Федеральное государственное бюджетное учреждение науки
Институт системного программирования им. В.П. Иванникова
Российской академии наук

На правах рукописи

Аничкин Антон Сергеевич

ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ СРЕДА ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЙ ТЕОРИИ РАСПИСАНИЙ

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
профессор, д. ф.-м. н.
Семенов Виталий Адольфович

Москва – 2018

Оглавление

Введение

Актуальность работы

Теория расписаний находит широкое применение в научных и индустриальных областях, связанных с управлением вычислительными ресурсами, планированием проектной деятельности, управлением производством, организацией транспортных потоков, диспетчеризацией воздушного сообщения. Однако многообразие существующих математических моделей и вычислительных методов, а также перманентное появление новых порождает острую проблему разработки и развития программных приложений теории расписаний на единой методологической и инструментальной основе.

Использование универсальных математических библиотек (например, MathCAD, MATLAB), библиотек, специализирующихся на поиске расписаний (например, PSPLIB, LibRCPS, MPSPLib), или систем проектного планирования общего назначения (например, Oracle Primavera, MS Project, Synchro, Spider Project, Gemini, Merlin, Zoho Projects, ManagePro, Smartsheet, GanttPro, Asana, Acunote, Teamweek, Bitrix24, Jira, Isetia) для подобных целей обычно оказывается невозможным из-за особенностей прикладных задач или крайне неэффективным в силу зависимости вычислительной сложности от частных условий. В самом деле, классические постановки «открытой линии», «рабочего цеха» или «потоковой линии», имеющие полиномиальную сложность при небольшом числе машин, простых моделях обслуживания и отсутствии директивных сроков, не следует решать как общие задачи проектного планирования (Resource-Constrained Project Scheduling Problem; RCPSP), являющиеся NP-полными.

С другой стороны, принятая практика разработки специализированных приложений планирования (например, Planets, Atlas, Moses, Cosytec, Forwards TAP-AI, Optiservice, Mosar, Cobra, SAS-Pilot, STP, Popular) является

неприемлемой в силу чрезмерных затрат на их разработку и сопровождение, обусловленных вариативностью математических моделей и сложностью реализации вычислительных методов. Адаптация унаследованных открытых кодов, изначально не предназначенных для подобных целей и не предусматривающих возможности для их развития и многократного использования, также малоэффективна даже для разработки программ близкой функциональности.

В связи с этим актуальным представляется создание единой инструментальной среды для программной реализации приложений теории расписаний. Подобная среда должна предоставлять развитые средства для разработки новых программ на основе ранее реализованных компонентов. При этом возможности развития, адаптации и конфигурирования компонентов должны обеспечивать построение эффективных программ составления расписаний, релевантных условиям и вычислительной сложности решаемых прикладных задач.

Цель и задачи работы

Главной целью диссертационной работы является разработка и апробация инструментальной среды для программной реализации моделей, методов и приложений теории расписаний. Организация инструментальной среды в виде объектно-ориентированного каркаса позволит существенно сократить затраты на разработку приложений теории расписаний, а также обеспечит надлежащую степень их надежности и эффективности при решении актуальных индустриальных задач высокой размерности.

Применение каркасного подхода обусловлено его широким распространением в практике построения многократно используемого программного обеспечения благодаря возможностям достижения компромисса между унификацией типовых компонентов и гибкостью, необходимой для построения специализированных приложений. Каркасный подход естественным образом воплощается в рамках объектно-ориентированной парадигмы

программирования в результате построения приложений в виде систем классов вместе с реализованными механизмами взаимодействия и предусмотренными возможностями их дальнейшего развития, адаптации и конфигурирования.

Для достижения декларируемой цели были поставлены следующие научные и практические задачи:

- выделить широкий класс задач теории расписаний, допускающий обобщённую программную реализацию средств решения в составе инструментальной среды. Математически обосновать использование данного класса задач путём формулировки и доказательства условий существования решений, возможности их поиска с помощью точных и приближённых алгоритмов;
- провести объектно-ориентированный анализ предметных областей, связанных с теорией расписаний и проектным планированием, и на его основе спроектировать и реализовать на языке Си++ инструментальную среду. К среде предъявляются требования универсальности (наличие готовых к использованию программных компонентов для решения типовых задач теории расписаний), эффективности (обеспечение высокой производительности при решении индустриально значимых задач высокой размерности) и гибкости (возможность многократного использования программных компонентов при создании новых приложений с относительно низкими затратами на доработку);
- провести экспериментальное исследование инструментальной среды при создании, сопровождении и развитии системы визуального моделирования и планирования проектов, а также выработать общие методологические рекомендации для построения на ее основе целевых приложений.

Научная новизна

Научная новизна диссертационной работы заключается в получении следующих оригинальных результатов:

- предложен и математически формализован класс задач обобщённого проектного планирования (Generally Constrained Project Scheduling Problem; GCPSP), охватывающий разнообразные задачи теории расписаний и проектного планирования в расширенных постановках. GCPSP задача ставится как оптимизационная задача на множестве решений, локально согласованных с эквивалентной системой ограничений с приоритетами;
- доказаны конструктивные теоремы о существовании решения задач в классе GCPSP, о сводимости классических постановок теории расписаний к задачам данного класса, а также о возможности их точного и приближённого решения на основе предложенного обобщённого алгоритма;
- разработана объектно-ориентированная среда, предусматривающая развитые инструментальные возможности для программной реализации моделей, методов и приложений теории расписаний, а также предоставляющая эффективные средства решения индустриальных задач высокой размерности;
- разработан метод построения и инкрементального развития приложений теории расписаний на основе объектно-ориентированной среды.

Теоретическая и практическая значимость

Теоретическая значимость диссертационной работы заключается в определении и математической формализации класса задач обобщённого проектного планирования, в формулировке и доказательстве утверждений о существовании решения и о сводимости классических постановок к задачам данного класса. Теоретическую значимость имеет также разработанный метод построения и инкрементального развития приложений теории расписания на основе объектно-ориентированной среды.

Практическая значимость полученных результатов заключается в возможности применения инструментальной среды для разработки программных приложений теории расписаний в различных научных и индустриальных областях. Открытая архитектура объектно-ориентированной среды, развитый

набор компонентов для задания условий и решения типовых задач, а также предусмотренные инструментальные возможности обеспечивают построение целевых приложений при относительно низких затратах.

Объектно-ориентированная среда успешно прошла экспериментальное исследование в ходе создания, многолетнего сопровождения и развития системы визуального планирования и моделирования индустриальных проектов. Данная коммерческая система внедрена более чем в трёх сотнях ведущих индустриальных компаний в тридцати шести странах мира, в том числе, и в Российской Федерации. Использование среды позволило относительно просто реализовать типовые средства проектного планирования, а в дальнейшем — развить их с учетом временных, пространственных, ресурсных, финансовых и логистических факторов. Как результат, существенно расширились функциональные возможности системы для достоверного планирования технологически сложных проектов и масштабных программ.

Методология и методы исследования

Результаты диссертационной работы получены на основе математических моделей и методов теории расписаний. При разработке инструментальной среды применялась методология объектно-ориентированного программирования.

Положения, выносимые на защиту

- Задачи обобщённого проектного планирования GCPSP и методы их решения.
- Объектно-ориентированная среда для построения и развития приложений теории расписаний.
- Метод инкрементальной разработки приложений на основе предложенной объектно-ориентированной среды.

Апробация работы

Основные положения и результаты настоящей диссертационной работы обсуждались и докладывались на следующих российских и международных научных конференциях:

- конференция, посвященная 80-летию со дня рождения академика В.А. Мельникова, Вычислительный центр им. А.А. Дородницына Российской академии наук, Москва, Россия, 2009 г.;
- XXXVI международная конференция «Информационные технологии в науке, образовании, телекоммуникации и бизнесе» IT + SE`09, весенняя сессия, Ялта-Гурзуф, Крым, Украина, 20 – 30 мая 2009 г.;
- международная конференция «20th ISPE International Conference on Concurrent Engineering», Мельбурн, Австралия, 2013 г.;
- международная конференция «13th International Conference on Construction Applications of Virtual Reality (CONVR)», Лондон, Великобритания, 2013 г.;
- XLIV международная конференция «Информационные технологии в науке, образовании, телекоммуникации и бизнесе» IT + S&E`15, весенняя сессия, Ялта-Гурзуф, Крым, Россия, 22 мая – 01 июня 2015 г.;
- XX Байкальская Всероссийская конференция с международным участием, школа-семинар научной молодёжи, «Информационный и математические технологии в науке и управлении», Байкальская сессия, 1 – 7 июля 2015 г.;
- XLVI международная конференция «Информационные технологии в науке, образовании, телекоммуникации и бизнесе» IT + S&E`17, весенняя сессия, Ялта-Гурзуф, Крым, Россия, 22 мая – 01 июня 2017 г.;
- семинар «Объектно-ориентированная среда для разработки приложений теории расписаний», Институт прикладной математики им. М.В. Келдыша Российской академии наук, Москва, Россия, 7 декабря 2017 г.

Публикации

По теме диссертационной работы опубликовано 13 работ, в том числе 6 статей [1, 2, 3, 4, 5, 6] в реферируемых научных журналах из списка изданий, рекомендованных ВАК РФ. Работа [1] опубликована в книге, индексируемой Web of Science.

В работах [3, 4, 5, 7, 8] все научные результаты принадлежат автору. Научным руководителем Семеновым В.А. осуществлялась постановка и

формализация задач, а также проводились редакторские правки. В работе [1] автором была математически формализована классическая постановка задачи проектного планирования. Персональный вклад автора в работе [2] заключается в математической формализации задач проектного планирования в расширенных постановках, а также в описании схемы поиска расписания. В работе [6] вклад автора заключается в личном участии в разработке целевого приложения, а также в проведении сравнительного анализа.

Личный вклад автора

Все представленные в диссертационной работе результаты получены автором лично.

Объем и структура диссертации

Представленная диссертационная работа состоит из введения, четырех глав основного содержания, заключения, библиографического списка, состоящего из 207 публикаций. Общий объём работы составляет 168 страниц.

В главе 1 проведён обзор основных современных моделей и методов, используемых при решении задач теории расписаний и, в частности, проектного планирования. Глава 2 посвящена математической формализации задач проектного планирования и их обобщенной постановке. В главе 3 описывается разработанная объектно-ориентированная среда для разработки приложений теории расписания. Глава 4 посвящена экспериментальным исследованиям разработанной среды. В заключении перечисляются основные результаты настоящей диссертационной работы.

Глава 1. Современные модели и методы теории расписаний

В настоящей главе проводится систематизация моделей и методов теории расписания с целью проведения их объектного анализа и последующего построения универсального каркаса для реализации программных приложений. Глава представлена в виде обзора работ в области теории расписаний. При этом главное внимание уделяется задачам и методам проектного планирования (Resource Constrained Project Scheduling Problem или, сокращенно, RCPSP), которые, с одной стороны, находят широкое практическое применение, а с другой стороны, — обобщают математические постановки, возникающие в смежных предметных областях и дисциплинах.

Задачи теории расписаний обычно формулируются как задачи оптимизации обслуживания конечного множества требований в системе, содержащей ограниченное число машин. Для каждого требования указывается время обработки на каждой машине, порядок обслуживания и сроки выполнения. Традиционно задачи теории расписания делят на четыре основных класса:

- постановка «открытая линия» (open shop) предполагает многостадийное выполнение каждого требования на заданном подмножестве машин в произвольном порядке;
- постановка «рабочий цех» (job shop) устанавливает для каждого требования строгий порядок выполнения на заданном подмножестве машин;
- постановка «потоковая линия» (flow shop) фиксирует порядок использования машин и предполагает последовательное многостадийное выполнение каждого требования на каждой машине в установленном

порядке; решением задачи является последовательность требований, при которой минимизируется общее время обслуживания;

- постановка с директивными сроками (release dates) предполагает задание для каждого требования времени обслуживания, а также директивных сроков его поступления и окончания. В отличие от многостадийных постановок, требования в данном классе задач могут выполняться на одной машине. Поэтому обычно допускаются прерывания и произвольный порядок обслуживания требований. При наличии нескольких расписаний, удовлетворяющих предписанным директивным срокам, в качестве решения выбирается расписание с минимальным общим количеством прерываний.

Большинство практически содержательных задач составления расписаний допускают произвольные комбинации ограничений и дисциплин обслуживания и являются NP-полными задачами. Лишь немногие постановки с частными условиями могут быть решены за полиномиальное время. Например, расписание с прерываниями, удовлетворяющее директивным срокам при произвольном числе машин, может быть построено за $O(n^3)$ [9]. Для задач в постановке «потоковая линия» расписание для двух машин с минимальным общим временем обслуживания может быть составлено за $O(n \log(n))$ [9]. Тем самым, вычислительная сложность задач может существенно варьироваться в зависимости от конкретных условий. Развёрнутые обзоры задач теории расписаний с анализом их вычислительной сложности можно найти в ряде источников [10, 11].

Примечательно, что постановки «открытая линия», «рабочий цех» и «потоковая линия» являются частными случаями задач ресурсного планирования проектов [12]. Более того, известные задачи об упаковке в контейнеры (линейная, двумерная упаковка) [13], разнообразные постановки «О рюкзаке» (упаковки по стоимости и весу) [14], классическая задача «О коммивояжере», а также задачи составления расписаний в учебных заведениях могут формулироваться и решаться как задачи ресурсного планирования RCPSP [15, 16, 17, 18]. Тем самым,

обсуждаемый класс задач приобретает особое значение в контексте проводимой систематизации и концептуализации теории расписаний, а также в связи с построением универсального объектно-ориентированного каркаса для разработки программных приложений.

Поскольку в теории расписаний нет единой принятой терминологии, в дальнейшем будут употребляться понятия ресурсного планирования. Вместо «требование», «активность», «процесс» или «операция» будет употребляться термин «работа». Вместо «машина» и/или «станок» будет использоваться понятие «обобщённый ресурс», который охватывает как возобновимые ресурсы (например, комплект оборудования или штат сотрудников), так и невозобновимые ресурсы (связанные, например, с материальными и финансовыми затратами). Классическая RCPSP-задача ставится как задача минимизации общего времени выполнения всего проекта при соблюдении временных отношений между работами и не нарушении условий доступности ресурсов. Математическая формализация классической задачи была проведена Притскером [19], а её NP-полнота доказана Блазевицем [20]. Если в прикладной задаче учитывается большее количество ограничений, используются более сложные модели работ или ресурсов или иные целевые функции, то такая задача относится к классу задач проектного планирования в расширенных постановках.

Для решения классической RCPSP-задачи были разработаны точные и приближенные методы [21, 22, 23]. Первую группу составляют метод прямого перебора, метод «ветвей и границ» [24], методы линейного программирования [25], метод динамического программирования [26] и метод декомпозиции [27]. Методы обеспечивают поиск оптимального расписания, но в силу высокой вычислительной сложности применимы лишь к небольшим проектам. Приближенные методы, такие как метод Монте-Карло [28], метод частичного перебора [29], метод направленного перебора [29], упрощенный метод «ветвей и границ» [24], а также современные методы последовательного и параллельного составления расписаний на основе эвристических правил [30] позволяют

генерировать эффективные расписания для масштабных проектов за разумное время [23, 31].

С релаксацией ресурсных ограничений вычислительная сложность RCPSP-задачи может быть существенно понижена. Так, при отсутствии ресурсных ограничений задача вовсе сводится к задаче поиска наидлиннейшего пути в плане, которая решается методом критических путей (Critical Path Method или, сокращенно, CPM) за линейное время $O(n)$. Кроме упомянутой классической задачи опубликовано значительное число работ, посвященных частным и обобщённым постановкам ресурсного планирования. Главным образом они отличаются целевыми функциями, способами исполнения работ, типами временных или иных ограничений, а также моделями ресурсов. Немногие из этих задач получили практическое распространение и лишь несколько методов составления расписаний для RCPSP-задач реализованы в составе современных программных систем управления проектами. Причина, видимо, заключается в сложности поддержки многовариантных постановок и трудности обобщённой программной реализации методов планирования, рассчитанной на широкие классы задач.

В разделе 1. 1 настоящей главы будет рассмотрена классическая постановка RCPSP-задач, а также уточнена роль нотации Грэхэма [32] и правил Брюкера [33] в систематизации и классификации задач теории расписания. Раздел 1. 2 посвящён анализу моделей ресурсов, применяемых в задачах планирования проектов. Выделение классов возобновимых, невозобновимых, ограничено-возобновимых, частично возобновимых, эксклюзивных, логистических, непрерывно разделяемых ресурсов и ресурсов с переменной доступностью позволяет охватить наиболее содержательные случаи. В разделе 1. 3 детально обсуждаются особенности моделей исполнения работ. Особое внимание уделяется работам с прерываниями, альтернативным режимам исполнения работ, профилям использования ресурсов, учёту накладных расходов, а также обеспечению компромиссов. Важные факторы календарно-сетевого планирования, включая основные виды временных ограничений,

рассматриваются в разделе 1. 4. Проводимый анализ охватывает отношения предшествования с минимальными и максимальными лагами, явные временные ограничения, ограничения рабочего времени и логические зависимости между работами. Наконец, в заключительном 1. 5 разделе обсуждается выбор целевой функции, необходимой для корректной постановки соответствующей оптимизационной задачи. Минимизация временных показателей проекта, устойчивость к задержкам, обеспечение консервативности расписания, минимизация затрат на возобновимые и невозобновимые ресурсы, минимизация общей стоимости проекта, максимизация чистой приведенной стоимости могут применяться в качестве критериев поиска оптимального расписания.

1. 1. Классическая постановка RCPSP-задачи

Классическая постановка RCPSP-задачи составления расписания формализуется следующим образом. Исходными данными является проект, состоящий из J работ. Каждая работа имеет свой номер (индекс) $j = 1,2,\dots,J$. Кроме того каждая работа j характеризуется временем своего выполнения (продолжительностью) p_j, которое может быть нулевым ($p_j \geq 0$). На работу могут накладываться технологические ограничения, связанные с невозможностью начать её выполнение ранее, чем завершатся одна или более связанные с ней предыдущие работы, именуемые предшественниками. Данная работа по отношению к своим предшественникам является последователем. P_j является множеством, содержащем всех предшественников работы j. Взаимозависимости между работами не должны иметь циклический характер.

Кроме работ в проекте может присутствовать K возобновимых ресурсов. Максимально доступное количество каждого ресурса $k = 1,2,\dots,K$ ограничено константой R_k. Каждая работа j может требовать для своего выполнения некоторое количество r_{jk} ресурса k. Причём r_{jk} не может превышать предел доступности R_k ресурса k. Потребляемых работой ресурсов может быть несколько. Потребление ресурса работой означает, что с началом выполнения

работы j ресурс k в количестве r_{jk} считается занятым, то есть не доступным для выполнения других работ, а с окончанием выполнения работы данное количество ресурса высвобождается. Таким образом, потребление ресурса носит равномерный характер в течение всего времени выполнения работы. Работы с нулевой продолжительностью не требуют для своего выполнения ресурсов, так как время захвата ресурсов совпадает со временем их высвобождения.

Работа не может быть прервана, то есть если работа была начата, то она не может приостановиться и временно высвободить все используемые ею ресурсы. Для упрощения вычисления временных рамок выполнения всего проекта вводится две дополнительные фиктивные работы с нулевой продолжительностью ($j = 0$ и $j = J + 1$), соответствующие началу и завершению всего проекта. Первая работа ($j = 0$) становиться предшественником для всех работ, ранее не имевших предшественников. Последняя ($j = J + 1$) — последователем для всех работ, ранее не имевших последователей.

Все данные считаются известными и детерминированными. Все параметры и константы являются целочисленными и неотрицательными. Неизвестными считаются время S_j и время C_j начала и завершения работы $j = 0,1, \dots, J, J + 1$ соответственно, причём $C_j = S_j + p_j$.

Классическая RCPSP задача построения расписания сводиться к поиску (вычислению) всех значений S_j и/или C_j таких, чтобы время выполнение всего проекта $C_{MAX} = C_{J+1}$ было минимальным из всех возможных. В терминах общепринятой нотации Грэхема [32] данная задача обозначается как $PS|prec|C_{MAX}$.

Нотация Грэхема для обозначения классов задач теории расписания представляет собой комбинацию трёх характеристик $\alpha|\beta|\gamma$. Первая характеристика α может задаваться только одним значением, однозначно определяющим модель ресурсов. Вторая характеристика β описывает используемую модель исполнения работ. Данная характеристика может быть представлена одним или несколькими значениями или отсутствовать вовсе.

Третья характеристика γ определяет целевую функцию, минимизация которой и является задачей составления оптимального расписания. Сама целевая функция может быть как простой, так и составной. Задание данной характеристики является обязательным, поскольку именно она определяет стратегию поиска решения и качество полученного результата.

1. 2. Модели ресурсов

1. 2. 1. Невозобновимые и ограничено-возобновимые ресурсы

В классической постановке RCPSP-задачи рассматриваются только, так называемые, возобновимые ресурсы, которые доступны в любой момент времени в фиксированном количестве. Однако часто рассматривают три вида ресурсов: возобновимые (renewable), невозобновимые (nonrenewable) и ограничено-возобновимые (doubly constrained). Такая классификация впервые была предложена в [34, 35].

Доступность возобновимых ресурсов, таких как рабочие или машины, определяется в каждый момент времени. Ограничения, связанные с невозобновимыми ресурсами, например, с бюджетным планом, распространяются на весь проект. Классическая RCPSP-задача предусматривает задание только возобновимых ресурсов. Невозобновимые ресурсы могут учитываться, например, в рамках мультимодальной постановки (см. подраздел 1. 3. 4). В классической постановке сумма потраченных невозобновимых ресурсов всегда будет одинаковой вне зависимости от построенного расписания.

