 - day_c2) >= 0,
f"Neg_Day_Diff_Constraint_{c1}_{c2}_1")
            model += (day_dif - (day_c2 - day_c1) >= 0,
f"Neg_Day_Diff_Constraint_{c1}_{c2}_2")

            # Условие, что только одна из бинарных переменных может быть равна
1
            model += (day_dif_d1 + day_dif_d2 == 1,
f"Binary_Day_Diff_Constraint_{c1}_{c2}")

            # Ограничение на минимальную разность
            model += (day_dif >= 1, f"Min_Day_Diff_Constraint_{c1}_{c2}")

            #model += (day_c2 - day_c1 >= 1, f"Different_Day_{c1}_{c2}_2") //

        # 2.4 Друг за другом в один день
        for c1, c2 in consecutive_pairs:
            # Создаем переменные для времени занятий
            time_c1 = lpSum(t * a[c1, t, r] for t in T for r in R)
            time_c2 = lpSum(t * a[c2, t, r] for t in T for r in R)
            # Проверяем, что c2 идет сразу после c1 и в тот же день
            model += (time_c2 - time_c1 == 1, f"Consecutive_Time_{c1}_{c2}")
```

```
            model += (
                lpSum(slot_to_day[t] * a[c1, t, r] for t in T for r in R)
                == lpSum(slot_to_day[t] * a[c2, t, r] for t in T for r in R),
                f"Consecutive_Same_Day_{c1}_{c2}"
            )

    # 2.5 В определенное время (аналогично для недоступного времени)
    for c, t_target in fixed_time_assignments:  # Список пар (c, t_target)
        model += (
            lpSum(a[c, t_target, r] for r in R) == 1,
            f"Fixed_Time_{c}_{t_target}"
        )

    # 3.1 уже задали до этого
    # 3.2 Один преподаватель на несколько занятий
    for teacher, assigned_classes in same_teacher_assignments:  # Список
(p, [c1, c2, ...])
        for c in assigned_classes:
            model += (
                lpSum(z[teacher, t, c] for t in T) == 1,
                f"Same_Teacher_{teacher}_{c}"
            )

    # 3.3 Нагрузка преподавателей
    teacher_max_hours = {p: Z_p for p in P}  # Максимальное количество
часов для каждого преподавателя
    for p in P:
        model += (
            lpSum(z[p, t, c] for t in T for c in C) <=
teacher_max_hours[p],
            f"Teacher_Workload_{p}"
        )

# %% [markdown]
# ## Целевая функция

# %% [markdown]
# ### Штрафы за пробелы в расписании для студентов

# %%

group_classes = {g: [] for g in G}
for c in C:
    assigned_groups = random.sample(G, k=random.randint(1, 3))
```

```
    for g in assigned_groups:
        group_classes[g].append(c)

# %%
# Вспомогательные переменные для |b[t] - b[t+1]|
y = LpVariable.dicts("y", [
    (g, d, t)
    for g in G
    for d in range(1, days + 1)
    for t in range(1, slot_per_day[d - 1]) # -1 because we have n - 1 (x_1
- x_2) sums
], lowBound=0, cat=LpInteger)

# Бинарные переменные для условий
d1 = LpVariable.dicts("d1", [
    (g, d, t)
    for g in G
    for d in range(1, days + 1)
    for t in range(1, slot_per_day[d - 1])
], cat=LpBinary)

d2 = LpVariable.dicts("d2", [
    (g, d, t)
    for g in G
    for d in range(1, days + 1)
    for t in range(1, slot_per_day[d - 1])
], cat=LpBinary)

# %%
for g in G:
    for d in range(1, days + 1):
        for i in range(1, slot_per_day[d - 1]):
            t_current = get_slot_id(d, i)
            t_next = get_slot_id(d, i)

            # b[t_current] = sum(z[p, t_current, c] для всех c
            C_g = group_classes[g]

            b_current = lpSum(a[(c, t_current, r)] for c in C_g for r in
R)
            b_next = lpSum(a[c, t_next, r] for c in C_g for r in R)

            # Ограничения для y = |b_current - b_next|
            model += y[g, d, i] >= b_current - b_next
```

```
            model += y[g, d, i] >= b_next - b_current
            model += y[g, d, i] <= (b_current - b_next) + 2 * d2[g, d, i]
            model += y[g, d, i] <= (b_next - b_current) + 2 * d1[g, d, i]
            model += d1[g, d, i] + d2[g, d, i] == 1

# %%
penalty_blanks_groups = lpSum(
    y[g, d, t]
    for g in G
    for d in range(1, days + 1)
    for t in range(1, slot_per_day[d - 1])
)


# %% [markdown]
# ### Штрафы за количество пар в день для студентов

# %%
for g in G:
    for d in range(1, days + 1):
        C_g = group_classes[g]
        num_classes = lpSum(a[(c, t, r)] for c in C_g for r in R for t in
range(1, slot_per_day[d - 1])) - 4
        penalty_blanks_groups += - num_classes

# %% [markdown]
# ### Штрафы за занятия в субботу

# %%
for g in G:
    C_g = group_classes[g]
    num_classes = lpSum(a[(c, t, r)] for c in C_g for r in R for t in
range(1, slot_per_day[-1]))
    penalty_blanks_groups += - num_classes

# %%
model += penalty_blanks_groups

# %%
    model.writeMPS(mps_file)
```