Как правило, при планировании учитываются как возобновимые, так и невозобновимые ресурсы. В терминах $\alpha|\beta|\gamma$ нотации данная ресурсная модель обозначается как $\alpha = MPS; R; N$. Это обозначение несколько отличается от оригинального, приведённого Брюкером в публикации [33]. Однако данное обозначение удачно отражает комбинированный характер ресурсной модели.

Ограничено-возобновимые ресурсы сочетают в себе особенности как возобновимых, так и невозобновимых ресурсов. Ограничения, заданные для них,

содержательны как для определенных моментов времени, так и на протяжении всего периода выполнения. К ресурсам данного вида следует отнести рабочего, который не может быть задействован в более чем в пяти работах за весь период выполнения проекта. Однако, такая двойственность позволяет интерпретировать ограничено-возобновимые ресурсы как простую комбинацию возобновимых и невозобновимых ресурсов, избегая выделения самостоятельного вида.

1. 2. 2. Частично возобновимые ресурсы

Понятие частично возобновимого ресурса (partially renewable resource) впервые было введено в [36]. Суть данной ресурсной модели заключается в следующем. Для каждого частично возобновимого ресурса k имеется множество Π_k, представленное временными подмножествами Q_{ki}. Считается, что ресурс k доступен в количестве $R_k^{\Pi}(Q_{ki})$ в течение времени, определённого подмножеством $Q_{ki} \subseteq \{1, \dots, T\}$, где $Q_{ki} \in \Pi_k$ и T — условное время завершения проекта. Потребление ресурса k работой j по-прежнему представляет собой константное значение r_{jk}. В более общем случае потребление частично возобновимого ресурса определяется для каждого подмножества Q_{ki}. Другими словами, данный вид ресурсов позволяет определить несколько альтернативный сценариев их потребления и доступности. Примером является обычный рабочий, который может работать либо каждый день с понедельника по пятницу, либо только в выходные, но не все семь дней. Π_k в этом случае будет состоять из двух подмножеств: с понедельника до пятницы (Q_{k1}) и с субботы до воскресения (Q_{k2}). При этом $R_k^{\Pi}(Q_{k1}) = R_k^{\Pi}(Q_{k2}) = 1$.

Авторы публикации [37] рассмотрели данную ресурсную модель в рамках мультимодальной постановки RCPSP-задачи (см. подраздел 1. 3. 4). В публикации [38] авторами был рассмотрен частный случай данной ресурсной модели, когда подмножества Q_{ki} представляют собой временные интервалы $[t_1, t_2]$. Такой подход представляется более естественным, поскольку ресурсы обычно доступны на некотором промежутке времени, пусть даже и очень коротком.

Для обозначения ресурсного планирования с частично возобновимыми ресурсами обычно использую $\alpha = PS; PR$.

1. 2. 3. Логистические ресурсы

Понятие логистического (или кумулятивного) ресурса (cumulative resource) было введено в [39], где оно рассматривалось в сочетании с минимальными и максимальными временными лагами (см. подразделы 1. 4. 1 и 1. 4. 2) в рамках RCPSP-задачи. Данный вид ресурсов позволяет учитывать процессы производства, связанные с временным размещением промежуточных результатов деятельности на некоторой площади, в резервуаре или таре, которые в этом случае следует рассматривать в качестве самостоятельных логистических ресурсов. Следует отметить, что логистические ресурсы могут определяться как в контексте возобновимых, так и невозобновимых ресурсов.

Логистический ресурс характеризуется своей номинальной ёмкостью R_k и некоторым её запасом $\overline{R_k}$. Каждая работа j может производить или потреблять (перерабатывать) продукцию в объёме r_{jk} ($r_{jk} > 0$ в случае производства и $r_{jk} < 0$ в случае потребления). Использование подобной модели обозначается как $\alpha = PS; Cu$.

В публикации [40] логистические ресурсы рассматривались в рамках мультимодальной постановки RCPSP-задачи (см. раздел 1. 3. 4) применительно к процессам разработки и тестирования в автомобильной промышленности. Тестовый автомобиль моделировался в качестве логистического ресурса, который сначала производился (занимал некоторый объём), потом использовался (занимаемый объём не изменялся), а затем уничтожался в краш-тесте (объём освобождался). В публикациях [41, 42] данные ресурсы применялись для планирования серийного производства в обрабатывающей промышленности.

1. 2. 4. Непрерывно разделяемые ресурсы

В классической постановке доступность ресурсов рассчитывается дискретным образом. В [43] авторами рассматривается возможность их

произвольного непрерывного исчисления и разделения (continuous resource). Это вполне естественно для проектов с ресурсами, представляющими собой, например, энергетические источники или какие-либо жидкие расходные материалы. Примечательно, что данная возможность применима как к возобновимым, так и невозобновимым ресурсам [44]. В нотации $\alpha|\beta|\gamma$ использование данного ресурсного концепта обозначается как $\alpha = PS; Co$.

Более сложные постановки, связанные с непрерывно разделяемыми ресурсами, рассматривались в [45, 46, 47, 48]. Следует отметить публикацию [49], в которой фактор непрерывности применялся для ограничено возобновимых ресурсов.

1. 2. 5. Эксклюзивные ресурсы

Эксклюзивными называются ресурсы (dedicated resource), которые в один момент времени могут быть использованы только в одной работе [50, 51]. Такие ресурсы могут моделироваться возобновимыми ресурсами с лимитом использования $R_k = 1$ в любой момент времени. Поэтому постановки с подобными ресурсами являются частными случаями RCPSP-задачи. Авторы работ [52, 53] использовали дизъюнктивную функцию планирования для интерпретации RCPSP-задач с эксклюзивными ресурсами. В $\alpha|\beta|\gamma$ нотации данный вид ресурсов представляется как $\alpha = PS; Rm, 1, 1$, что говорит о наличие в проекте m возобновимых ресурсов с доступным количеством 1 и использованием в работах в количестве 1.

1. 2. 6. Ресурсы с переменной доступностью

В классической постановке уровень доступности возобновимых ресурсов фиксирован на протяжении всего проекта. Подобное допущение не всегда выполняется на практике. Например, в реальных проектах количество доступных специалистов может варьироваться в связи с периодическими отпусками, занятостью в других работах и т.п.

Чтобы учитывать данный фактор, вводится понятие доступности возобновимого ресурса k в момент времени t как функции времени $R_k(t)$. Ресурсы с переменной доступностью рассматривались в работах [41, 54, 55, 56, 57, 58, 59]. Часто данная модель применяется в задачах планирования работ с прерываниями (см. подраздел 1. 3. 1). В [60] обсуждаются аспекты применения данной модели в медицинских исследовательских проектах для контроля доступности исследовательского персонала и лабораторного оборудования.

В нотации $\alpha|\beta|\gamma$ переменная доступность ресурсов отражается как $\alpha = PS; Rm; *,*$, что означает наличие в проекте m возобновимых ресурсов с переменной доступностью и произвольными уровнями потребления в работах.

Авторы публикации [17] ввели понятие дизъюнктивных ресурсов с доступностью, изменяемой во времени от 0 до 1. Заметим, что этот случай соответствует ограниченной функции $R_t(t) \in \{0, 1\}$ и является вариацией эксклюзивных ресурсов. В работе [61] рассматривается задача, в которой доступность ресурсов изменяется только в определённые моменты времени, связанные с вехами проекта (milestone).

В работе [62] показано, что задача с ресурсами переменной доступности может быть редуцирована к задаче с ресурсами фиксированной доступности путем введения минимальных и максимальных лагов. Для этого предлагается заменить переменную доступность $R_k(t)$ ресурса k на постоянную, соответствующую максимальному значению доступности ресурса за всё время выполнения проекта ($R_k = \max_{0 \le t \le T} R_k(t)$), а в периоды, когда доступность меньше максимальной, ввести фиктивные работы, компенсирующие избыток ресурса ($r_{jkt} = R_k - R_k(t)$). Местоположение таких фиктивных работ в расписании предполагается зафиксировать с помощью минимальных и максимальных временных лагов.

Следует отметить, что задачи с ресурсами переменной доступности являются частными случаями RCPSP-задач с частично возобновимыми ресурсами.

1. 3. Модели исполнения работ

1. 3. 1. Работы с прерываниями

Классическая постановка не допускает прерываний уже начатых работ. Однако рядом исследователей рассматривались работы с прерываниями (preemptive scheduling), которые могут происходить в дискретные моменты времени, обычно, кратные точности представления временных интервалов [17, 38, 63, 64]. В нотации Брюкера такие задачи обозначаются как $prmt$ в поле β [33]. Изучалась также модель выполнения, допускающая фиксированное число прерываний, не превышающее для каждой индивидуальной работы некоторый заданный порог [65]. В [66, 67] предлагается определить первоначальный интервал, на протяжении которого работы не могут прерываться. Менее содержательной, на наш взгляд, выглядит модель, описанная в работе [63] и устанавливающая возможность параллельного исполнения отдельных стадий работы, сформированных в результате прерываний (fast tracking). Довольно часто прерывания работ интерпретируются в терминах календарей, которые устанавливают допустимые интервалы для проведения работ [38, 68]. Однако календари обычно учитываются при вычислении дат начала и завершения работ с учетом выходных и рабочих дней. Поэтому их использование для моделирования прерываний кажется искусственным.

1. 3. 2. Профильное использование ресурсов

В традиционной постановке предполагается, что каждая работа j может требовать для своего выполнения некоторое фиксированное количество r_{jk} ресурса k, которое не может изменяться в ходе выполнения. Данная ресурсная модель естественным образом обобщается путём введения профиля r_{jkt} для каждого t-ого интервала выполнения работы j при потреблении ресурса k. Так называемые, зависимые от времени ресурсы (time-dependent resources) находят применение во многих приложениях [60, 69, 70]. Тем не менее, существует

довольно элегантный прием смоделировать эту ситуацию путем представления работы эквивалентной цепочкой подзадач с фиксированными уровнями потребления ресурса [62]. Чтобы избежать наложения или прерывания подзадач по времени, для каждой пары соседей i и $i+1$ в цепочке устанавливаются минимальные и максимальные лаги с нулевой задержкой (см. подразделы 1. 4. 1 и 1. 4. 2).

1. 3. 3. Учет накладных временных затрат

Другим направлением в развитии модели исполнения работ является учёт времени подготовки используемых работой ресурсов (setup times), например, для переналадки оборудования. В [71] рассматривается три альтернативы, выражающие характер зависимости времени подготовки от последовательности работ или всего плана работ, в которых задействован ресурс. Соответственно, различают времена подготовки, независимые от последовательности (sequence-independent setup times), зависимые от последовательности (sequence-dependent setup times) и зависимые от расписания (schedule-dependent setup times). Первые учитывают зависимость времени подготовки ресурса от него самого и работы, в котором он участвует. Вторые принимают во внимание и предшествующую работу, использующую тот же ресурс. Наконец, третьи учитывают характер использования ресурсов всеми предшественниками. В нотации Брюкера [33] независимые от последовательности и зависимые от последовательности времена обозначаются в поле β как s_j и s_{ij} соответственно.

В [72] обсуждаются иные временные факторы, в частности, рассматриваются факторы перенастройки и/или отключения/консервации/вывода из эксплуатации ресурса (removal times). Хотя временные факторы подготовки ресурсов обычно рассматриваются как самостоятельные концептуальные элементы, часто для их представления и учета используется мультимодальность исполнения работ (см. подраздел 1. 3. 4) [41, 73] или механизмы прерываний и параллельного исполнения работ в сочетании с мультимодальностью [74].

1. 3. 4. Альтернативные режимы исполнения работ

В традиционной постановке каждая работа выполняется единственным способом, однозначно определяемым заданной продолжительностью и требуемым количеством ресурсов. Начиная с пионерской работы Эльмагхраби [75], концепция работ была существенно пересмотрена в сторону поддержки альтернативных (или мультимодальных) режимов M_j для каждой индивидуальной работы j. Каждый режим m отражает один из реализуемых способов исполнить работу за фиксированный период p_{jm} при фиксированном количестве r_{jmk} ресурса k. Изменения режима в ходе выполнения работы и во время её прерывания не допускаются. Класс задач планирования, допускающих мультимодальное исполнение работ, называют Multimodal Resource Constrained Project Scheduling Problem или, сокращенно, MRCPSP. Решением задачи MRCPSP является расписание, которое включает в себя не только даты начала работ S_j, но и режимы m_j, в котором следует выполнять работы. В нотации Брюкера данный класс задач обозначается как MPS в соответствующем поле α.

Задачи MRCPSP с возобновимыми ресурсами активно изучались в последнее время [57, 76, 77, 78, 79, 80, 81]. Постановки MRCPSP с обобщенными отношениями предшествования (лагами) рассмотрены в публикациях [17, 56, 73, 82, 83, 84, 85], а постановки с обобщенными ресурсными ограничениями — в [37]. В публикациях [86, 87, 88] вводится и обосновывается понятие качества мультимодального расписания, как некоторой интегральной характеристики, отражающей важность и приоритетность использования одних режимов над другими. В публикациях [73, 89] рассматриваются группы работ, на которые накладываются условия их согласованного выполнения в одних и тех же режимах (mode identity constraints).

1. 3. 5. Учет компромиссов

В RCPSP-задачах выделяют два рода компромиссов: компромисс между временем исполнения всего проекта и уровнями потребления ресурсов, а также компромисс между временем исполнения проекта и его стоимостью.

В задаче поиска компромиссного расписания первого рода вводится понятие нагрузки φ_j для работы j. Работа может быть выполнена при любой дискретной комбинации времени выполнения p_j и потребления ресурса r_{jk}, такой что $p_j r_{jk} \geq \varphi_j$. При этом содержательными являются комбинации минимальных допустимых пар значений, для которых имеет место $(p_j - 1)\, r_{jk} < \varphi_j$ и $p_j(r_{jk} - 1) < \varphi_j$. Задача обеспечения компромисса первого рода для возобновимых ресурсов исследовалась в публикациях [90, 91, 92].

В задаче поиска компромиссного расписания второго рода во внимание принимается единственный невозобновимый ресурс, который играет роль бюджета проекта. При этом ставится задача минимизации времени выполнения проекта при фиксированном уровне бюджета, а также двойственная ей задача минимизации бюджета проекта при заданном предельном сроке его окончания (deadline) [93, 94].

Примечательно, что обе постановки могут рассматриваться как частные случаи задачи MRCPSP с альтернативным исполнением работ. Единственное отличие состоит в том, что в задачах поиска компромиссных расписаний вместо допустимых комбинаций значений времени выполнения работы и потребляемыми ею ресурсами явно оперируют с нагрузками. Иногда, чтобы подчеркнуть этот нюанс, для описания этого класса задач уточняют нотацию $\alpha = T_{tr}PS$. В постановке $\alpha = T_{tc}PS$ при заданном крайнем сроке окончания проекта минимизируется бюджет, причем функция стоимости имеет квадратичную зависимость от задержки фактических сроков выполнения работ [95].

1. 4. Временные ограничения

1. 4. 1. Предшествования с минимальными лагами

В классической постановке работа не может стартовать пока все предшественники не завершат выполнение. Понятие простого предшествования обычно расширяется введением минимального временного лага d_{ij}^{FS}, определяющее ограничение на возможный старт последователя j относительно времени завершения предшественника i как $C_i + d_{ij}^{FS} \leq S_j$. Отрицательные значения лагов допускаются, означая, что работы могут выполняться с наложением временных интервалов. Часто в приложениях рассматривается сразу четыре вида временных лагов d_{ij}^{FS}, d_{ij}^{SS}, d_{ij}^{SF}, d_{ij}^{FF}, связывающие времена старта или завершения предшественника со временем старта или завершения последователя. При условии фиксированной продолжительности планируемых работ данные ограничения легко трансформируются друг в друга [62], однако при наличии альтернативных сценариев выполнения работ и актуализации проектного плана поддержка ограничений всех видов является необходимой.

Для данных видов ограничений используется значение $temp$ для поля β в нотации Брюкера [33]. Ограничения данного вида рассматриваются в многочисленных публикациях [55, 58, 59, 96, 97, 98, 99], однако в публикации [100] было отмечено, что данные ограничения могут использоваться для моделирования процессов настройки ресурсов и загрузки производственных линий (sequence-independent setup times).

1. 4. 2. Предшествования с максимальными лагами

Аналогично отношениям предшествования с минимальным лагом вводятся временные ограничения с максимальным лагом. Для предшественника i и последователя j данный вид отношений выражается следующим образом: $C_i + d'^{FS}_{ij} \geq S_j$. Задание ограничения данного вида означает, что последующая работа должна стартовать не позже, чем через d'^{FS}_{ij} единиц времени после

завершения предшествующей работы. Заметим, что одновременное задание нулевого минимального лага и нулевого максимального лага означает, что последователь должен стартовать сразу после завершения предшественника. Комбинация времен начала и завершения работ для отношения предшествования приводит к четырем видам максимальных лагов d'^{FS}_{ij}, d'^{SS}_{ij}, d'^{SF}_{ij}, d'^{FF}_{ij}, аналогичных ранее рассмотренных. Класс задач планирования, в которых допускается одновременное использование минимальных и максимальных лагов часто обозначается как $RCPSP/max$. Он был рассмотрен в ряде публикаций [41, 62, 101, 102, 103, 104, 105, 106]. В контексте анализа альтернативных сценариев выполнения работ он изучался также в публикациях [17, 82, 83, 84, 85]. А в [107] для них предложена довольно интересная модель лагов, учитывающая режимы выполнения предшественника m_i и последователя m_j. Однако, следует отметить, что использование максимальных временных лагов способно привести к циклическим структурам. Этот аспект подробно рассмотрен в публикации [108].

1. 4. 3. Явные временные ограничения

В расширенной RCPSP постановке могут применяться такие понятия, как директивный срок выполнения работы и крайний срок выполнения работы (deadline). Принципиальная разница между этими понятиями заключается в том, что значение первого носит рекомендательный характер, а второго — обязательный. В первом случае работу желательно выполнить к назначенному сроку. Причём опережение или запаздывание может «наказываться» штрафом. Во втором случае работа обязана завершиться до заданного крайнего срока. Нарушение этого условия делает дальнейший поиск расписания бессмысленным. Возможны случаи, когда данное условие не может быть соблюдено ни при каких обстоятельствах. Это лишь означает, что исходная задача планирования поставлена математически некорректно.

Существуют также понятия директивного и самого раннего срока начала выполнения работы. Первое носит рекомендательный характер, второе — строго

обязательный. Однако понятие директивного срока начала выполнения работы применяют крайне редко, так как оно однозначно связано с более важным директивным сроком завершения работы. Поэтому в публикациях чаще всего рассматриваются самый ранний срок начала и крайний срок завершения выполнения работы, а также директивный срок завершения работы. Задачам с директивными сроками завершения работ посвящены многочисленные публикации [106, 109, 110, 111, 112]. Основное внимание при этом уделяется минимизации штрафов, возникающих из-за нарушения директивных сроков выполнения работ.

Понятия самого раннего и крайнего сроков выполнения работ, а также их применения в задачах планирования подробно рассматриваются в публикациях [45, 47, 55, 69, 97, 113]. Отдельного внимания заслуживает работа [113], авторы которой рассмотрели, так называемую, кумулятивную задачу построения расписания. Это задача ресурсного планирования, в которой отсутствуют отношения предшествования между работами, однако для каждой работы определены самый ранний срок начала и крайний срок завершения. В результате все работы оказываются «запертыми» в некоторых временных рамках, а иногда и строго фиксируемыми по времени исполнения.

Следует заметить, что задание самого раннего срока начала выполнения работы эквивалентно определению минимального временного лага относительно стартовой вехи проекта, а задание крайнего срока завершения — определению максимального временного лага относительно финальной вехи проекта. Данное наблюдение позволяет представлять и разрешать временные ограничения в рамках единой дисциплины обработки отношений предшествования.

В нотации $\alpha|\beta|\gamma$ ограничения, накладываемые на начало выполнения работы j, соответствуют обозначению r_j, а накладываемые на завершение работы — d_j.

1. 4. 4. Ограничения рабочего времени

Понятие ограниченного рабочего времени было введено в [114]. Временная шкала, в соответствии с которой проводится планирование, разбивается на циклически повторяющиеся рабочие и нерабочие временные окна или интервалы. Каждая работа может начать своё выполнение только в рабочее время. Продолжительность выполнения работы не распространяется на нерабочие интервалы. Например, нормированный рабочий день предполагает наличие интервалов с 9 часов утра до 17 часов вечера каждый день, кроме субботы и воскресенья и исключая возможные обеденные перерывы.

В [17] отмечена возможность моделирования рабочих интервалов с помощью дополнительного возобновимого ресурса, лимит доступности которого равен 1 в рабочее время и 0 — в нерабочее. В работе [115] авторы применили дисциплину ограничения рабочего времени при компромиссном планировании проекта с соблюдением требований к срокам реализации проекта и его стоимости.

1. 4. 5. Иные временные ограничения

В работе [40] было введено отношение частичного порядка между работами. Данное отношение применялось в проектах автомобилестроительной отрасли, связанных с проведением инженерных и тестовых работ. Отношение частичного порядка между работами i и j подразумевает, что либо работа i должна быть завершена до начала работы j, либо данные работы должны выполняться в разных режимах (то есть использовать разные ресурсы). Подобное отношение может устанавливаться, если выполнение работы j предполагает уничтожение ресурса, который мог быть использован другой работой i. Например, работы по сборке автомобиля должны предшествовать краш-тесту, в результате которого автомобиль разрушается.

В работе [60], посвящённой медицинским исследовательским проектам, применяется отношение между работами, исключающее их завершение за один и тот же временной период. При этом не уточняется, какая работа должна быть

завершена первой. В работе [17] вводится ограничение параллельности. Данное ограничение предполагает принудительное выполнение двух и более работ параллельно на одном и том же отрезке времени. Там же рассматривается противоположный случай, когда отрезки выполнения работ не могут пересекаться. Кроме того, авторы обобщили данное условие введением временного лага с момента окончания одной работы до начала другой. Следует отметить, что данное ограничение не может моделироваться отношениями предшествования с минимальными и максимальными лагами, поскольку очерёдность выполнения работ не определёна. Авторы публикации [17] также ввели понятие «коридора параллельности» $[l, u]$, который предполагает, что ассоциируемые с ним работы должны выполняться параллельно в течении l временных единиц по крайней мере, но не более, чем в течении u временных единиц.

Представляет несомненный практический интерес также отношение предшествования с лагом, выраженном в процентном соотношении [47]. Оно предполагает установку процентного значения лага g_{ij} между работами i и j, и означает, что работа j не может начаться раньше, чем работа i завершится на g_{ij} процентов. При этом процент завершения стартовавшей работы j не может превышать процента завершения работы i. Такой вид временных ограничений важен для проектов, в которых моменты начала работ разделены по времени и не допускается опережение одних работ другими.

В работе [54] рассматривается довольно занятное ограничение, которое предполагает, что никакая работа k не может быть запланирована между заданными работами i и j. В работе [56] используется близкий вид ограничений, в котором еще учитывается зависимость от возобновимого ресурса. Работа j, связанная данным ограничением с работой i, не может начаться раньше, чем завершиться работа i, при этом ни одна другая работа k, которая использует некоторый заданный в отношении ресурс r, не может начаться после завершения работы i и до начала работы j. Данный вид ограничений может использоваться,

например, для обеспечения гарантии, что никакая посторонняя работа не сможет начаться между вспомогательной работой по установке ресурса и работой, непосредственно использующей данный ресурс.

Временные ограничения на перемещения ресурсов рассматриваются в [116]. Согласно предложенной модели перемещение ресурса r из местоположения, в котором данный ресурс использовался при выполнении работы i, в местоположение, в котором он будет использоваться работой j, занимает Δ_{ijr} временных единиц. Данное время может явно задаваться или насчитываться по местоположению проведения работ.

1. 4. 6. Логические зависимости

Авторы некоторых публикаций [117, 118] предпочитают ассоциировать работы с дополнительной логикой в исполнении последователей. Согласно этой модели работы, заданные в классической постановке, следует интерпретировать как логические операции «И», поскольку все последователи рано или поздно обязаны быть выполнены. Однако рассматриваются также логические операции «ИЛИ» и «Исключающее ИЛИ». Первый тип операций предполагает, что хотя бы один последователь должен быть выполнен. При втором типе выполняется строго один последователь. По мнению авторов, подобная модель может успешно применяться при моделировании проектов научно-исследовательской и опытно-конструкторской деятельности. Например, после работы «Тестирование прототипа», ассоциируемой с операцией «Исключающее ИЛИ», должна быть выполнена либо работа «Доработка прототипа», либо работа «Начало производства». Выполнение одной работы исключает выполнение другой. При этом решение о том, какая работа будет выполняться, принимается после завершения работы предшественника. Часто в литературе подобные зависимости рассматривают в контексте стохастического подхода [119]. Однако, данная модель вполне жизнеспособна и для планирования детерминированных процессов.

В работе [120] рассматриваются иные зависимости. Авторы предлагают использовать явные взаимосвязи между двумя работами i и j, отличные от отношений предшествования. Ими могут быть связаны любые работы i и j, в том числе не имеющие общего предшественника или последователя. Изложенная в работе модель предполагает также использование трёх типов логических операций: «И», «ИЛИ» и «Исключающее ИЛИ». Подобная модель использовалась при планировании наземных процессов в аэропорту, таких как заправка топливом, чистка салона, снабжение питанием. Следует заметить, что описанная модель является более гибкой, чем описанные ранее логические зависимости. Модель также обобщается на случай отношений между двумя и более работами.

1. 5. Целевые функции

1. 5. 1. Минимизация временных показателей проекта

Целевые функции временных показателей являются наиболее распространёнными, так как чаще всего расписание ориентировано на оптимизацию именно временных характеристик проекта. Самой распространённой целевой функцией является уже упомянутая в классической постановке минимизация времени выполнения всего проекта C_{MAX}.

Однако существуют и пользуются достаточной популярностью и другие целевые функции, оперирующие такими временными показателями, как задержка, запаздывание и опережение. Данные понятия тесно связаны с понятием директивного срока d_j выполнения работы j — временем, не позже которого работа должны быть завершена. Исходя из этого, задержкой работы j является значение $L_j = C_j - d_j$. Запаздывание схоже с задержкой с той лишь разницей, что не может быть отрицательным, то есть запаздывание $T_j = \max\{0, (C_j - d_j)\}$. Аналогично определяется и опережение $E_j = \max\{0, (d_j - C_j)\}$.

В некоторых публикациях [38, 98, 109, 121] рассматривается минимизация взвешенного запаздывания. Заметим, что минимизация времени выполнения всего

проекта является частным случаем данной целевой функции. В работе [106] описывается функция минимизации наибольшей задержки и суммарного взвешенного запаздывания, а в [122] обсуждается целевая функция, ориентированная на минимизацию суммы всех взвешенных значений опережения и запаздывания. Такая целевая функция позволяет планировать проект с временем завершения каждой работы предельно близким к соответствующему ей директивному сроку выполнения, то есть «точно в срок». При этом если работу не удаётся выполнить в срок, весовые коэффициенты позволяют нивелировать между опережением и запаздыванием для каждой работы (то есть устанавливать приоритет одного над другим). Иногда минимизации суммы абсолютных значений опережения и запаздывания для каждой работы рассматривается в рамках мультимодальной постановки RCPSP-задачи [123]. Альтернативной целевой функцией может быть минимизация максимального из всех значений опережения и запаздывания. В работе [124] предлагается вариация предыдущей целевой функции с той лишь разницей, что опережение и запаздывание изменяются относительно некоторого определённого отрезка времени («временного окна»), в течение которого работа должна быть выполнена.

В публикации [125] автор предложил определить набор «временных окон» для каждой работы. Исходя из некоторых соображений качества и выгоды, желательно, чтобы работа была выполнена в одно из этих «временных окон». Целевая функция, описанная в работе, ориентирована на минимизацию штрафов, возникающих в случае выполнения работы вне выделенного ей «временного окна». Стимулом для разработки такого подхода послужил один биотехнологический проект. В работе отмечено, что предложенный подход подобен временным ограничениям, налагаемым на работу: «временные окна» могут представлять собой жёсткие временные ограничения.

В работе [38] предлагается минимизировать сумму всех времён завершения работ, в то время как в публикации [126] предлагается минимизировать их взвешенную сумму. Развитием данных подходов является целевая функция минимизации среднего времени потока [127], которое соответствует

среднеарифметическому всех времён завершения работ. Заметим, что функции минимизации общего времени завершения и среднего времени завершения являются эквивалентными.

В нотации $\alpha|\beta|\gamma$ описанные выше целевые функции могут быть представленные в характеристике γ следующими символьными обозначениями: в случаях, когда целевой функцией является минимизация выполнения всего проекта, в характеристике указывается C_{MAX}; минимизация наибольшей задержки — L_{MAX}; минимизация общего взвешенного времени завершения — $\sum w_j C_j$; минимизация общего взвешенного запаздывания — $\sum w_j T_j$. Целевые функции могут быть и более сложные. Например, если необходимо минимизировать общее взвешенное количество запаздывающих работ, то в характеристике γ необходимо задать функцию $\sum w_j U_j$, где $U_j \in \{0,1\}$ является битовым индикатором, который показывает, является ли работа j запаздывающей или нет (если работа запаздывает, то $U_j = 1$; в противном же случае $U_j = 0$). Если, например, необходимо минимизировать общую сумму всех взвешенных опережений и запаздываний, то есть необходимо выполнить каждую работу предельно точно в установленный срок, то γ будет содержать функцию $\sum\left(w_j^E E_j + w_j^T T_j\right)$. Таким образом, как видно, целевые функции временных показателей могут быть достаточно разнообразными и сложными.

1. 5. 2. Устойчивость расписания к задержкам

Во время выполнения проекта могут возникать непредвиденные ранее в расписании задержки при выполнении тех или иных работ. Появление таких задержек может привести к дополнительным временным и стоимостным затратам, что может быть крайне не приемлемо в условиях ограниченного бюджета. Поэтому планировщик проекта и ответственный за проект заинтересованы в построении расписания, устойчивого к непредвиденным задержкам, то есть такого расписания, в котором появление таких задержек будет сказываться только на ограниченном количестве соседних работ или не будет

сказываться вовсе. Такой подход к планированию обычно называют проактивным. Не смотря на то, что большинство современных подходов, позволяющих решать подобные задачи, работают в рамках стохастического концепта, который позволяет явно задавать некоторую неопределённость, в данной работе рассмотрены только детерминистические подходы.

Так в публикации [128] вводиться понятие свободного временного люфта sl_j работы j. Содержательный смысл данной временной характеристики заключается в том, что при появлении в работе j непредвиденной задержки $L'_j \leq sl_j$ она никак не сказывается на выполнении любой другой работы. В этой же публикации описывается двойная целевая функция («бицелевая функция») минимизации общего времени выполнения всего проекта и максимизации общего свободного временного люфта, который может выступать в роли меры надёжности расписания. Такой же подход описан в работе [129]. Иногда на практике используется модифицированный вариант данного подхода: в качестве основной целевой функции иногда предлагается использовать максимизацию наименьшего свободного временного люфта $\min_i sl_i$ [130]. При этом для достижения наименьшей продолжительности всего проекта предполагается использование использовать дополнительного временного ограничения в виде крайнего срока выполнения всего проекта (deadline).

Придерживаясь подхода, изложенного в публикации [128], предпринималась попытка увеличить количество учитываемых факторов [131]. В первую очередь рассматривались кроме свободного временного люфта ещё и суммарное количество всех непосредственных последователей и/или суммарное количество всех используемых ресурсов. Кроме того предлагались варианты данного подхода, в которых вышеперечисленные характеристики использовались вместе с весовыми коэффициентами, а свободный временной люфт заменяется бинарным значением $\alpha_j \in \{0,1\}$, которое выступало в роли индикатора наличия данного люфта (если $sl_j = 0$, то $\alpha_j = 0$; если $sl_j > 0$, то $\alpha_j = 1$). Суть последнего подхода заключается в том, чтобы избежать необъективности, возникающей в

случае очень больших значений временных люфтов. Использование больших люфтов не имеет практической значимости, в то время как гораздо меньшего люфта может быть достаточно для компенсации типовых задержек.

В работе [132] вводиться понятие переработки (то есть выполнение работы дольше запланированной продолжительности). Значение RT_{jt} является ожидаемым временем переработки (то есть задержкой) для работы j при условии, если работа будет завершена в момент времени t; RC_{jt} — ожидаемые затраты из-за переработки. В соответствии с сущностью данных понятий предполагается, что RT_{jt} и RC_{jt} возрастают со временем. Целевой функцией в данной публикации является минимизация суммы времени и стоимости переработки.

Таким образом использование подобных целевых функций позволяет повысить устойчивость расписания к непредвиденным задержкам и минимизировать затраты связанные с их устранением.

1. 5. 3. Обеспечение консервативности расписания

Часто в ходе выполнения проекта возникают непредвиденные ситуации (например, сверхбольшие задержки или перебои с поставками ресурсов), в результате которых дальнейшее выполнение проекта в соответствии с составленным расписанием не представляется возможным. В таких случаях расписание становиться неактуальным и не соответствующим сложившимся условиям. Такое расписание должно быть пересмотрено и перепланировано. Специфика данного перепланирования заключается в том, что проект уже находиться на стадии выполнения: некоторые работы могут быть уже выполнены и должны быть проигнорированы в ходе выполнения перепланирования; некоторые работы начаты, но ещё не закончены — такие работы участвуют в процессе перепланирования, но имеют фиксированные времена начала и ожидаемого завершения своего выполнения. Кроме того при перепланировании может измениться количество и характер доступности ресурсов. В отличие от проактивного планирования, в котором построение устойчивого к задержкам

расписания главным образом сводиться к закладыванию непредвиденных задержек в само расписание, здесь имеет место возникновение непредвиденных событий, выходящих за рамки заложенной надёжности и требующие перестроения всего расписания. Такое планирование обычно называют реактивным.

Основной целевой функцией при перепланировании проекта в большинстве случаев является минимизация вносимых изменений. Этот критерий тесно связан с тем фактом, что ранее представленное расписание, как правило, удовлетворяло все заинтересованные стороны и вынуждало их планировать свою деятельность в соответствии с данным расписанием. Сильно изменённое расписание может привести к дополнительным накладным затратам, задержкам, изменениям в ресурсо-обеспечении или даже к срыву проекта в целом.

В работе [82] предлагается минимизировать вносимые в оригинальное расписание изменения путём минимизации количества работ, которые получать в новом расписании новое значение времени начала своего выполнения. А в публикации [133] предлагается перепланировать расписание так, чтобы минимизировать сумму всех отклонений нового времени завершения каждой работы от первоначального значения. Данная задача может быть решена в рамках постановки задачи «выполнить точно в срок», где минимизируется взвешенная сумма всех опережений и запаздываний каждой работы. Для редуцирования задачи необходимо оригинальные значения времени завершения каждой работы рассматривать в качестве соответствующих директивных сроков их выполнения. Иногда имеет смысл измерять и минимизировать степень нарушения расписания, суммируя все отклонения значений начала и завершения каждой из работ [134].

Отдельно стоит отметить особенности перепланирования проектов, выполняемых в рамках мультимодальной постановки RCPSP-задачи [135]. Обычно в таких постановках принято штрафовать все изменения количества и интенсивности использования ресурсов. Кроме того штрафовать предлагается и изменения режимов выполнения каждой из работ. Некоторые особо критические работы, могут быть зафиксированы с помощью введения дополнительных

ограничений. Это может быть очень удобно и полезно в случаях с работами, запланированными в непосредственной близости от того момента времени, в который возникла необходимость в перепланировании проекта, то есть работами, которые должны вот-вот начать своё выполнение. Часто на практике вносить изменения в такие работы и, например, откладывать их выполнение не целесообразно. Кроме того данный подход может применяться в случаях, когда необходимо, чтобы новое составленное расписание в какой-то определённый момент времени догнало первоначальное расписание (восстановилось).

1. 5. 4. Минимизация затрат на возобновимые ресурсы

В современной тематической литературе достаточно широко обсуждаются различные целевые функции, связанные с возобновимыми ресурсами. Наиболее обширный и подробный обзор данных функция приведён в публикации [136]. Здесь будут рассмотрены только наиболее популярные и применяемые функции.

Одной из самых распространённых является функция, ориентированная на решение так называемой задачи ресурсных инвестиций. Данная задача является зеркальной копией классической RCPSP-задачи. Так в классической постановке RCPSP-задачи основным является минимизация общего времени выполнения всего проекта, не превышая при этом заданного лимита использования каждого из возобновимых ресурсов. В данной же задаче основным является минимизация количества используемых возобновимых ресурсов, не превышая при этом некоторый заданный крайний срок выполнения всего проекта (deadline). Целевой функцией в данной задаче будет минимизация суммы стоимостных затрат на поддержание доступности требуемого количества каждого ресурса, то есть $\sum_k C_k(R_k)$, где R_k — выделенное количество ресурса k (квота), $C_k(R_k)$ — стоимость выделения ресурса k в количестве R_k. Очевидно, что $C_k(R_k)$ является неубывающей функцией, прямо зависящей от размера выделяемой квоты. В частном и наиболее распространённом случае данная стоимостная функция является линейной, то есть $C_k(R_k) = c_k R_k$, где c_k — стоимость поддержания одной единицы ресурса k в доступном состоянии.

Задача ресурсных инвестиция были успешно решена в ряде работ [105, 106, 137, 138, 139]. Стоит отметить работу [140] в которой представлена расширенная задача ресурсных инвестиций: вместо крайнего срока завершения проекта (deadline) в ней используется директивный срок, что допускает появления задержки завершения всего проекта. Целевой функцией является минимизация общей суммы стоимостных затрат на поддержание ресурсов и на штраф за запаздывание. Штраф за запаздывание проекта прямо пропорционален величине запаздывания.

Иногда основное внимание уделяется арендуемым возобновимым ресурсам [141]. Основная особенность в данном случае заключается в том, что возобновимый ресурс берётся в аренду, стоимость которой прямо зависит от количества арендуемого ресурса и периода аренды. Стоимость аренды на период t одной единицы ресурса k складывается из фиксированной стоимости c_k^f за факт аренды единицы ресурса и из стоимости аренды c_k^v единица ресурса за единицу времени (ставка аренды). Таким образом, если необходимо взять в аренду r_k единиц ресурса k на время t, то стоимость аренды составит $r_k\left(c_k^f + c_k^v t\right)$. Очевидно, что целевой функцией в данной постановке является минимизация суммы всех стоимостных затрат на аренду ресурсов. Как можно заметить, если ставка аренды будет нулевой ($c_k^v = 0$), то данная задача сводиться к задаче ресурсных инвестиций. Более подробно данная задача рассмотрена в работе [142].

Другой не менее важной задачей является выравнивание загруженности ресурсов. Суть данной задачи сводиться к минимизации изменений интенсивности использования ресурсов при переходе от одного момента времени к другому соседнему, не превышая при этом крайнего срока выполнения проекта (deadline). Если интенсивность использование ресурсов представить в виде графика, то целью данной задачи является предельное сглаживание данного графика. В работах [38, 105, 106] для решения данной задачи в роли целевой функции выступила минимизация наибольшего изменения или сумма всех

изменений интенсивности использования ресурсов, а в работе [143] предлагается минимизировать для этой цели сумму всех изменений, возведённых в квадрат.

В некоторых случаях полезно минимизировать используемое количество только тех возобновимых ресурсов, потребление которых превышает некоторый заданный уровень [105, 121, 144]. Или, как в работе [38], иногда рассматривается минимизация накапливаемого отклонения уровня использования ресурсов от некоторого заданного константного уровня. Такой подход позволяет контролировать и устранять не только перерасход ресурсов, но и возможные простои. Интересный подход изложен в работе [54], где предлагается минимизировать как количество, так и продолжительность разрывов в графике (профиле) потребления ресурсов. Очевидно, что данная стратегия также ориентирована на снижение простоя ресурсов. В более развитых постановках [45] предлагается разделить всё доступное количество возобновимых ресурсов на количество, доступное изнутри, и количество, доступное извне. Целевая функция при такой постановке предполагает минимизацию расходов, связанных с использованием только взятых извне ресурсов.

В нотации $\alpha|\beta|\gamma$ все описанные выше целевые функции для решения задачи по выравниванию загруженности ресурсов можно обобщенно представить в характеристике γ как $\sum c_k f\big(r_k(S,t)\big)$, где c_k — стоимость использования одной единицы ресурса k, f — функция профиля использования ресурса k в количестве $r_k(S,t)$ (t — момент времени в расписании S). Следует заметить, что если $f = R_k$, что соответствует равномерному профилю потребления ресурса k, то данная задача (задача выравнивания загруженности ресурсов) сводиться к задаче ресурсных инвестиций.

1. 5. 5. Минимизация невозобновимых ресурсов

Минимизация потребления невозобновимых ресурсов имеет место только в мультимодальной постановке RCPSP-задачи, так как во всех других постановках потребление ресурсов задачами фиксировано, и, следовательно, сумма

потребления невозобновимых ресурсов всегда будет одинаковой. Во всём остальном целевые функции, ориентированные на невозобновимые ресурсы являются, фактически, точной копией возобновимых ресурсов. В традиционной мультимодальной постановке RCPSP-задачи минимизируется время выполнения всего проекта, не превышая при этом предела доступности ресурсов. Если же необходимо минимизировать потребление невозобновимых ресурсов, то аналогично задаче ресурсных инвестиций в постановке задачи вводить крайний срок выполнения проекта (deadline), который не должен быть превышен.

Такая целевая функция, ориентированная на невозобновимые ресурсы, была подробно рассмотрена в публикациях [93, 94] в рамках дискретной задачи компромисса времени и стоимости. В данных публикациях невозобновимые ресурсы интерпретировались в качестве денег. Аналогичный подход изложен и в работах [38, 88]. А вот в публикациях [38, 121] предлагается минимизировать количество только тех используемых невозобновимых ресурсов, суммарное потребление которых превышает некоторый заданный предел.

Все эти целевые функции можно обобщённо представить в нотации $\alpha|\beta|\gamma$ как $\gamma = \sum c_k f\big(r_k(S)\big)$, где c_k — стоимость потребления одной единицы невозобновимого ресурса k, f — функция потребления ресурса k в количестве $r_k(S)$ в расписании S.

1. 5. 6. Минимизация общей стоимости проекта

Оптимизация расписания по критерию стоимости часто является важнейшим аспектом. Как правило, такая оптимизация выполняется в рамках RCPSP-задачи без ресурсов. Обычно для каждой работы j вводиться понятие стоимости её выполнения c_{jt}, зависящее от момента времени t, когда работа j начала своё выполнение [145, 146, 147]. Как верно замечено в работе [147], целевая функция, ориентированная на стоимость, является обобщением разных других хорошо известных целевых функций, таких как минимизация времени

выполнения всего проекта, выполнение работ точно в срок или максимизация чистой приведённой стоимости.

Обычно минимизируют общую стоимость проекта, которая состоит из стоимостей опережений и запаздываний относительно директивных сроков выполнения работ, а также из стоимостей, связанных с продолжительностью выполнения работ [148]. Более экзотические постановки предполагают, что продолжительность выполнения работы может быть сокращена за дополнительную плату. При этом проект должен быть завершён не позднее некоторого заданного крайнего срока (deadline).

В публикации [149] рассматривается минимизация стоимости, состоящей из стоимостей сбоев выполнения работ (то есть неожиданного сокращения продолжительности их выполнения), стоимостей материальных (невозобновимых) ресурсов и расходов на поддержание инвентаря (возобновимых ресурсов). Также в данной работе рассматриваются бонусы и штрафы за раннее и за позднее (относительно некоторого заданного директивного срока) завершение проекта соответственно.

Интересная постановка представлена в публикации [150]. Авторы рассмотрели постановку с работами, продолжительность выполнения которых находиться в некотором заданном интервале, едином для всех работ. Такой подход привёл авторов к целевой функции, в которой стоимость выполнения каждой работы, связанная с их продолжительностью, является выпуклой функцией.

Также заслуживает внимания двухкомпонентная целевая функция [151]. Стоимость выполнения проекта, заложенная в данную целевую функцию, прямо пропорциональна продолжительности выполнения проекта. При этом работы в проекте могут быть скомбинированы (объединены) для уменьшения продолжительности выполнения проекта, а стоимость выполнения данных работ при этом будет складываться. Данная целевая функция может быть использована только в рамках постановки без ресурсных ограничений.

1. 5. 7. Максимизация чистой приведенной стоимости

Другой не менее важный тип целевой функции появляется, когда в ходе выполнения проекта имеют место финансовые потоки. Отток денежных средств происходит в результате выполнения какой-либо работы или использования какого-либо ресурса. С другой стороны, приток денежных средств происходит в результате инвестиций, авансовых платежей или после достижения определённых ключевых точек выполнения всего проекта. Также не исключён вариант, когда приток денежных средств осуществляет один раз в конце выполнения проекта (оплата за выполнение проекта). Чистой приведённой стоимостью является разница между суммами всех притоков и оттоков денежных средств. Этот показатель является одним из важнейших для инвесторов. Поэтому целевой функций должна быть максимизация чистой приведённой стоимости.

Данная задача решается в рамках стандартных ограничений RCPSP-задачи. Обычно максимизации подвергается общая сумма оттоков и притоков денежных средств, при этом оттоки участвуют в сумме со знаком «минус». Кроме того в данной постановке допускается участие скидок при финансовых оттоках. Скидки могут быть как фиксированными, так и зависящими от каких-либо проектных показателей. Всё это детально рассматривается в работах [152, 153, 154, 155]. Кроме того следует отметить работы [46, 156], в которых исследуется максимизация чистой приведённой стоимости в рамках мультимодальной постановки, а также [157] — в рамках задачи ресурсных инвестиций. Так же данной целевой функции посвящены публикации [61, 104, 158, 159, 160].

В нотации $\alpha|\beta|\gamma$ целевая функция максимизации чистой приведённой стоимости может быть представлена как $\sum c_j^F \beta^{F_j}$, где c_j^F — денежный поток (приток или отток) работы j, β^{F_j} — скидка, предоставляемая в данном денежном потоке.

1. 5. 8. Многоцелевые функции

Во всех ранее обсуждаемых случаях применялась целевая функция, оптимизирующая только один параметр или характеристику проекта, в то время как все остальные при необходимости контролировались с помощью введения дополнительных ограничений. Однако в реальности часто необходимо оптимизировать выполнение проекта по нескольким критериям, то есть применить многоцелевую функцию.

Так наиболее частым и распространённым способом получения многоцелевой функции является взвешенное суммирование всех интересующих параметров и/или характеристик. Обычно взвешенному суммированию подвергаются общее время выполнения проекта, опережение и запаздывание работ, равномерность загруженности ресурсов и потребление ресурсов в целом. Такой подход можно встретить во многих работах [38, 54, 128].

Другим получивший широкое распространение способом получения многоцелевой функции является генерация расписаний, оптимизированных с точки зрения закона Парето. Общий принцип данного закона прост: 20% приложенных усилий дают 80% результата, а остальные 20% результата достигаются остальными 80% усилий. Данный принцип применительно к методам построения расписания был достаточно подробно изучен и успешно применён в некоторых публикациях [144, 161, 162, 163]. Особенно стоит отметить работу [164], в которой авторами представлена оригинальная многоцелевая функция, основанная на системе знаний.

Таким образом, многоцелевые функции, с одной стороны, являются наиболее применимыми в реальных условиях, в другой — наиболее сложными. При этом многоцелевые функции по своей природе является составными, что позволяет достаточно гибко выбирать оптимизируемые критерии проекта.

Глава 2. Математическая формализация задач проектного планирования в расширенной постановке

Задачи проектного планирования RCPSP (Resource-Constrained Project Scheduling Problem) в значительной степени обобщают постановки задач теории расписаний, и поэтому алгоритмы проектного планирования часто рассматриваются в качестве универсальных инструментов для их решения. Тем не менее, в зависимости от индивидуальных характеристик задачи вычислительная сложность составления расписания может существенно варьироваться и поэтому для частных постановок обычно применяют специальные алгоритмы. Например, как уже отмечалось ранее, задача проектного планирования RCPSP, являющаяся NP-полной, редуцируется к частным постановкам «открытой линии», «рабочего цеха» или «потоковой линии», имеющим полиномиальную сложность при небольшом числе машин, простых моделях обслуживания и отсутствии директивных сроков.

Вместе с тем, на практике возникает необходимость в обобщении задач проектного планирования, обусловленная особенностями осуществления проектной деятельности разными организациями и индустриальными сообществами, различиями в представлении проектных данных, а также альтернативными способами математической формализации задач проектного планирования. Большое число существующих программных приложений для календарно-сетевого планирования и управления проектами с близкими функциями подтверждает этот факт. В качестве таких приложений следует указать популярные программные системы Oracle Primavera, MS Project, Synchro, Spider Project, Gemini, Merlin, Zoho Projects, ManagePro, обеспечивающие автономную работу планировщиков и управленческого персонала на

изолированных компьютерах или групповую работу в корпоративной сети. Ряд систем конфигурируется в виде универсальных Интернет-сервисов и web-клиентов к ним. К подобным решениям относятся сервисы Smartsheet, GanttPro, Asana, Acunote, Teamweek, Bitrix24, Jira, ISETIA. Примечательно, что почти все приложения реализуют алгоритмы для поиска критического пути или вычисления критических работ, а большая часть приложений — алгоритмы решения задач RCPSP в расширенных постановках. В типовых приложениях речь идет о проектных планах, содержащих десятки тысяч работ, поэтому для решения задач RCPSP применяются приближенные, основанные на эвристиках алгоритмы полиномиальной сложности. Применение точных методов при составлении оптимальных расписаний для проектных планов с числом работ выше сотни не представляется возможным из-за экстремально высокой вычислительной сложности задач RCPSP.

В настоящей главе предпринимается попытка математического обобщения и формализации задач проектного планирования с учетом особенностей расширенных постановок RCPSP, встречающихся в программных приложениях. Ожидается, что полученные результаты позволят разработать архитектуру типового приложения теории расписаний, разработать программно-инструментальную среду общего назначения, предлагающую данную архитектуру и обеспечивающую задание условий и эффективное решение разнообразных классов задач проектного планирования, а также разработать метод построения и инкрементального развития приложений на основе данной среды. Развитые инструментальные возможности среды позволят относительно просто адаптировать ее к новым постановкам прикладных задач.

В разделе 2. 1 приводится формализация классической постановки задач RCPSP, и кратко упоминаются алгоритмы их приближенного решения. Недостатки и ограничения математической постановки детально обсуждаются в разделе 2. 2, в котором особое внимание уделяется вопросам структуризации проектного плана и моделям исполнения работ. Кратко обсуждаются вопросы математической формализации, связанные с заданием альтернативных целевых

функций оптимизационной задачи и возможным переопределенным характером системы наложенных алгебраических ограничений. Раздел 2. 3 посвящён математической формализации задачи обобщённого проектного планирования GCPSP (Generally Constrained Project Scheduling Problem), а также доказательству некоторых утверждений о достаточных условиях существовании решения. В разделе 2. 4 описывается алгоритм приближенного решения задач GCPSP, который можно рассматривать в качестве развития известного алгоритма последовательной диспетчеризации. Формулируется утверждение об эквивалентности алгоритмов в случаях, когда задача обобщённого проектного планирования редуцируется к классической постановке RCPSP.

2. 1. Формализация классической постановки RCPSP-задач

Рассмотрим классическую постановку задач RCPSP, следуя работам [165, 166, 10].

Пусть проект состоит из N работ, R возобновимых ресурсов и L связей с соответствующими индексами $i = 1,2,\dots,N$, $r = 1,2,\dots,R$ и $l = 1,2,\dots,L$ соответственно. Пусть задано время старта проекта t_0, а для каждой работы i определено ее время выполнения $d_i \geq 0$ и количества потребляемых ей ресурсов $q_{i,r} \geq 0$. Требуется найти расписание проекта (время старта каждой работы x_i, $i = 1,2,\dots,N$), при котором минимизируется время выполнения всего проекта.

Уточним условия задачи. Потребление работой i ресурса r в количестве $q_{i,r}$ означает, что с началом выполнения работы данное количество ресурса становится недоступным для использования в других работах, а с ее окончанием — высвобождается. Одна работа может потреблять несколько разных ресурсов. Один ресурс может одновременно использоваться несколькими работами при условии, что его суммарное потребление не превышает доступное количество $Q_r > 0$. Пусть $I(t,r) = \{i = 1,..,n \mid x_i \leq t < x_i + d_i \;\&\; q_{i,r} \neq 0\}$ — множество индексов работ, выполняемых в момент времени $t \geq t_0$ и использующих ресурс r.

Тогда условие корректного потребления ресурса r и необходимое условие существования решения задачи может быть выражено следующим образом:

$$\sum_{i \in I(t,r)} q_{i,r} \leq Q_r$$

Наконец, связи определяют бинарные отношения предшествования между работами таким образом, что работа-последователь не может стартовать до того, как работа-предшественник завершится. Пусть $i_{(l)}$ — индекс работы-предшественника, участвующего в связи l, а $i^{(l)}$ — индекс работы-последователя, участвующего в связи l. Тогда данное отношение имеет вид $x_{i_{(l)}} + d_{i_{(l)}} \leq x_{i^{(l)}}$. В дальнейшем будем записывать $i \to j$, если существует связь l, такая что $i = i_{(l)}$ и $j = i^{(l)}$.

Тогда задача RCPSP формализуется следующим образом:

$$\min \left(\max_{i=1,\dots N} (x_i + d_i) \right)$$

$$\text{при } x_i \geq t_0, i = 1, \dots, N$$

$$x_{i_{(l)}} + d_{i_{(l)}} \leq x_{i^{(l)}}, l = 1,2, \dots, L$$

$$\forall\, t \geq t_0 \text{ имеет место } \textstyle\sum_{i \in I(t,r)} q_{i,r} \leq Q_r, r = 1,2, \dots, R$$

Неизвестные переменные в постановке задачи обычно целочисленные, поскольку на практике обычно используется дискретное представление временных параметров. Задача считается корректно поставленной, если заданные связи не образуют циклов и уровни потребления ресурсов индивидуальными работами не превышают их общедоступное количество. В противном случае система алгебраических ограничений может оказаться переопределенной, а задача — не иметь решений. В случае отсутствия ресурсных ограничений задача RCPSP естественным образом редуцируется к задаче поиска критического пути или вычисления критических работ.

На сегодняшний день существует несколько популярных алгоритмов приближенного решения задач RCPSP. Наиболее известным является последовательный алгоритм составления расписания, иногда называемый

алгоритмом последовательной диспетчеризации [10, 167]. Он позволяет за фиксированное количество шагов построить согласованное расписание или убедиться в его отсутствии, если в проектном плане есть неразрешимые ограничения. На каждом шаге алгоритма выбирается одна из работ, предшественники которой уже обработаны и для них определены времена старта и завершения. Для выбранной на основе эвристических правил работы вычисляется наиболее ранее допустимое время старта и осуществляется переход к следующей работе. В предположении отсутствии циклов последовательность работ с обработанными предшественниками всегда существует, а для работ всегда может быть определено время старта, согласованное со всеми наложенными ограничениями.

В классическом варианте алгоритм предполагает использование множества обработанных работ S и множества необработанных работ D, все предшественники которых входят в S [165]. На каждом шаге алгоритма множества корректируются в соответствии с приведенным ниже псевдокодом:

Begin
  $S \coloneqq \emptyset$;
  While $|S| < N$ **do**
    $D \coloneqq \{i = 1,..,N | i \notin S \,\&\, j \in S \;\forall\, j \to i\}$;
    For $r \coloneqq 1$ **to** R **do**
      $K_r \coloneqq \left\{Q_r - \sum_{i \in I(\tau,r)} q_{i,r} \,\middle|\, \tau = t_0, t_0 + 1, \dots, T\right\}$;
    End for;
    $i^* \coloneqq \operatorname{argmax}_{i \in D}\{v(i)\}$;
    If $\nexists\, j \to i^*$ **then**
      $t_{min} \coloneqq t_0$;
    Else
      $t_{min} \coloneqq \max_{j \to i^*}\left(x_j + d_j\right)$;
    End if;
    $x_{i^*} \coloneqq \min\left\{t \middle| t \geq t_{min} \,\&\, q_{i,r} \leq K_r(\tau), \tau = t, t + 1, \dots, t + d_{i^*}, r = 1,2, \dots, R\right\}$;
    $S \coloneqq S \cup \{i^*\}$;
  End while;
End;

Здесь T — время, на протяжении которого выполняется расписание, K_r — вспомогательное множество значений доступного ресурса в дискретные моменты

времени, $v(i)$ — эвристическое правило, определяющее приоритет и очередность планирования работы i.

Алгоритм относительно прост и допускает многочисленные варианты, что делает его привлекательным для программной реализации. Один из вариантов заключается в попытке составить расписание в предположении отсутствия ресурсных ограничений и определить ранние и поздние даты выполнения работ, как это делается при поиске критического пути [167]. Затем проводится анализ ресурсных ограничений с учетом допустимых задержек индивидуальных работ. Можно ожидать, что для большей части работ ресурсные ограничения разрешатся таким способом. При этом следует предусмотреть возможность перезапустить или продолжить процесс составления расписания для оставшихся конфликтных работ и их последователей. Иначе применение алгоритма на практике может оказаться довольно рискованным. Другой известный вариант алгоритма основан на выделении групп несвязанных между собой работ и на их одновременном анализе [165]. Было установлено, что в ряде случаев он приводит к более качественным приближенным решениям. Однако реализация данного варианта сопряжена с дополнительными расходами на перераспределение работ по группам и согласование частных расписаний, что может оказаться критичным для больших проектных планов.

Все упомянутые выше алгоритмы имеют полиномиальную сложность и обеспечивают поиск приближенных решений для индустриально значимых задач, размерность которых может составлять десятки и сотни тысяч неизвестных переменных. На практике часто требуется удостовериться в качестве получаемых решений, для чего желательно получить оптимальные решения, хотя бы для подобных задач низкой размерности. В этом случае можно применить точные комбинаторные алгоритмы с известными оптимизационными приемами типа «альфа-бета-отсечения» или «метода ветвей и границ» [168]. Например, лучший из известных точных алгоритмов Брукера за приемлемое время может решать задачи размерности не больше 60 [169]. По результатам сравнения приближенных и точных решений можно скорректировать параметры применяемых алгоритмов и

настроить их эвристики с тем, чтобы обеспечить надлежащее качество результатов и при решении задач высокой размерности. К сожалению, теоретические исследования дают слишком грубые оценки, не позволяющие судить о качестве приближенных решений, полученных известных алгоритмами.

2. 2. Ограничения классической постановки

Хотя приведенная математическая постановка RCPSP является классической для теории расписаний и прикладных дисциплин, связанных с календарно-сетевым планированием и управлением проектной деятельностью, она имеет ряд принципиальных ограничений для широкого практического использования. На необходимость решения задач проектного планирования в расширенной постановке указывают авторы работы [10]. В предыдущей главе достаточно детально рассмотрены особенности прикладных задач, не учитываемые классической постановкой. В настоящем разделе будет неформально определён класс задач обобщённого проектного планирования, который мог бы допускать альтернативные критерии и сложные модели исполнения работ, наложения связей, привлечения ресурсов, календарей, финансов при составлении расписаний. В конечном итоге подобные расширения приводят к новым видам целевых функций и функциональных ограничений для решаемых оптимизационных задач.

Опишем главные отличия, характеризующие обсуждаемый класс задач, более систематическим образом.

Во-первых, в нем допускается задание работ различных видов, таких как простые активности, вехи начала и завершения работ, «короткие гамаки», «длинные гамаки», а также составные работы, обеспечивающие иерархическую многоуровневую структуризацию проектного плана. При этом предполагается, что работы могут исполняться в обычном режиме, в режиме с прерываниями, с профильным использованием ресурсов, с учетом накладных временных затрат, в альтернативных мультимодальных режимах, а также с учетом компромиссов. При

задании условий задачи может указываться также статус работ, как планируемых, стартованных, прерванных, возобновленных или завершенных, а также процент выполнения для незавершенных работ.

Во-вторых, взаимосвязи работ могут устанавливать отношения предшествования не только между окончанием предшествующей работы и началом последующей работы, но также и между любыми событиями, связанными с их началом и завершением. Для формируемых взаимосвязей «начало-начало», «начало-окончание», «окончание-начало» и «окончание-окончание» могут быть установлены минимальные и максимальные величины задержки, пересчитанные в календарные даты с использованием рабочих календарей проекта. Примечательно, что величины задержек могут отрицательными, а взаимосвязи могут устанавливаться и для составных работ, обеспечивая тем самым возможность составления расписаний при крупноблочной структуризации проектного плана и при его последующей детализации. Важным аспектом составления расписания для актуализируемого проектного плана является учет нарушенных взаимосвязей и возможность восстановить отношения предшествования, хотя бы для оставшихся частей незавершенных работ. Подобные ситуации приводят к неоднозначной интерпретации ограничений и нуждаются в более строгой формализации.

В-третьих, исполнение работ и привлечение ресурсов в рамках проектной деятельности обычно осуществляется на основе календарей, определяющие графики работы. В обсуждаемых задачах предполагается использование модели календарей, в которой рабочие интервалы могут задаваться на основе регулярных и исключительных правил с требуемой точностью и дискретизацией представления временных параметров. При фиксировании трудоемкости работы и доминирующего ресурса ее продолжительность может определяться доступным количеством ресурса, а не только рабочим календарем.

В-четвертых, для работ могут быть заданы явные временные ограничения, определяющие алгебраические условия их начала и завершения. Ограничения могут задаваться условиями типа «начать не раньше», «завершить не позже»,

«начать в фиксированную дату» и т.п., так и спецификаторами «выполнить как можно раньше» или «выполнить как можно позже». Кроме того, каждой работе могут быть приписаны правила выравнивания, устанавливающие условия начала или завершения работы строго в начале или конце рабочего интервала (часа, дня, недели, месяца, года и т.п.) независимо от возможности исполнить работу несколько раньше или позже. Важной особенностью обсуждаемых задач является возможность задания временных ограничений для составных работ, что, как и в случае с взаимосвязями, обеспечивает возможность составления расписаний при крупноблочной структуризации проектного плана. Наконец, система наложенных алгебраических ограничений может оказаться переопределенной и не иметь решений. Это означает, что должен быть задан непротиворечивый способ разрешения подобных ситуаций, например, с использованием приоритетов, приписанных индивидуальным ограничениям.

В-пятых, ресурсная модель должна допускать использование невозобновимых, ограничено-возобновимых, частично возобновимых, логистических, непрерывно разделяемых, исключительных ресурсов, а также ресурсов с переменной доступностью. Возобновимые ресурсы обычно моделируют использование персонала и техники, которые могут объединяться в «бригады». Ресурсы, связанные с персоналом определенной специализации и квалификации, могут формировать соответствующие «компетенции». Невозобновимые ресурсы обычно ассоциируют с расходными материалами, для которых могут определяться цепочки поставок. Финансовое обеспечение проектной деятельности также может моделироваться невозобновимыми ресурсами, ассоциируемыми, например, с банковскими счетами участвующих в проекте организаций. Количество доступных ресурсов может быть нефиксированным и зависеть от времени, например, как в случае поставок материалов или привлечения дополнительных инвестиций в ходе реализации проекта.

В-шестых, допускается выбор альтернативных целевых функций для минимизации временных показателей проекта, обеспечения консервативности и

устойчивости расписания к задержкам, минимизации затрат на возобновимые ресурсы, минимизации невозобновимых ресурсов, минимизации общей стоимости проекта, максимизации чистой приведенной стоимости, а также достижения многокритериальных показателей проекта.

В-седьмых, задача проектного планирования может решаться в инкрементальной постановке, когда требуется скорректировать расписание с учетом измененных условий исходной задачи или при актуализации данных непосредственно на проектной площадке в режиме реального времени.

2. 3. Математическая формализация GCPSP-задач

Обсудим предлагаемый способ математической формализации задач проектного планирования. Следуя ему, оптимизационная задача ставится на множестве допустимых решений, связанных с частной задачей удовлетворения ограничений с приоритетами (или предпочтениями). В отличие от RCPSP в предлагаемой постановке не уточняется семантика неизвестных переменных, например, как времен начала, завершения, прерывания и возобновления работ или их продолжительностей. Также не конкретизируем вид целевой функции и функциональных ограничений, например, в виде явных временных или ресурсных условий. Вместо этого определяем класс задач GCPSP в математически нейтральной форме в рамках общих предположений относительно свойств целевой функции, вида алгебраических ограничений и способа их разрешения относительно тех или иных переменных. Таким способом определяемый класс задач GCPSP расширяет условия классической постановки RCPSP и удовлетворяет основным требованиям к прикладным постановкам, перечисленным в предыдущем разделе.

Определение 1

Четвёрку множеств $\langle D, X, C, P \rangle$ назовем задачей в ограничениях с приоритетами, где $D = \{D_1 \times D_2 \times \ldots \times D_n\}$ — домен, определяющий область допустимых значений множества переменных задачи $X = \{x_1, x_2, \ldots, x_n\} \in D$,

$x_i \in D_i,\ i = 1,2,\dots,n;\ C = \{c_1(X_1), c_2(X_2), \dots, c_m(X_m)\}$ — множество предикатов $c_j(X_j),\ j = 1,2,\dots,m,$ определенных на подмножествах переменных $X_j = \left\{x_{i_1^{(j)}}, x_{i_2^{(j)}}, \dots, x_{i_{k_j}^{(j)}}\right\} \subseteq X$ и называемых ограничениями задачи; $P = \{p_1, p_2, \dots, p_m\}$ — множество натуральных чисел $p_j \in N$, называемых приоритетами ограничений. Пусть $I(X_j) = \{i_1^{(j)}, i_2^{(j)}, \dots, i_{k_j}^{(j)}\} \subseteq \{1,2,\dots,n\}$ — множество индексов подмножества переменных X_j. Тогда будем говорить, что переменная задачи x_i, $1 \le i \le n$ участвует в ограничении $c_j(X_j),\ 1 \le j \le m$ или ограничение ассоциировано с переменной, если $i \in I(X_j)$. Ограничение $c_j(X_j)$ будем называть удовлетворённым при любом множестве значений переменных $X_j^* \subseteq X^* \in D$, при котором соответствующий предикат принимает значение «верно».

Определение 2

Пусть $\langle D, X, C, P\rangle$ — задача в ограничениях с приоритетами. Значения переменных $X^* \in D$, при которых все ограничения задачи удовлетворены, будем называть решением задачи. Пусть $C^*(X)$ — множество ограничений, которые удовлетворены при значениях переменных X. Тогда X^* является решением задачи, только если $C^*(X^*) = C$.

Определение 3

Пусть $\langle D, X, C, P\rangle$ — задача в ограничениях с приоритетами. Значения переменных $X^* \in D$ будем называть согласованным решением задачи в ограничениях с приоритетами, если, по крайней мере, хотя бы одно ограничение удовлетворено и не существует других значений переменных Y, повышающих максимальный приоритет разрешенных ограничений: $C^*(X^*) \neq \emptyset$ и для любого неразрешенного ограничения $c_j \in C \backslash C^*(X^*),\ p_j \in P$ не существует значений $Y = \{y_1, y_2, \dots, y_n\} \in D,\ Y \neq X^*$, таких что удовлетворяется каждое ограничение $c_k \in C^*(Y), p_k \in P$ с приоритетом $p_k \ge p_j$.

Определение 4

Пусть $\langle D, X, C, P\rangle$ — задача в ограничениях с приоритетами. Значения переменных $X^* \in D$ будем называть локально согласованным решением задачи в ограничениях с приоритетами, если, по крайней мере, хотя бы одно ограничение удовлетворено и согласованы приоритеты разрешенных ограничений относительно каждой переменной: $C^*(X^*) \neq \emptyset$, а также для любой переменной $x_i \in X$ и для любого ассоциированного с ней неудовлетворенного ограничения $c_j(X_j) \in C \backslash C^*(X^*)$, $p_j \in P$, $i \in I(X_j)$ не существует значения переменной $y_i \in D_i$, $y_i \neq x_i$ такого, что удовлетворяется каждое ассоциированное с переменной ограничение $c_k \in C^*(Y)$, $p_k \in P$, $i \in I(X_k)$ с приоритетом $p_k \geq p_j$, где множество значений переменных $Y = \{x_1^*, x_2^*, \dots, y_i, \dots, x_n^*\}$.

Определение 5

Задачей обобщённого проектного планирования GCPSP будем называть задачу оптимизации целевой функции проекта на множестве локально согласованных расписаний. Для определенности ограничимся следующей математической постановкой

$$\min f(X)$$

$$X \in D^*$$

где $f(X)\colon Z^n \to R$ — целевая функция, $D^* \subseteq Z^n$ — множество локально согласованных решений системы ограничений $c_j(X_j)$, $X_j = \left\{x_{i_1^{(j)}}, x_{i_2^{(j)}}, \dots, x_{i_{k_j}^{(j)}}\right\} \subseteq X$ с приоритетами $p_j \in N$, $j = 1,2,\dots,m$.

Предполагается, что ограничения разрешимы относительно переменных одним из нижеприведенных способов:

(1) $c_j(X_j) \Leftrightarrow x_{i_1^{(j)}} = g_j\left(x_{i_2^{(j)}}, x_{i_3^{(j)}}, \dots, x_{i_{k_j}^{(j)}}\right)$ (ограничения А)

(2) $c_j(X_j) \Leftrightarrow x_{i_1^{(j)}} \in D_j^+\left(x_{i_2^{(j)}}, x_{i_3^{(j)}}, \dots, x_{i_{k_j}^{(j)}}\right)$ (ограничения В)

(3) $c_j(X_j) \Leftrightarrow x_{i_2^{(j)}} \in D_j^+\left(x_{i_1^{(j)}}\right) \parallel x_{i_3^{(j)}} \in D_j^+\left(x_{i_1^{(j)}}\right) \parallel \ldots \parallel x_{i_{k_j}^{(j)}} \in D_j^+\left(x_{i_1^{(j)}}\right)$ (ограничения C)

(4) $c_j(X_j) \Leftrightarrow$ для любого упорядочивания переменных ограничения X_j, представимого биекцией множеств индексов $\pi^{(j)}: \{1,2,\ldots,k_j\} \to \left\{i_1^{(j)}, i_2^{(j)}, \ldots, i_{k_j}^{(j)}\right\}$, имеет место $x_{\pi_1^{(j)}} \in D^+_{\pi^{(j)},1}$, $x_{\pi_2^{(j)}} \in D^+_{\pi^{(j)},2}\left(x_{\pi_1^{(j)}}\right)$, $x_{\pi_3^{(j)}} \in D^+_{\pi^{(j)},3}\left(x_{\pi_1^{(j)}}, x_{\pi_2^{(j)}}\right)$, …, $x_{\pi_{k_j}^{(j)}} \in D^+_{\pi^{(j)},k_j}\left(x_{\pi_1^{(j)}}, x_{\pi_2^{(j)}}, \ldots, x_{\pi_{k_j-1}^{(j)}}\right)$ (ограничения D)

(5) $c_j(X_j) \Leftrightarrow x_{i_1^{(j)}} \in D_j^-\left(x_{i_2^{(j)}}, \ldots, x_{i_{k_j}^{(j)}}\right)$ (ограничения E)

где $D_j^+ \in Z$, $D^+_{\pi^{(j)},k} \in Z$ — подмножества целых чисел, содержащие открытые положительные интервалы, а $D_j^- \in Z$ — подмножества целых чисел, содержащие открытые отрицательные интервалы.

Естественным визуальным представлением задачи GCPSP является двудольный направленный граф $G\langle X, C, R\rangle$, где X — множество вершин, ассоциированных с переменными задачи, C — множество вершин, ассоциированных с ограничениями задачи, и R — множество ребер графа, соединяющих вершины ограничений с соответствующими вершинами переменных следующим образом. Если переменная задачи x_i участвует в ограничении c_j в качестве независимой переменной, то ребро является входящим в вершину ограничения (в применяемой нотации $x_i \to c_j$). Если переменная задачи x_i является зависимой переменной ограничения c_j, то ребро входит в вершину переменной ($c_j \to x_i$). Если ограничение c_j предусматривает правила разрешения относительно каждой своей переменной x_i, то ребра между соответствующими вершинами будем представлять как двунаправленные и записывать $x_i \leftrightarrow c_j$. Будем также говорить, что переменная x_{i_1} прямо предшествует переменной x_{i_2} или $x_{i_1} \to x_{i_2}$, если существует такое ограничение задачи c_j, что $x_{i_1} \to c_j \to x_{i_2}$.

На рис. 1 представлен пример графа задачи GCPSP, на котором темными кружками отображены переменные задачи, а большими светлыми кругами — ограничения. Однонаправленные и двунаправленные ребра указывают характер зависимостей между переменными, который связан со способами разрешения ограничений.

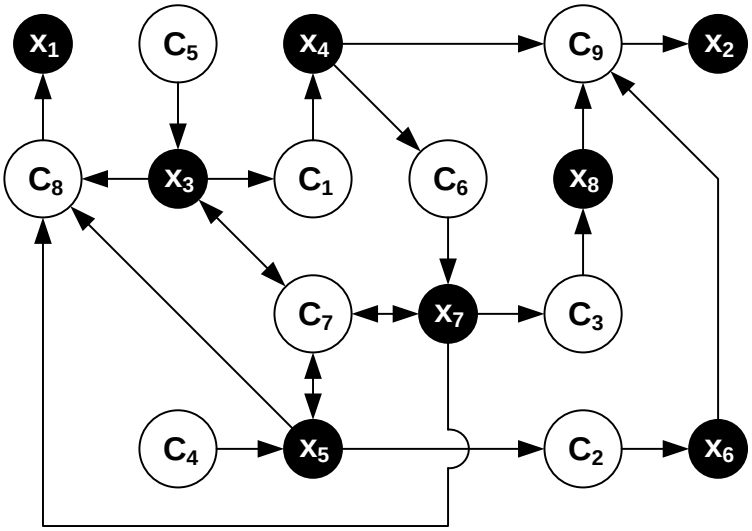

Рис. 1. Пример графа задачи GCPSP, отображающего взаимосвязи между переменными и ограничениями.

В предположении, что граф задачи ацикличен, план разрешения системы ограничений представляется в виде перестановки переменных $\pi: \{1,2,\dots,n\} \to \{1,2,\dots,n\}$, в соответствии с которой они могут быть последовательно выражены друг через друга или уточнена область их допустимых значений в соответствии с наложенными ограничениями. Циклом в данном случае называется упорядоченный набор пар вершин $x_{i_1} \to c_{j_1} \to x_{i_2} \to c_{j_2}, \dots, x_{i_k} \to c_{j_k} \to x_{i_1} \subseteq G\langle X, C, R\rangle$, такой, что вершины-переменные и вершины-ограничения соединены последовательно направленными ребрами от вершины x_{i_l} к вершине c_{j_l} для всех $l = 1,..,k$, от вершины c_{j_l} к вершине $x_{i_{l+1}}$ для всех $l = 1,..,k-1$, и от вершины c_{j_k} к вершине x_{i_1}. Заметим, что двунаправленные ребра не включаются в цикл. План разрешения системы ограничений визуально будем отображать на графе задачи, приписывая индексы упорядочивания переменных соответствующим вершинам.

Сформулируем достаточные условия разрешимости системы ограничений с приоритетами и существования решения для связанной с ней задачи обобщённого проектного планирования GCPSP.

Теорема 1

Пусть $\langle D, X, C, P \rangle$ — обобщённая задача GCPSP. Решение задачи всегда существует, если

(1) целевая функция задачи $f(X)$ монотонно неубывающая по каждой переменной;
(2) граф задачи $G\langle X, C, R \rangle$ ацикличен;
(3) ограничения (A) имеют наивысший приоритет, а приоритеты ограничений (B), (C) и (D) выше приоритета ограничений (E);
(4) ни одна переменная задачи не ассоциирована с более чем одним ограничением (A) в качестве зависимой переменной;
(5) каждая переменная задачи участвует, по крайней мере, в одном ограничении (A), (B), (C) или (D) в качестве зависимой переменной.

Доказательство

Прежде всего, покажем, что система ограничений задачи разрешима и множество допустимых значений для переменных задачи не пусто. Допустимость значений будем интерпретировать в терминах локальной согласованности переменных в соответствии с назначенными приоритетами ограничений. В предположении ацикличности графа задачи существует упорядочивание переменных, при котором допустимые значения или область допустимых значений для переменных с большими индексами выражаются через соответствующие значения переменных с меньшими индексами. Покажем, что при обходе переменных по возрастанию индексов всегда существует непустое множество допустимых значений переменной, удовлетворяющее ограничения с входящими ребрами в вершину-переменную. Рассмотрим следующие взаимоисключающие случаи для определения области допустимых значений переменной:

- У вершины-переменной отсутствуют входящие ребра. В этом случае область допустимых значений переменной — все множество целых чисел.
- Имеется входящее ребро от унарного ограничения (*A*). В этом случае переменная принимает единственное допустимое значение в соответствии с явной функцией разрешения ограничения, независимо от других наложенных ограничений (*B*), (*C*) и (*D*), которые имеют меньший приоритет. Наличие нескольких ограничений (*A*) условиями утверждения исключаются.
- Имеются входящие ребра от ограничений (*B*) и (*C*) и отсутствует входящее ребро от ограничений (*A*). Независимо от назначенных приоритетов для переменной задачи всегда существует положительный полуинтервал, удовлетворяющий данным ограничениям. Ограничения (*D*) имеют меньший приоритет и поэтому либо нарушаются, либо удовлетворяются путем уточнения допустимой области переменной в виде интервала или точки.
- Входящие ребра только от ограничений (*D*) не могут существовать в силу принятых допущений.

Таким образом, множество локально согласованных решений системы ограничений с приоритетами не пусто, причем они ограничены снизу. В силу неубывания целевой функции задачи по каждой из переменной ее минимум достигается на данном множестве и задача обобщённого проектного планирования GCPSP имеет решение. ■

Следующее утверждение определяет достаточные условия разрешимости задач проектного планирования GCPSP в случае, когда приоритеты назначаются ограничениям индивидуально независимо от их общего вида.

Теорема 2

Пусть $\langle D, X, C, P \rangle$ — обобщённая задача GCPSP. Решение задачи всегда существует, если

(1) целевая функция задачи $f(X)$ по каждой переменной не убывает на некотором положительном полуинтервале и не возрастает на некотором отрицательном полуинтервале;

(2) граф задачи $G\langle X, C, R\rangle$ ацикличен;

(3) для каждой переменной задачи и ограничений, в которых она участвует как зависимая переменная, наивысший приоритет имеют либо одно ограничение (A), либо несколько ограничений (B), (C) и (D), либо несколько ограничений (E).

Доказательство

Система ограничений задачи разрешима в смысле локальной согласованности переменных задачи. В самом деле, для каждой зависимой переменной всегда могут быть разрешены ограничения с наивысшим приоритетом. В случае единственного ограничения (A) это очевидно. В случае нескольких ограничений (B), (C) и (D) всегда существует положительный полуинтервал, которому удовлетворяют данные ограничения. В случае нескольких ограничений (E) существует отрицательный полуинтервал, которому удовлетворяют ограничения данного вида. Выполнимость ограничений с низкими приоритетами не имеет никакого значения для множества допустимых решений. В силу свойств монотонности целевой функции по каждой переменной на данном множестве существует, по крайней мере, одно оптимальное решение. ■

2. 4. Алгоритм приближенного решения GCPSP-задач

Опишем алгоритм приближенного решения задач проектного планирования в постановке GCPSP. Его можно рассматривать в качестве обобщения известной схемы последовательной диспетчеризации работ.

На каждом шаге алгоритма $k = 1,2, \dots, n$ определяется одна из переменных задачи x_i, $i \in I(X)$. При этом выбор очередной переменной подчиняется следующей общей схеме. Пусть $I_{PASSIVE} \subseteq I(X)$ — множество индексов переменных, значения которых определены к началу шага k, а $I_{ACTIVE} \subseteq I(X)$ —

множество индексов неопределенных переменных, все прямые предшественники которых содержатся в $I_{PASSIVE}$ или вовсе не имеют предшественников. Таким образом, имеют место следующие соотношения между рабочими множествами: $I_{PASSIVE} \cap I_{ACTIVE} = \emptyset$ и если $i \in I_{ACTIVE}$, то либо не существует $j \in I(X)$ такого что $x_j \to x_i$, либо для каждого $j \in I(X)$ такого что $x_j \to x_i$, имеет место $j \in I_{PASSIVE}$.

Тем самым, множество $I(X)\backslash(I_{PASSIVE} \cup I_{ACTIVE})$ включает в себя индексы тех переменных, которые не участвовали в анализе до текущего шага и должны быть определены на следующих шагах алгоритма.

На начальном шаге инициализируются вспомогательные множества $I_{PASSIVE} = \emptyset$ и $I_{ACTIVE} = \{i \in I(X) \mid \nexists\, j \in I(X), x_j \to x_i\}$. В силу условий описанного выше утверждения граф задачи ацикличен, множество переменных, не имеющих предшественников, не пусто и алгоритм корректно стартует. На каждом шаге алгоритма отбирается активная переменная x_i, $i \in I_{ACTIVE}$, определяется ее значение и корректируются рабочие множества индексов таким образом, что активная переменная перемещается из I_{ACTIVE} в $I_{PASSIVE}$, а множество I_{ACTIVE} пополняется новыми переменными, все предшественники которых уже находятся в $I_{PASSIVE}$. Поскольку коррекция рабочих множеств осуществляется на каждом шаге алгоритма для поиска пополняемых переменных достаточно проанализировать только предшественников последователей переменной x_i. Таким образом, при переходе к следующему шагу рабочие множества корректируются следующим образом:

$$I_{ACTIVE} = I_{ACTIVE}\backslash\{i\},$$

$$I_{PASSIVE} = I_{PASSIVE} \cup \{i\},$$

$$I_{ACTIVE} = I_{ACTIVE} \cup \{k \in I\backslash(I_{PASSIVE} \cup I_{ACTIVE}) \mid x_i \to x_k, \forall\, x_{i'} \to x_k, i' \in I_{PASSIVE}\}.$$

Выборка активной переменной из множества I_{ACTIVE} выполняется на основе эвристических правил вида $i' H\, i''$, где $i', i'' \in I_{ACTIVE}$, определяющих линейный порядок на множестве переменных и позволяющих на текущем шаге выбрать переменную, наиболее предпочтительную для достижения оптимума целевой функцией. Иногда правила задают частичный порядок на множестве переменных.

В подобных случаях можно определить иерархическую стратегию $H = \{H_l\}$, $l = 1,2,\dots,L$, заключающуюся в последовательном применении элементарных правил H_l в ожидании того, что одно из них позволит вынести окончательный вердикт о предпочтении одной переменной над другой. Если ни одно из правил не позволяет установить это, то приоритетной может считаться та переменная, которая имеет наименьший индекс. Очевидно, что порядок применения элементарных правил в рамках иерархической стратегии может влиять на упорядочивание переменных и качество приближенного решения задачи. Алгоритмом допускается широкий набор эвристических правил, применяемым в задачах RCPSP [170]. Ниже приводится математически эквивалентная интерпретация некоторых популярных правил, сохраняя их оригинальные названия:

- LIS (Least Immediate Successors): выбрать переменную с наименьшим количеством непосредственных последователей;
- MIS (Most Immediate Successors): выбрать переменную с наибольшим количеством непосредственных последователей;
- MTS (Most Total Successors): выбрать переменную с наибольшим количеством всех последователей;
- LTS (Least Total Successors): выбрать переменную с наименьшим количеством всех последователей;
- LSC (Longest Successors Chain): выбрать переменную с наибольшим числом переменных в цепочках последователей;
- SSC (Shortest Successors Chain) выбрать переменную с наименьшим числом переменных в цепочках последователей;
- SPT (Shortest Process Time): выбрать переменную с наименьшей разностью $x_j - x_i$ среди всех пар $x_j \to x_i$, связанных ограничениями вида (A) и (B). Здесь x_i — активная переменная, значение которой определяется в предположении ее отбора, а x_j — ее непосредственный последователь, значение которого рассчитывается на основе правила разрешения

соответствующего ограничения и выбора минимально допустимого значения;

- LPT (Longest Process Time): выбрать переменную с наибольшей разностью $x_j - x_i$ среди всех пар $x_j \to x_i$, связанных ограничениями вида (A) и (B).

Существуют и более сложные правила, применение которых предполагает решение вспомогательных задач, связанных с локальной оптимизацией целевой функции. Важно, чтобы применяемые эвристики были релевантны целевой функции.

Значение для выбранной активной переменной определяется, исходя из требования минимизации целевой функции при условии удовлетворения максимального количества ассоциированных с ней ограничений. В случаях, когда система ограничений является переопределенной, ограничения разрешаются последовательно в соответствии с заданными приоритетами. В предположениях описанного выше утверждения всегда существует способ обеспечить локальную согласованность решения, при которой в первую очередь разрешаются ограничения с высокими приоритетами, а затем, если возможно, ограничения с низкими приоритетами.

В самом деле, после построения множеств $I_{PASSIVE}$ и I_{ACTIVE} и выбора активной переменной x_i, $i \in I_{ACTIVE}$ все независимые переменные ассоциированных с ней ограничений (A), (B), (C), (D) и (E) находятся в множестве $I_{PASSIVE}$ и поэтому их значения уже определены и могут быть использованы для расчета допустимых значений x_i.

В случае наложенного ограничения вида (A) переменная принимает единственное значение в соответствии с правилом разрешения ограничения $x_i = g_j\left(x_{i_2^{(j)}}, x_{i_3^{(j)}}, \dots, x_{i_{k_j}^{(j)}}\right)$, где $\left\{i_2^{(j)}, i_3^{(j)}, \dots, i_{k_j}^{(j)}\right\} \subseteq I_{PASSIVE}$, независимо от других наложенных ограничений.

В случае ограничений вида (B), (C), (D) для переменной x_i вначале определяется допустимое множество значений как пересечение множеств

интервалов $D_j^+\left(x_{i_2^{(j)}}, x_{i_3^{(j)}}, \ldots, x_{i_{k_j}^{(j)}}\right)$ для всех j таких, что $c_j \to x_i$. Пересечение не пусто, поскольку множества включают в себя положительные полуинтервалы. Значение переменной выбирается, исходя из требования минимизации целевой функции по данной переменной. В случае неубывания целевой функции точка минимума всегда существует, а если она не единственна, то выбирается наименьшее значение. Переменные задачи обычно семантически связаны с датами исполнения и сроками, которые предпочтительно сократить даже при достижении целевой функцией минимума. Таким образом, значение выбранной переменной в этом случае определяется следующим образом:

$$x_i^* = \operatorname{argmin} f(x_i),$$

$$x_i \in \bigcap_{c_j \to x_i} D_j^+\left(x_{i_2^{(j)}}, x_{i_3^{(j)}}, \ldots, x_{i_{k_j}^{(j)}}\right)$$

Анализ ограничений (C) и (D) имеет свои особенности. Чтобы удовлетворить ограничение вида (C) достаточно выполнить соответствующее условие для одной зависимой переменной. Действительно, попытка распространить условие $x_{i^{(j)}} \in D_j^+\left(x_{i_1^{(j)}}\right)$ на все зависимые переменные может оказаться обременительной для поиска качественного приближенного решения. Поэтому алгоритм предусматривает иную схему. Ограничение разрешается только для самой последней обрабатываемой переменной ограничения и только в том случае, если оно не было автоматически удовлетворено в результате определения значений предыдущих переменных. С этой целью для каждого ограничения вида (C) рассчитывается и хранится статус его выполнения, а также ведется подсчет числа обработанных ограничений. Такая схема обеспечивает разумную стратегию удовлетворения ограничений и минимизации целевой функции в случаях, когда переменные связаны между собой множественными ограничениями.

Для активной переменной все ассоциированные с ней ограничения (D) удовлетворяются в рамках общей схемы, поскольку они могут быть разрешены

относительно любой переменной независимо от переменных множества $I(X)\backslash I_{PASSIVE}$, значения которых на текущем шаге алгоритма еще не определены.

Ниже приводится псевдокод описанного выше обобщённого алгоритма.

```
Begin
  X := ⊥;
  I_PASSIVE := ∅;
  I_ACTIVE := {i | i ∈ I, ∄ i' ∈ I, x_i' → x_i};
  While |I_ACTIVE| > 0 do
    i := selectVariable(I_ACTIVE);
    x_i := computeVariable(i, X);
    I_PASSIVE := updatePassive(I_PASSIVE, i);
    I_ACTIVE := updateActive(I_ACTIVE, i);
  End while;
End;

Function selectVariable(I_ACTIVE)
Begin
  i := I_ACTIVE[1];
  For k := 2 to |I_ACTIVE| do
    If I_ACTIVE[k] H i then
      i := I_ACTIVE[k];
    End if;
  End for;
  Return i;
End;

Function computeVariable(i, X)
Begin
  D* := Z;
  J := {j | c_j → x_i | c_j ↔ x_i};
  J := sort(J, P);
  For j := 1 to |J| do
    D := resolve(c_j, X_j, i);
    If j = 1 then
      If D = ∅ then
        «Ошибка в постановке задачи или решения не
        существует!»
        Stop;
      Else
        D* := D;
```

```
      End if;
    Else
      If D* ∩ D ≠ ∅ then
        D* := D* ∩ D;
      Else
        «Ограничение не может быть удовлетворено!»
      End if;
    End if;
  End for;
  Return argmin_{x_i ∈ D*}{f(x_i)};
End;

Function updatePassive(I_PASSIVE, i)
Begin
  Return I_PASSIVE ∪ {i};
End;

Function updateActive(I_ACTIVE, i)
Begin
  Return
  (I_ACTIVE \ {i}) ∪ {k ∈ I \ (I_PASSIVE ∪ I_ACTIVE) | x_i → x_k, ∀ x_i' → x_k, i' ∈ I_PASSIVE};
End;
```

Вспомогательная функция $sort(J, P)$ сортирует ограничения J по убыванию приоритетов P. Перед вызовом функции множество J формируется из ограничений, в которых x_i участвует в виде зависимой переменной. Функция $resolve(c_j, X_j, i)$ разрешает ограничение c_j относительно i переменной при фиксированных значениях остальных переменных из множества X_j. Возвращаемый результат — множество допустимых значений переменной D^*. В силу сделанных допущений относительно вида ограничений, функция допускает обобщённую реализацию. Из множества D^* переменной x_i присваивается одно из значений, при котором достигается минимум целевой функции по данной переменной.

В наихудшем случае описанный алгоритм имеет квадратичную вычислительную сложность $O(n^2)$ от количества переменных задачи n, поскольку на каждом шаге алгоритма из множества активных переменных, мощность которого ограничена n, необходимо выбирать наиболее предпочтительную

переменную путем последовательного применения эвристических правил. Данная полиномиальная оценка приемлема для решения прикладных задач высокой размерности.

Приведённый алгоритм может быть легко трансформирован для поиска точного решения. Для этого достаточно подменить функцию $selectVariable$ на аналогичную, в которой выбор приоритетной переменной из множества активных I_{ACTIVE} осуществляется на основе перебора вариантов с отсечением по верхней оценке целевой функции задачи U_f. Псевдокод функции приводится ниже.

```
Function selectVariable(I_ACTIVE, X, U_f)
Begin
  If U_f ≠ ⊥ and f(X) > U_f then
    Return ⊥;
  Else
    If |I_ACTIVE| = 1 then
      Return I_ACTIVE[1];
    Else
      i_BEST := ⊥;
      f_BEST := ⊥;
      For i := 1 to |I_ACTIVE| do
        X' := X;
        x'_i := computeVariable(i, X');
        I'_ACTIVE := updateActive(I_ACTIVE, i);
        While |I'_ACTIVE| > 0 do
          i' := selectVariable(I'_ACTIVE, X', f_BEST);
          If i' = ⊥ then
            Break while;
            Next i;
          Else
            x'_i' := computeVariable(i', X');
            I'_ACTIVE := updateActive(I'_ACTIVE, i');
          End if;
        End while;
        If i_BEST = ⊥ then
          i_BEST := i;
          f_BEST := f(X');
        Else
          f := f(X');
```

```
          If f < f_BEST then
            i_BEST := i';
            f_BEST := f;
          End if;
        End if;
      End for;
      Return i_BEST;
    End if;
  End if;
End;
```

Таким образом, точное решение находится путём перебора всех последовательностей назначений переменных, при которых достигается минимальное значение целевой функции на множестве локально согласованных решений. При этом алгоритм реализует метод «ветвей и границ», при котором из анализа исключаются последовательности, заведомо приводящие к неоптимальным решениям.

Теорема 3

Класс задач RCPSP принадлежат классу задач проектного планирования GCPSP. В предположениях классической постановки RCPSP существуют решения эквивалентных задач GCPSP. Обобщённый алгоритм проектного планирования сводится к алгоритму последовательной диспетчеризации работ в случае задач RCPSP.

Доказательство

Покажем, что любая задача RCPSP удовлетворяет условиям обобщённой постановки GCPSP. Выберем в качестве неизвестных переменных задачи целочисленные даты начала работ $x = (x_1, x_2, \dots x_n)$. Очевидно, что функция минимизации проектного времени $\min(\max_{i=1,\dots N}(x_i + d_i))$ удовлетворяет условиям задачи GCPSP. Условия начала проекта $x_i \geq t_0, i = 1, \dots, N$ выражаются ограничениями вида (B), в которых отсутствуют независимые переменные, а область допустимых значений зависимых переменных определяется положительными полуинтервалами. Отношения предшествования работ $x_{i_{(l)}} +$

$d_{i_{(l)}} \leq x_{i^{(l)}}, l = 1,2, \dots, L$ также могут быть представлены ограничениями вида (B), если в качестве зависимых переменных использовать даты начала работ-последователей. Наконец, ресурсные условия

$$\sum_{i \in I(k)} \theta(t, x_i, d_i)\, r_{ik} \leq R_k, k = 1,2, \dots, K$$

могут быть отнесены к ограничениям (D), поскольку последовательно разрешимы относительно каждой переменной при фиксированных значениях других.

В предположениях классической постановки задач RCPSP для эквивалентных задач GCPSP также существует решение. Для доказательства воспользуемся достаточными условиями утверждения 1. Во-первых, целевая функция задачи RCPSP монотонно не убывает по каждой переменной. Во-вторых, граф задачи GCPSP ацикличен при условии, что отношения предшествования не образуют циклов. В-третьих, порождаемые условиями RCPSP ограничения вида (B) всегда разрешимы. Порождаемые ограничения вида (D) разрешимы, если уровни потребления ресурсов индивидуальными работами не превышают их доступное количество. Наконец, поскольку в эквивалентной задаче отсутствуют ограничения (A) и (E), а условия старта проекта распространяются на все переменные задачи, то при любом назначении приоритетов на индивидуальные ограничения (B) и (D) решение задачи GCPSP существует согласно утверждению 1.

Для доказательства сводимости обобщённого алгоритма в случае задачи RCPSP достаточно адресоваться к описанию алгоритма последовательной диспетчеризации [10], а также проинтерпретировать работы в терминах переменных, а отношения предшествования — в терминах ограничений и соответствующих правил разрешения задачи GCPSP. При использовании упомянутых выше эвристик LIS, MIS, MTS, LTS, LSC, SSC, связанных с количеством непосредственных или общих последователей-переменных, предпочтение будет отдаваться тем же активным работам, что и в алгоритме

последовательной диспетчеризации с аналогичными эвристиками, но выраженными в терминах предшествования работ. ■

Вместе с тем, область практического применения рассмотренной обобщённой постановки GCPSP существенно шире, поскольку охватываются прикладные задачи проектного планирования в расширенных постановках с учетом альтернативных критериев составления расписаний, сложных моделей исполнения работ, многопараметрических взаимосвязей, особенностей привлечения ресурсов, применения календарных графиков, финансового и логистического обеспечения проектных работ. Для краткости остановимся на некоторых примерах, характерных для расширенных постановок.

При планировании обычно применяются составные работы, которые обеспечивают иерархическую многоуровневую структуризацию проектного плана. Даты начала составных работ x_i и их продолжительности d_i связаны с параметрами дочерних работ $j \prec i$ следующим образом:

$$x_i = \min_{j=1,\dots,N \mid j \prec i} (x_j)$$

$$d_i = \max_{j=1,\dots,N \mid j \prec i} (x_j + d_j - x_i)$$

Данные условия представимы ограничениями (A) в постановке GCPSP и порождают ацикличный подграф задачи, допускающий последовательное вычисление значений переменных, начиная с параметров дочерних работ и заканчивая параметрами составных работ. Вычисления естественным образом обобщаются на случай многоуровневых составных работ, для которых обход начинается с простых работ нижнего уровня и распространяются на работы верхних уровней.

Для представления вех в проектном плане можно использовать простые работы с нулевой продолжительностью. Более содержательным является моделирование «коротких гамаков» и «длинных гамаков», которые можно описать в постановке GCPSP с помощью соответствующих ограничений (A).

Пусть дата начала «короткого гамака» x_i и его продолжительность d_i, тогда

$$x_i = \max_{j=1,\dots,N \mid x_j \to x_i} (x_j + d_j)$$

$$d_i = \min_{j=1,\dots,N \mid x_i \to x_j} (x_j - x_i)$$

Для «длинного гамака» соотношения приобретают вид

$$x_i = \min_{j=1,\dots,N \mid x_j \to x_i} (x_j + d_j)$$

$$d_i = \max_{j=1,\dots,N \mid x_i \to x_j} (x_j - x_i)$$

При условии, что взаимосвязи «гамаков» не образуют циклов, данные ограничения в постановке GCPSP также приводят к ацикличному графу и могут быть последовательно разрешены.

Другим примером расширенной постановки задач проектного планирования могут служить отношения предшествования типа «начало-начало», «начало-окончание», «окончание-начало» и «окончание-окончание», для которых могут быть установлены величины задержки. Пусть $i_{(l)}$ — индекс работы-предшественника, участвующего в связи l, $i^{(l)}$ — индекс работы-последователя, участвующего в связи l, и Δt_l — задержка связи, для которой допустимы и отрицательные значения. Тогда отношения предшествования выражаются следующими ограничениями в соответствии с перечисленными выше типами:

$$x_{i^{(l)}} \geq x_{i_{(l)}} + \Delta t_l$$

$$x_{i^{(l)}} \geq x_{i_{(l)}} - d_{i^{(l)}} + \Delta t_l$$

$$x_{i^{(l)}} \geq x_{i_{(l)}} + d_{i_{(l)}} + \Delta t_l$$

$$x_{i^{(l)}} \geq x_{i_{(l)}} + d_{i_{(l)}} - d_{i^{(l)}} + \Delta t_l$$

В постановке GCPSP данные условия выражаются ограничениями вида (B).

Другим важным аспектом является учет рабочих календарей, в соответствии с которыми исполняются работы и привлекаются ресурсы. Продолжительности, трудозатраты, задержки обычно выражаются в единицах рабочего времени, которые должны быть пересчитаны в календарные даты. Например, если x_i — дата начала простой работы и d_i — ее продолжительность, то дата завершения определяется не выражением $x_i + d_i$, как в приведенных выше формулах, а

функцией календаря как $g(x_i, d_i)$. Примечательно, что общий вид ограничений в рамках обобщённой постановки GCPSP при этом не меняется.

Характерной особенностью прикладных постановок является также задание алгебраических условий для дат начала и завершения работ. Данные условия типа «начать не раньше», «завершить не раньше» или «начать не позже», «завершить не позже» выражаются в постановке GCPSP ограничениями вида (B) и (E) соответственно:

$$x_i \geq t_i,$$
$$x_i \geq t_i - d_i,$$
$$x_i \leq t_i,$$
$$x_i \leq t_i - d_i,$$

где t_i — соответствующие директивные сроки. Согласованность ограничений обычно не контролируется пользователем, и они могут оказаться переопределенными даже в случае одной проектной работы. Для разрешения подобных ситуаций ограничениям могут быть приписаны приоритеты, а выполнимость условий проинтерпретирована в терминах локальной согласованности. Принципиально, что обобщённая постановка GCPSP предусматривает и такую возможность.

Рассмотрим более интересный случай, когда временные условия задаются пользователем для дат начала и завершения составных работ. Поскольку сами параметры составных работ являются зависимыми переменными от соответствующих параметров дочерних работ, непосредственный анализ таких ограничений не возможен. Однако подобные условия могут быть переопределены эквивалентным образом. Например, если для составной работы i задано условие «начать не раньше» в виде $x_i \geq t_i$, то оно может быть переопределено для всех дочерних работ $j \prec i$ как $x_j \geq t_i$, поскольку $x_i = \min_{j=1,\ldots,N \mid j \prec i}(x_j)$. Тем самым условие в постановке GCPSP представимо ограничениями вида (B). Если для составной работы i задано условие «завершить не раньше» в виде $x_i \geq t_i - d_i$, то оно переопределяется для дочерних работ $j \prec i$ как $\bigvee_{j \prec i} x_j \geq t_i -$

d_j, поскольку $d_i = \max_{j=1,\dots,N \mid j \prec i}(x_j + d_j - x_i)$. Данное условие в постановке GCPSP представляется ограничением вида (C).

Наконец, в классической постановке RCPSP рассматриваются только возобновимые ресурсы с постоянным профилем использования. В расширенных постановках обычно участвуют модели возобновимых и невозобновимых ресурсов с переменными профилями использования и доступности. Ресурсные ограничения в подобных случаях могут быть описаны следующим образом:

$$\forall\, t \geq t_0 \text{ имеет место } \sum_{i \in I(t,r)} q_{i,r}(t, x_i, d_i) \leq Q_r(t), r = 1,2, \dots, R,$$

где монотонно неубывающая ограниченная функция $Q_r(t)$ определяет доступное количество ресурса r на момент времени t в предположении, что данный ресурс не расходовался. Ограниченная функция $q_{i,r}(t, x_i, d_i)$ задает профиль использования ресурса r задачей i на момент времени t, а множество $I(t,r)$ определяет индексы работ, использовавших или использующих ресурс r на момент времени t.

Пусть для возобновимых ресурсов $q_{i,r}(t, x_i, d_i) = 0$ при $t < x_i$ или $t \geq x_i + d_i$ и $q_{i,r} = \max_{x_i \leq t < x_i + d_i} \{q_{i,r}(t, x_i, d_i)\}$, а для невозобновимых ресурсов имеет место $q_{i,r}(t, x_i, d_i) = 0$ при $t < x_i$ и $q_{i,r}(t, x_i, d_i) = q_{i,r}$ при $t \geq x_i + d_i$, где $q_{i,r} = \max_{t \geq x_i} \{q_{i,r}(t, x_i, d_i)\}$. Тогда можно показать, что условие в постановке GCPSP представляется ограничением вида (D), если для невозобновимых ресурсов выполняется $\sum q_{i,r} \leq Q_r$, а для возобновимых ресурсов имеет место $q_{i,r} \leq Q_r$, где $Q_r = \max_{t \geq t_0} \{Q_r(t)\}$.

Рассмотренные примеры демонстрируют общность предложенной постановки GCPSP для разнообразных прикладных задач проектного планирования.

Глава 3. Объектно-ориентированная среда для разработки приложений теории расписаний

Многообразие существующих математических моделей и вычислительных методов, а также перманентное появление новых ставит перед программистами довольно острую проблему эволюционной разработки серий приложений на единой методологической, программной и инструментальной основе. В частности, подобная проблема возникает при создании перспективных систем календарно-сетевого планирования и управления индустриальными проектами, в которых задачи составления расписаний решаются в обобщённой постановке с учетом множества факторов, влияющих на ход выполнения проектных работ. В таких постановках учитываются не только типовые временные условия, отношения предшествования между работами, ресурсные ограничения, но и специфические требования пространственно-временной согласованности проектных работ, их финансового и логистического обеспечения. Данные требования существенны для масштабных индустриальных программ, в которых риски технологических и организационных ошибок чрезвычайно высоки, а сроки и бюджеты жестко ограничены. Примерами специфических требований могут служить условия привлечения инвестиционных средств, ограничения по поставкам материалов, правила размещения и использования оборудования, особенности монтажа элементов конструкций возводимого сооружения, условия резервирование рабочих зон при организации проектных работ.

Использование математических библиотек для решения подобных задач, как правило, оказывается невозможным из-за принципиальных отличий в их постановках или крайне неэффективным в силу зависимости вычислительной сложности составления расписания от частных условий.

Стратегия разработки программ составления расписаний заново для каждого нового типа приложения также является неприемлемой в силу сложности современных математических моделей и вычислительных методов. Данные методы должны учитывать большое количество факторов и использовать развитые системы эвристик для поиска приемлемых приближенных решений в тех случаях, когда задача имеет высокую размерность. Разработка и программная реализация подобных методов часто оказывается предметом интенсивных научных исследований. Адаптация унаследованных открытых кодов, изначально непредназначенных для подобных целей и не предусматривающих возможности для их развития, также малоэффективна даже для написания программ близкой функциональности.

Решением данной проблемы может стать создание единой инструментальной среды для программной реализации моделей, методов и приложений теории расписаний. Подобная среда должна предоставлять развитые средства для разработки новых программ на основе ранее реализованных модулей. При этом возможности развития, адаптации и конфигурации модулей должны обеспечивать построение эффективных программ составления расписаний, релевантных условиям и сложности решаемых прикладных задач.

Ранее предпринимались попытки организации подобных сред в виде расширяемых математических библиотек. В качестве одного из значимых результатов следует упомянуть библиотеку PSPLIB [166], в которой Колиш и Шпрехер реализовали несколько методов проектного планирования. Библиотека претерпела определенное развитие [171, 172, 173, 174], однако в настоящее время она преимущественно используется для тестирования других аналогичных программ и оценки производительности на специально подготовленных наборах контрольных примеров (benchmarks). Библиотека имеет принципиальные ограничения для решения индустриально значимых задач высокой размерности, а ее архитектура плохо приспособлена для реализации новых моделей и методов.

Более интересной в этом отношении является библиотека проектного планирования LibRCPS, разработанная Лемменем [175]. В ней предусмотрены

интерфейсы для задания входных данных, а также имеется возможность специфицировать некоторые алгоритмические детали поиска решения, например, выбрать тип целевой функции и применяемую эвристику. К сожалению, значительная часть планируемых возможностей и функций библиотеки осталась нереализованной. Несмотря на открытые исходные коды, библиотека не получила дальнейшего развития, а сообщения об опыте ее применения крайне скудны.

Более успешными в практическом отношении оказались программные системы календарно-сетевого планирования и управления проектами. Обычно они обеспечивают автономную работу пользователя на изолированном компьютере или групповую работу в корпоративной сети. В качестве популярных программных решений следует указать Oracle Primavera, MS Project, Synchro, Spider Project, Gemini, Merlin, Zoho Projects, ManagePro [176]. Ряд систем конфигурируется в виде универсальных Интернет-сервисов и WEB-клиентов к ним. К подобным решениям относятся Smartsheet, GanttPro, Asana, Acunote, Teamweek, Bitrix24, Jira, ISETIA [176]. Несмотря на наличие программных интерфейсов доступа к данным и возможность локального или удаленного вызова функций планирования, перечисленные системы обладают существенным общим недостатком. Главным образом он связан с предопределенным характером реализованных алгоритмов и невозможностью их модификации для решения новых классов прикладных задач. Тем самым, данные системы не предоставляют инструментальных возможностей, необходимых для их дальнейшего функционального развития.

Вместе с тем, потребность в подобных инструментах велика, поскольку разрабатывается большое число специализированных систем, ориентированных на частные индустриальные приложения. Для примера приведем лишь некоторые их них, акцентируя внимание на разнообразие прикладных постановок.

Система PLANETS (PLanning Activities on NETworkS) [177, 178], разработанная Университетом Каталония (University of Catalonia) в Барселоне для испанской электрической компании, как инструмент календарного планирования перестройки и технического обслуживания электрической сети без нарушения

обслуживания потребителей. Система ATLAS [179, 180] осуществляет планирование производства гербицидов на заводе Monsanto в Антверпене. Система MOSES [181] была разработана компанией COSYTEC для производителя питания для животных в Великобритании. Система FORWARDC [182] представляет собой систему поддержки принятия решений (СППР) и используется на нефтеперегонных заводах в Европе при планировании поставок сырой нефти, ее обработки, смешивания и доставки потребителям. Xerox использовал систему планирования различных видов работ на копировальных машинах [183]. TAP-AI [184] — активная система планирования, предназначенная для ежедневного управления деятельностью авиалиний SAS. OPTISERVICE [185] — программный пакет для назначения персонала для всех заграничных теле- и радиостанций сети RFO с учетом ограничений по времени и условиям оплаты квалифицированных журналистов и технических работников. Система MOSAR [186], разработанная компаниями Cisi и COSYTEC для министерства юстиции Франции, назначает охранников тюрем по 200 тюрьмам Франции по чередующимся сменам. Система COBRA [187] позволяет разработать диаграммы рабочих планов машинистов поездов компании North Western Trains в Великобритании. Проект DAYSY Esprit (пакет SAS-Pilot) [184] осуществляет переназначение летных экипажей по полетам. Система краткосрочного планирования (The Short Term Planning (STP)) для компании Renault [188] решает задачу транспортировки автомобилей заказчикам с учетом множества ограничений. Средства финансового планирования применяются при утверждении бюджетов районов Москвы и их последующем контроле [189]. Планирование некоторых телекоммуникационных сетей мобильной связи осуществляется с помощью системы POPULAR [190]. Примечательно, что во многих системах используются технологии логического программирования в ограничениях, которые оказываются конкурентоспособными как по гибкости задания условий прикладных задач, так и по эффективности их решения. В частности, языки и системы логического программирования в ограничениях CHIP, 2LP, ILOG, ECLiPSe послужили математической основой для реализации

некоторых из перечисленных выше специализированных систем. Тем не менее, интерпретация задач обобщённого проектного планирования в терминах логического программирования невозможна и данные технологии могут применяться лишь в качестве элемента более общих подходов к планированию.

Важный шаг к систематизации и концептуализации задач проектного планирования был сделан в связи со становлением технологий информационного моделирования процессов строительства BIM (Building Information Modeling) [191, 192]. Появившиеся информационные стандарты и, в частности, модель IFC (ISO 16739:2013 Industry Foundation Classes) [193] позволяют специфицировать некоторые типовые условия задач календарно-сетевого планирования, используя программные интерфейсы доступа к данным или альтернативные способы их представления в файлах открытых форматов. Однако данные стандарты не регламентируют математические методы планирования и поэтому не могут служить полноценной основой для реализации программных приложений соответствующей функциональности.

Настоящая глава посвящена проблемам создания инструментальной среды для программной реализации моделей, методов и приложений теории расписаний в виде объектно-ориентированного каркаса или архитектурного шаблона (object-oriented framework). Объектно-ориентированные каркасы с успехом применяются при разработке сложных программных систем с расширяемым функционалом и при создании линеек программных продуктов в смежных предметных областях. Поэтому использование данного подхода для достижения декларируемых целей представляется вполне оправданным.

Требования, предъявляемые к объектно-ориентированному каркасу, а также общие принципы его построения обсуждаются в разделе 3. 1. Раздел 3. 2 посвящён организации классов прикладных данных для представления условий задач проектного планирования RCPSP в расширенных постановках. Классы математических объектов и вычислительных алгоритмов для редукции задач проектного планирования к задачам условной оптимизации и их решения описываются в разделе 3. 3. Методологические аспекты разработки программных

приложений теории расписаний на основе модулей каркаса рассматриваются в разделе 3. 4.

3. 1. Общие принципы и организация каркаса

3. 1. 1. Понятие объектно-ориентированного каркаса

Согласно наиболее распространённому определению [194], каркас (или фреймворк — от английского «framework») представляет собой программную платформу, определяющую структуру целевой программной системы и облегчающую разработку, сопровождение и объединение компонентов в ее составе. Целевые системы в этом случае состоят из самого каркаса, являющегося их неизменной частью, и модулей расширения, конфигурации которых могут гибко меняться для обеспечения требуемой функциональности системы и желаемых характеристик. Организация каркаса при этом должна предусматривать, так называемые, точки расширения (hot spots), благодаря которым модули с одними и теми же интерфейсами могут применяться в качестве альтернативных реализаций основных функций системы. Точки расширения во многом задают правила конфигурирования и направления возможной функциональной эволюции целевой системы.

Каркасный подход естественным образом воплощается в рамках парадигмы объектно-ориентированного программирования (ООП). Поскольку ООП предполагает систематизацию и концептуализацию предметной области, относительно просто определяются элементы и точки расширения каркаса. Ключевые понятия предметной области, допускающие специализацию, оформляются в виде абстрактных классов каркаса с предопределенным интерфейсом. Они задают потенциальные точки расширения каркаса. Непосредственная реализация методов интерфейса осуществляется в наследуемых классах, чем обеспечивается полиморфизм включения и возможность конфигурирования приложений из специализированных элементов каркаса. Примечательно, что значительная часть методов может быть

имплементирована на уровне абстрактных классов, тем самым, избавляя разработчиков от необходимости повторной реализации общих и, часто нетривиальных, механизмов взаимодействия классов каркаса. В этом случае разработчики могут сосредоточиться на особенностях реализации методов для конкретных типов классов с учетом их специфических свойств и поведения. Естественно, чтобы достичь подобных преимуществ, требуется провести тщательный объектный анализ предметной области и спроектировать каркас рациональным образом, обеспечивающим его последующее многократное использование при относительно невысоких затратах на доработку целевых приложений.

Каркасный подход к построению приложений тесто связан с общими методологиями объектно-ориентированного программирования и, в частности, с теорией объектно-компонентного моделирования, предложенной и развитой Е.М. Лаврищевой [195, 196], а также методом построения систем с расширяемым функционалом, описанным М.М. Горбуновым-Посадовым [197]. Примечательно, что организация и функциональное назначение классов каркаса, как правило, соответствует концепции шаблонов проектирования (паттернов) [198, 199].

Во многом становление каркасного подхода было предопределено необходимостью разработки графических интерфейсов пользователя (GUI), которые имели тенденцию к выделению стандартной структуры приложения и типовых графических элементов. Популярные в настоящее время графические библиотеки MFC, Qt, GNOME, KDE в полной мере служат примерами успешного применения каркасного подхода.

Принципиальным отличием каркаса от библиотеки программ является то, что он не просто предоставляет наборы отдельных, часто несвязанных между собой функций, но и во многом предопределяет архитектуру всей целевой системы. Иногда указывают и на другое отличие, состоящее в инверсии управления и вызове пользовательских функций непосредственно из модулей каркаса [200]. Однако это наблюдение является не совсем верным. Например, в библиотеках условной нелинейной оптимизации подобная инверсия применяется

с целью задания математических функций в виде соответствующих императивных процедур расчета их значений и производных. Такой способ исключает необходимость интерпретации математических функций, заданных декларативным образом, и повышает эффективность вычислительного процесса.

В любом случае организация каркаса не препятствует интеграции любого числа сторонних библиотек самой разной функциональности для решения вспомогательных задач. Более того, она может предусматривать применение специализированных языков для описания прикладных задач, форматов обменных файлов для вывода и хранения результатов, языков запросов к СУБД, протоколов взаимодействия клиентских и серверных приложений и т.п. Тем самым, концепция каркаса существенно расширяет идею библиотечной организации программного обеспечения, предусматривая развитые инструментальные возможности для построения целевых приложений.

3. 1. 2. Общие требования и принципы построения каркаса

Обсуждаемый объектно-ориентированный каркас сочетает в себе функции математической библиотеки и инструментальной среды для построения программных приложений теории расписаний и проектного планирования. Сформулируем общие требования, которые предъявлялись к нему и учитывались при его разработке:

- универсальность, предполагающая наличие готовых к использованию программных модулей для математически строгой постановки и решения типовых задач теории расписаний и проектного планирования;
- эффективность, означающая в данном случае равномерно высокую производительность модулей для решения обсуждаемого класса задач и, прежде всего, для приближенного решения индустриально значимых задач проектного планирования высокой размерности;
- гибкость, подразумевающая возможность повторного использования имеющихся модулей при программной реализации новых моделей,

методов и приложений теории расписаний при относительно низких затратах на доработку.

Данные отчасти противоречивые требования нуждаются в уточнениях, поскольку в значительной степени формируют общий функциональный облик всего каркаса.

Как известно, теория расписаний охватывает довольно много классов задач с разными оценками вычислительной сложности и со своими алгоритмами решения. Общепринятая нотация Грэхема $\alpha|\beta|\gamma$, в которой характеристики описывают соответствующие модели исполнения операций и машин (работ и ресурсов) и целевые функции, задает общую классификацию подобных задач. В зависимости от индивидуальных характеристик $\alpha|\beta|\gamma$ вычислительная сложность составления расписания может существенно варьироваться и поэтому для этих целей обычно применяют специальные алгоритмы, ориентированные на частные классы задач.

Важно отметить, что при разработке каркаса, как универсальной математической библиотеки, не ставилась цель предоставить средства, которые бы обеспечили решение всех задач теории расписания за оптимальное время. Вместо этого предпринята попытка эффективно решать задачи проектного планирования в постановке RCPSP, к которой редуцируются все основные задачи теории расписаний. Однако и данная классическая постановка оказывается довольно частной для реализации приложений календарно-сетевого планирования и управления проектами, в которых применяются сложные многопараметрические модели работ, связей, ресурсов, календарей, счетов. Некоторые отличия в постановках задач указаны в работе [10], в которой авторы говорят о задаче RCPSP в расширенной постановке. Необходимые математические обобщения также обсуждались и систематизировались в первой главе настоящей работы.

В главе 2 был определён класс задач обобщённого проектного планирования Generally Constrained Project Scheduling Problem (GCPSP) и сформулированы утверждения о сводимости задач RCPSP в расширенных

постановках к задачам GCPSP. Существенно, что последние формулируются в математически нейтральной форме, которая определяет лишь тип целевой функции и вид алгебраических ограничений, возникающих в задачах проектного планирования. Таким образом, универсальность каркаса может обеспечиваться путем предоставления развитого набора программных модулей для задания условий задач проектного планирования в постановке GCPSP и их решения. Для математической редукции прикладных задач достаточно проинтерпретировать их условия в терминах постановки проектного планирования GCPSP, разрешить ее и представить результаты в представлении исходной задачи.

Требования эффективности также нуждаются в некоторых пояснениях. Поскольку задачи проектного планирования RCPSP и GCPSP являются NP-полными, а для индустриальной практики представляют интерес проекты с количеством работ, исчисляемых десятками и сотнями тысяч, главное внимание должно уделяться быстрым алгоритмам, обеспечивающим поиск приближенных решений за полиномиальное время. Лучший из известных точных алгоритмов Брукера за приемлемое время может решать задачи размерности не больше 60 [169], что делает невозможным его использование в обсуждаемых индустриальных приложениях. Вместе с тем, точные алгоритмы могут применяться для валидации приближенных алгоритмов и, в частности, для выбора и настройки применяемых в них эвристических правил. Поэтому состав каркаса может предусматривать программные модули, реализующие и некоторые точные алгоритмы. Однако вопросы эффективности становятся некритичными, поскольку оценка качества найденных приближенных решений может осуществляться на тестовых задачах очень низкой размерности.

Под гибкостью каркаса как программно-инструментальной среды подразумевается возможность реализации новых математических моделей, методов и приложений теории расписаний при относительно низких затратах на доработку имеющихся программных модулей. Поскольку задачи теории расписаний редуцируются к соответствующей постановке обобщённого проектного планирования GCPSP, для задания условий и решения которой

основные программные модули уже реализованы и включены в состав каркаса, доработка целевых приложений потребует небольших затрат. В тех случаях, когда необходимо решать частные классы задач за оптимальное время, потребуются дополнительные усилия на программную реализацию специальных алгоритмов. В предположении, что они основаны на уже реализованных в каркасе алгоритмах или используют общую с ними вычислительную стратегию, подобные затраты также могут быть минимизированы.

Таким образом, обсуждаемые требования универсальности, эффективности и гибкости, предъявляемые к каркасу, могут быть удовлетворены на основе изложенных выше принципов.

3. 1. 3. Организация и состав классов каркаса

Разработанный каркас представляет собой систему классов (в дальнейшем, учитывая практическую реализацию на языке Си++, будем использовать принятые термины «класс», «конкретный класс», «абстрактный класс» и «интерфейс»). В организации каркаса можно выделить следующие пакеты классов и отношения использования между ними (рис. 2):

- классы решателей (Solvers), реализующие общие алгоритмические схемы решения задач GCPSP (Schedulers), а также эвристики для поиска приближенных решений (Heuristics);
- классы математических объектов (Mathematics), предназначенные для задания условий и представления результатов проектного планирования в обобщённой постановке GCPSP;
- классы редукции (Reductions), обеспечивающие сводимость прикладных задач составления расписаний к постановке GCPSP и соответствующую интерпретацию прикладных данных;
- классы прикладных данных (Project Data), используемые для представления условий и результатов решения задач проектного планирования RCPSP в расширенных постановках.

В следующих разделах остановимся более подробно на основных группах классов, непосредственно связанных с решением задач теории расписаний. Заметим, что часть из них реализуется как конкретные классы, допускающие непосредственное конструирование объектов. Другая часть представляет собой интерфейсы или абстрактные классы, которые по существу определяют точки расширения каркаса и позволяют предоставить их альтернативные реализации. Примечательно, что некоторые функции каркаса, в частности общие алгоритмические схемы, могут реализовываться в абстрактных классах без конкретизации типов условий решаемых задач и особенностей конкретных алгоритмов. Более того, такой способ реализации каркаса является рациональным с точки зрения повторного использования модулей и функциональной эволюции целевых приложений.

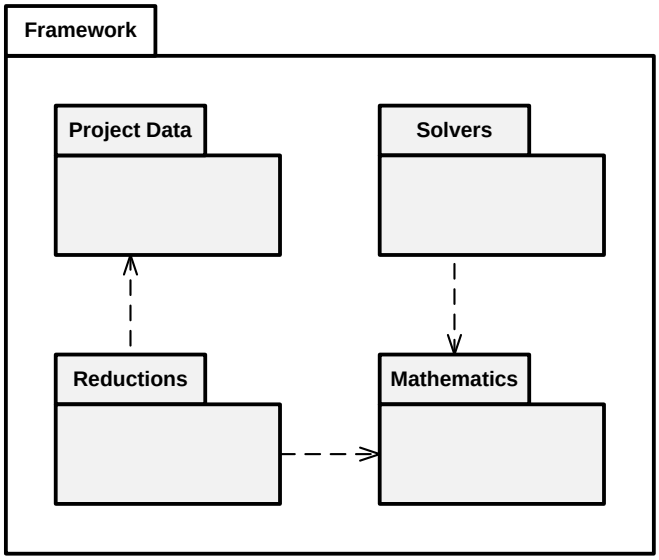

Рис. 2. Организация пакетов классов каркаса и основные отношения использования.

3. 2. Организация классов прикладных данных

В данном разделе подробно описываются классы и интерфейсы каркаса, относящиеся к группе Project Data и позволяющие задать условия и результаты задач проектного планирования в расширенных постановках RCPSP.

3. 2. 1. Класс «Проект» (Project)

Класс Project агрегирует в себе все данные, необходимые для математически корректной постановки задачи проектного планирования, и предоставляет необходимые интерфейсы доступа к ним. Сам проект представляется иерархией связанных между собой работ с назначенными календарями, ресурсами и счетами. В рамках ООП перечисленные понятия реализуются соответствующими классами Project, Task, Link, Calendar, Resource, Account соответственно (рис. 3). Классы Task и Resource являются абстрактными, что означает невозможность создания экземпляров и необходимость предоставления конкретных реализаций методов, объявленных в интерфейсах данных классов. В следующих подразделах подробно описываются особенности подобных реализаций. В частности, поясняются способы определения простых и составных типов работ и ресурсов. Дополнительные классы TaskRate, ResourceRate, ResourceUse, Supply и Replenishment используются для ассоциирования работ, ресурсов и счетов между собой и параметризации подобных отношений.

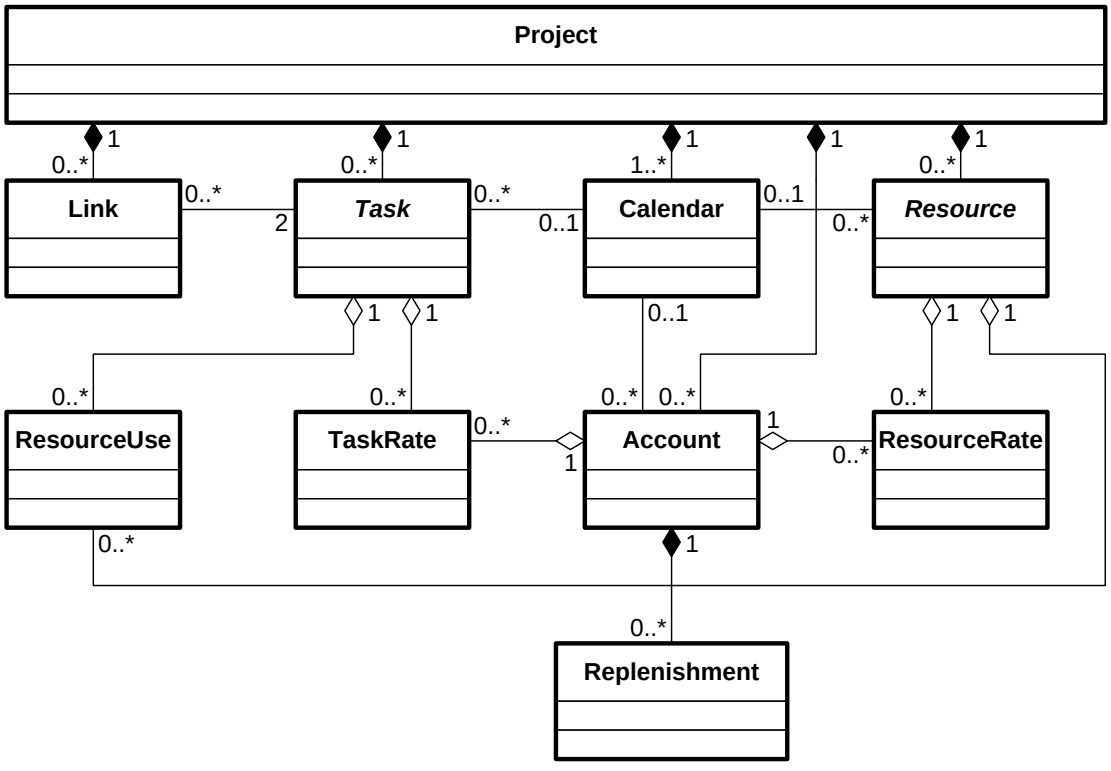

Рис. 3. Диаграмма классов UML для представления прикладных данных.

Перечисленные выше классы являются конкретными, однако следует принять во внимание, что в их основу положены довольно общие параметрические модели, охватывающие расширенные постановки обсуждаемого класса задач RCPSP. Вместе с тем, при необходимости данные модели могут быть развиты и реализованы путем непосредственного наследования классов каркаса.

Кроме агрегации экземпляров указанных на диаграмме классов, класс Project определяет собственные атрибуты. Такими атрибутами являются проектный календарь, используемый в качестве основного в тех случаях, когда не определён индивидуальный календарь для работ, ресурсов и счетов, а также временные проектные ограничения, определяющие время начала и/или завершения проекта и возможную стратегию прямого и обратного планирования проектных работ (как можно раньше и как можно позже соответственно).

3. 2. 2. Календарные данные

Работа с календарной информацией занимает важное место при постановке и решении задач проектного планирования. Сами календари представляют собой объекты с довольно сложной организацией данных и нетривиальными операциями пересчета календарных дат, времен, рабочих интервалов. Поэтому в их реализации используются вспомогательные классы Time, Date, DateAndTime, TimeInterval, Duration, DayOfWeek, WorkWeek, MonthOfYear, RecurrencePattern, RecurrenceTimeInterval, которые в составе каркаса выполняют и самостоятельные функции. Рассмотрим их более подробно.

Класс «Время» (Time)

Класс Time предназначен для представления времени в рамках одних суток с точностью до долей секунды. Переменная времени может принимать любое значение от 00:00:00 до 24:00:00. Класс реализует методы для установки времени суток с помощью компонентов-значений часа, минуты, секунды и долей секунды, а также методы для получения соответствующих компонентов-значений времени суток и его строкового представления. Интерфейс класса предусматривает

необходимые логические операции сравнения текущего времени с заданным значением, а также арифметические операции добавления заданной продолжительности к текущему времени и ее вычитания из текущего времени. Результат последних операций всегда приводится к суточному временному интервалу.

Класс «Дата» (Date)

Класс Date используется для представления календарной даты в виде компонентов-значений числа, месяца и года. Интерфейс класса предусматривает методы для установки календарных дат, сравнения дат, добавления к дате и вычитания из неё заданной продолжительности, вычисления продолжительности временного интервала между текущей датой и заданной.

Класс «Дата и время» (DateAndTime)

Класс DateAndTime предназначен для консолидированного представления календарной даты и времени суток, позволяющего оперировать абсолютными временными метками. Интерфейс класса во многом повторяет интерфейсы рассмотренных выше классов Date и Time.

Класс «Временной интервал» (TimeInterval)

Класс TimeInterval позволяет оперировать временными интервалами в пределах одних суток. Временной интервал задается нижней и верхней границей в предположении, что нижняя граница принадлежит интервалу, а верхняя — нет. Класс предоставляет методы для задания и получения границ временного интервала, вычисления его продолжительности, определения статуса принадлежности заданного момента времени текущему интервалу, а также вычисления теоретико-множественных операций пересечения и объединения текущего интервала с заданным интервалом.

Класс «Продолжительность» (Duration)

Данный класс используется для представления продолжительности проектных работ и лагов между ними. Продолжительность исчисляется с

точностью до долей секунды и может быть положительной, отрицательной или нулевой. В классе реализуются арифметические операции сложения и вычитания продолжительностей, операции умножения и деления текущей продолжительности на число, а также логические операции сравнения заданных продолжительностей.

Тип данных «День недели» (DayOfWeek)

Тип данных DayOfWeek предназначен для представления дней недели и естественным образом реализуется как перечислимый тип с семью предопределенными значениями: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday и Sunday.

Тип данных «Месяц года» (MonthOfYear)

Тип данных MonthOfYear предназначен для представления месяцев в году и реализуется в каркасе как перечислимый тип с двенадцатью предопределенными значениями: January, February, March, April, May, June, July, August, September, October, November и December.

Класс «Рабочая неделя» (WorkWeek)

Класс WorkWeek позволяет приписать логические признаки дням недели, например, с целью задания и определения их рабочего статуса. Класс реализуется как массив из семи логических значений, каждое из которых соответствует определенному дню недели в перечислении DayOfWeek. Интерфейс класса позволяет определить статус заданного дня недели и при необходимости изменить его.

Тип данных «Регулярное правило» (RecurrencePattern)

Обычно временные интервалы в рабочих календарях подчиняются некоторым регулярным правилам. Например, рабочие часы организации могут повторяться «ежедневно», «еженедельно», «ежемесячно», «в определённое число каждого месяца» и т. д. Для описания подобных регулярных правил в составе каркаса предусмотрен вспомогательный перечислимый тип данных

RecurrencePattern со следующими предопределенными значениями: Daily, Weekly, Monthly, MonthlyByDayOfMonth, MonthlyByPosition, ByDayCount, ByWeekdayCount, YearlyByDayOfMonth, YearlyByPosition. Именованные значения имеют понятную интерпретацию.

Класс «Регулярный временной интервал» (RecurrenceTimeInterval)

Класс RecurrenceTimeInterval предназначен для задания регулярных временных интервалов. Отличием от рассмотренного выше класса TimeInterval является более компактный, не избыточный способ представления временных интервалов в тех случаях, когда они подчиняются регулярным правилам, специфицируемым типом RecurrencePattern. Использование в подобных случаях класса TimeInterval привело бы к необходимости конструирования и последующего анализа огромного числа интервальных объектов для каждого проектного дня. Класс RecurrenceTimeInterval обобщает модель данных класса TimeInterval путем определения дополнительных атрибутов для задания регулярного правила, начальной и конечной даты его применения и соответствующих ему параметров (например, «день недели», «число месяца», «день года» и т.п.).

Класс «Календарь» (Calendar)

Исполнение работ и привлечение ресурсов в рамках проектной деятельности обычно осуществляется на основе рабочих календарей. Корректная постановка задачи проектного планирования предполагает, что должен быть определен, по крайней мере, один проектный календарь, используемый по-умолчанию. Для работы с календарями в состав объектно-ориентированного каркаса включен конкретный класс Calendar (рис. 4).

Календари могут быть рационально организованы в виде двух множеств, одно из которых соответствует регулярным интервалам рабочего времени, а другое — их исключениям в особые календарные дни. В классе Calendar для этих целей используется две коллекции workTime и exception с элементами соответствующего типа RecurrenceTimeInterval. Первая коллекция используется

для описания типовой рабочей недели (например, с понедельника по пятницу с 9:00 до 17:00). Вторая коллекция используется для описания исключений, которые могут происходить, например, в праздничные или предпраздничные дни. Фактическое рабочее расписание получается путем теоретико-множественной операции вычитания исключительных интервалов из основных рабочих интервалов. Выполнение данной операции может быть сопряжено со значительным объемом вычислений, поэтому следует избегать избыточной фрагментации исключений. Например, ежедневный обеденный перерыв с 13:00 до 14:00 нерационально описывать в виде исключения, а лучше представить прерываемыми рабочими интервалами.

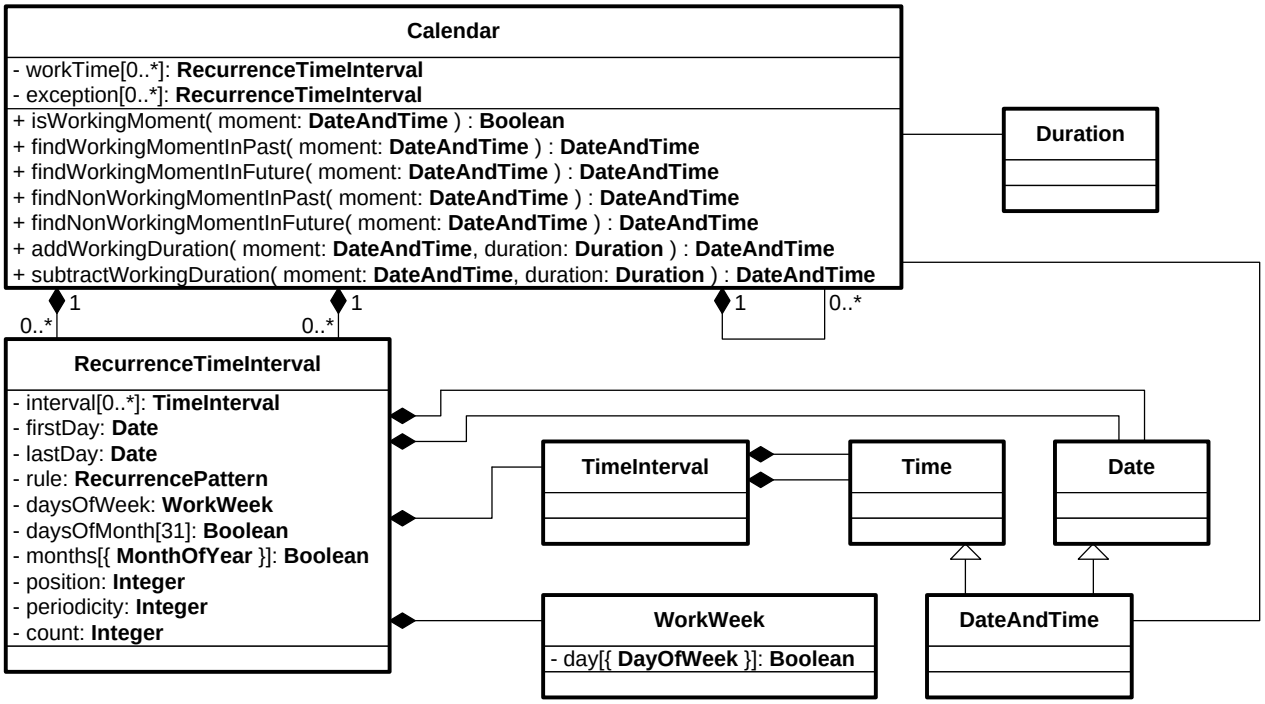

Рис. 4. Диаграмма классов UML для представления календарных данных.

Календари организуются в виде иерархической структуры с введенным отношением наследования между родительскими и дочерними элементами. Наследуемый календарь может уточнять или переопределять фактическое рабочее расписание родительского календаря. Предполагается, что собственные рабочие интервалы расширяют расписание родителя, а собственные исключения — ограничивают. Фактическое расписание наследника получается путем

объединения собственных рабочих интервалов с рабочими интервалами родителя после вычитания из них исключительных интервалов родителя и последующего вычитания из полученного результата исключительных интервалов наследника.

Класс Calendar реализует развитый набор операций для определения статуса заданной даты или заданного временного интервала, вычисления ближайших рабочих и нерабочих дат календаря, а также для пересчета даты завершения (или начала) работы по заданной дате ее начала (или завершения) и продолжительности. Интерфейс класса предусматривает также операции unite и intersect, которые позволяют сконструировать новые календари, фактическое расписание которых является объединением или пересечением расписаний заданных календарей-операндов. Данные операции упрощают реализацию основных вычислительных процедур составления расписаний с учетом календарей, которые могут быть индивидуально приписаны проектным работам, ресурсам, ограничениям предшествования и нуждаются в комплексном анализе при пересчете фактических дат начала и завершения проектных работ.

3. 2. 3. Проектные работы

Каркас использует естественное разделение проектных работ на простые и составные. Простыми работами являются элементарные активности Activity, вехи StartMilestone, FinishMilestone и «гамаки» ShortHammock, LongHammock. Для многоуровневого представления проектного плана в виде иерархии работ используются структуры работ WBS (принятое сокращение от Work Breakdown Structure) и мультимодальные работы MultimodalTask. Все перечисленные виды работ в каркасе реализуются соответствующими конкретными классами, наследуемыми от общего абстрактного класса Task (рис. 5). Допускается, что некоторые атрибуты, декларируемые в нем, могут оказаться производными в конкретных реализациях и поэтому попытки их установки могут приводить к исключительным ситуациям. Тем не менее, наличие общего интерфейса Task оказывается важным фактором, предопределяющим единую дисциплину реализации классов работ.

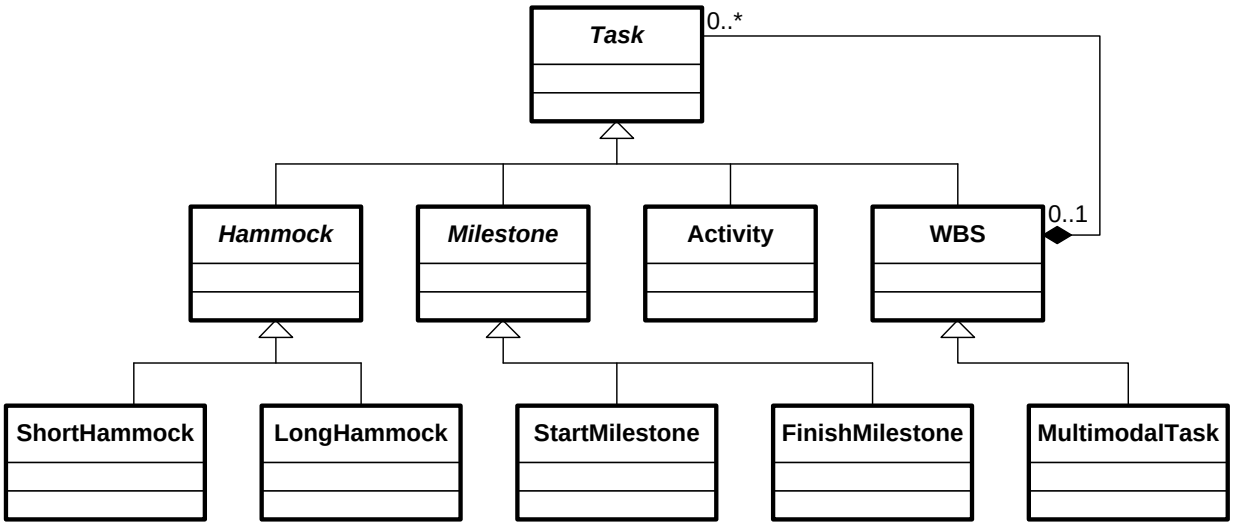

Рис. 5. Диаграмма иерархии классов UML для представления проектных работ.

Класс «Работа» (Task)

Абстрактный класс Task определяет общие методы получения и задания общих параметров работы. Прежде всего, это — планируемые и фактические даты начала и завершения работы, ее продолжительность, рабочий календарь, наложенные ограничения, используемые ресурсы, стоимость, счёт, приоритет, режим исполнения, статус и процент выполнения. В зависимости от вида работы логика реализации методов претерпевает существенные изменения и поэтому вынесена на уровень конкретных классов. Особенности полиморфной реализации далее обсуждаются на примере методов получения временных параметров работы.

Кратко уточним назначение перечисленных атрибутов работ. Рабочий календарь — календарь, в соответствии с которым осуществляется планирование и исполнение работы. В определении класса Task он реализуется опциональной ссылкой на объекты рассмотренного выше класса Calendar. При ее отсутствии применяется календарь проекта, наличие которого обязательно при конструировании объектов класса Project.

Статус определяет планируемое, стартованное, прерванное, возобновленное или финишированное состояние работы и представляется перечислимым типом TaskStatus с соответствующими значениями Planned, Started, Suspended, Resumed, Finished. Приоритет — натуральное число, характеризующее предпочтения

пользователя по приоритизации работ в ходе составления расписания. Обычно необходимость в их использовании возникает, когда одновременное выполнение работ невозможно из-за ограниченности общих ресурсов и требуется принять решение о порядке их выполнения.

Явные временные ограничения определяют допустимые интервалы или полуинтервалы для дат начала или завершения работ. Данные ограничения снабжаются спецификатором обязательности или приоритетности над ограничениями предшествования в тех случаях, когда формируемая система алгебраических уравнений и неравенств перегружена и не может быть полностью разрешена. Если спецификатор предписывает обязательное выполнение временного ограничения и это препятствует каким-либо ограничениям предшествования, то расписание строится до конца, но снабжается отчетом о неразрешенных ограничениях.

Правило выравнивания устанавливает необходимость принудительного выравнивания планируемой даты на начало или конец каждой минуты, часа, суток, недели, месяца или года и представляется в каркасе перечислимым типом SnapMode с соответствующими значениями SnapToMinute, SnapToHour, SnapToDay, SnapToWeek, SnapToMonth, SnapToYear. Правило выравнивания является опциональным атрибутом работы и может рассматриваться в качестве специфического временного ограничения, определяющего область допустимых значений для даты начала работы.

Для некоторых видов работ может быть указано максимально допустимое число прерываний, которое определяет возможные режимы их исполнения. Если данный атрибут не задан, то предполагается, что количество прерываний не ограничено. Нулевое значение интерпретируется как недопустимость прерываний.

Класс «Активность» (Activity)

Класс Activity реализует понятие простых операций, которые представляют собой терминальные работы в иерархическом представлении проектного плана.

Конкретный класс Activity наследует и реализует интерфейс класса Task таким образом, что любой параметр работы может быть задан индивидуально, но при одновременной коррекции других связанных с ним параметров. Тем самым, обеспечивается семантическая согласованность представления данных. Например, при установке планируемых дат начала и завершения работы, пересчитывается ее продолжительность. При установке доли выполнения пересчитывается прогнозируемое время завершения работы и т.п.

Классы «Вехи» (Milestone)

Близкое поведение реализуют классы StartMilestone и FinishMilestone. Основным отличием вех от простых операций является нулевая продолжительность и, как следствие, совпадающие даты начала и завершения работ. При этом даты в классе StartMilestone принудительно выравниваются на момент времени, допустимый для начала работы, а в классе FinishMilestone — на момент времени, допустимый для завершения работы. Установка даты начала вехи приводит к коррекции даты завершения и наоборот. Вызов метода установки продолжительности для вехи порождает соответствующее исключение.

Класс «Структура работы» (WBS)

Главной особенностью реализации класса WBS является наличие множественной композиции children на объекты типа Task, благодаря которой структуры работ могут содержать в себе дочерние работы любых типов, в том числе и работы более низких уровней. Таким образом, класс WBS позволяет структурировать проектный план в виде многоуровневой иерархии разнотипных работ. Структуры WBS не обязаны содержать дочерних работ, поскольку детализация проектного плана обычно происходит постепенно.

Явное задание временных параметров и других производных атрибутов WBS, используя методы наследуемого интерфейса Task, является некорректным и приводит к исключительным ситуациям. В самом деле, временные, ресурсные и стоимостные характеристики структуры работ определяются ее дочерними работами. Например, дата начала структуры работы определяется самым ранним

стартом дочерних работ, а дата ее завершения — самым поздним финишем. Вычисление данных параметров осуществляется путем рекурсивного обхода многоуровневого представления структуры работ и уточнения минимальных и максимальных значений соответствующих дат в дочерних работах. Аналогичным образом вычисляются ресурсные и стоимостные характеристики структур работ. Последние, в частности, применяются при оценке качества найденных решений в постановках, нацеленных на минимизацию сроков и стоимости проекта в условиях жестких ресурсных ограничений.

Классы «Гамаки» (Hammock)

Подобно структурам работ WBS реализуются «гамаки» классов ShortHammock и LongHammock. Они не предусматривают явного задания дат работ, поскольку рассчитываются по временным параметрам предшествующих и последующих работ. Дата начала «Короткого гамака» определяется самым поздним финишем из всех предшественников, а дата его завершения — самым ранним стартом последователей. Для «Длинного гамака» дата начала совпадает с самым ранним финишем из всех предшественников, а дата завершения — с самым поздним стартом последователей. В случае отсутствия предшественников и/или последователей «гамак» вырождается в работу с продолжительностью, равной продолжительности проекта.

Класс «Мультимодальная работа» (MultimodalTask)

Важным требованиям к средствам планирования сложных проектов является возможность задания альтернативных режимов выполнения. С этой целью в состав каркаса включен класс MultimodalTask, определение которого во многом повторяет класс WBS из-за использования композиции дочерних работ children. Однако логика реализации интерфейса Task принципиально отличается, поскольку выполнение структуры работ означает выполнение всех дочерних работ, а выполнение мультимодальной работы предполагает исполнение лишь одной из дочерних работ. Поскольку тип дочерних работ не конкретизируется композицией children в WBS и MultimodalTask, с их помощью удается строить

сложные стратегии проектной деятельности, например, многоуровневые альтернативы структур работ. Однако следует иметь в виду, что из-за комбинаторной неопределенности наличие мультимодальных работ в представлении проекта существенно усложняет поиск расписаний, близких к оптимальным.

3. 2. 4. Класс «Связь работ» (Link)

Класс Link предназначен для задания отношений предшествования и синхронизации между работами при постановке задач проектного планирования. В классе определяются две обязательные ассоциации на объекты типа Task, одна из которых указывает на предшествующую работу (upstreamTask), а другая — на последующую (downstreamTask). Четыре опциональных атрибута типа Duration определяют минимальную и максимальную задержку синхронизации, пересчитанную в календарные даты с использованием рабочих календарей предшественника и последователя. Незаданные минимальные задержки (upstreamMinLag и downstreamMinLag) интерпретируются как нулевые, а незаданные максимальные задержки (upstreamMaxLag и downstreamMaxLag) — как бесконечно большие величины. Опциональная ссылка на календарь типа Calendar используется для пересчета дат в соответствии с собственным календарем.

Кроме этого, объекты класса Link имеют в качестве атрибута спецификатор связи, представленный перечислимым типом LinkType. Данный атрибут может принимать одно из следующих значений: SSLink, SFLink, FSLink или FFLink. Значение SSLink означает, что связь устанавливается между началом предшественника и началом последователя, SFLink — между началом предшественника и окончанием последователя, FSLink — между окончанием предшественника и началом последователя и FFLink — между окончанием предшественника и окончанием последователя. С учётом данного спецификатора минимальные и максимальные задержки приобретают более понятный смысл. Например, для связи с типом FSLink при заданной минимальной задержке данный

вид связи устанавливает требование начать последующую работу не раньше, чем через установленное время после завершения предшествующей работы. При заданной максимальной задержке — не позже, чем через установленное время после завершения предшествующей. Примечательно, что задержки могут быть отрицательными и приводить к обратной последовательности выполнения предшественников и последователей.

3. 2. 5. Ресурсы

Каркас поддерживает несколько категорий ресурсов, реализуемых соответствующими классами простого ресурса SimpleResource, группового ресурса GroupResource, объединения ресурсов JointResource и семейства ресурсов FamilyResource. За исключением простого ресурса все остальные виды являются составными, что предполагает композицию дочерних или ассоциацию сторонних ресурсов.

Класс «Ресурс» (Resource)

Абстрактный класс Resource определяет базовый тип, от которого наследуются все перечисленные выше конкретные классы (рис. 6). Данный класс не предусматривает общий интерфейс для получения и задания ресурсных параметров в силу того, что простые и составные ресурсы параметризуются различными способами. Единственными общими методами, определяемыми в данном классе, являются правила исчисления стоимости ресурса.

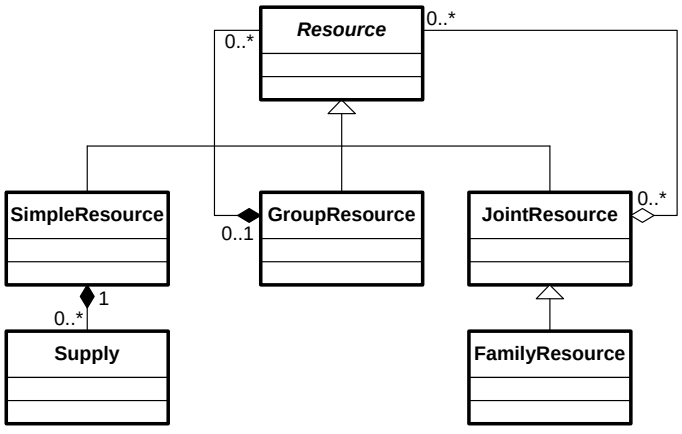

Рис. 6. Диаграмма иерархии классов UML для представления ресурсов.

Класс «Простой ресурс» (SimpleResource)

Класс SimpleResource реализует понятие простого ресурса со следующим набором параметров. Это — признаки возобновимости и разделяемости ресурса, рабочие календари, временные лаги и доступное количество ресурса.

Признак возобновимости представляется перечислимым типом со значениями Renewable и Expendable. Первое значение указывает на возврат используемого количества ресурса по окончании каждой работы в общий пул. Второе устанавливает, что ресурс тратится в ходе выполнения каждой работы, где он используется, и его доступное количество в ходе выполнения проекта только уменьшается. Обычно к возобновимым ресурсам относят рабочий персонал и технику, а к невозобновимым — расходные материалы и энергетические ресурсы.

Признак разделяемости представляется перечислимым типом со значениями Discrete и Continuous, устанавливающими привлекается ли ресурс дискретным образом или непрерывным. Например, если некоторая работа выполняется за один день и ее трудоемкость составляет 1,5 человеко-дня, то данный признак позволяет решить, необходимо ли привлечь для ее выполнения двух сотрудников на целый день (Discrete) или в условиях неполной занятости они могут одновременно участвовать в других параллельных работах (Continuous). Следует отметить, что доступное количество дискретных ресурсов всегда исчисляется в целых. Оба признака являются обязательными атрибутами простого ресурса, поскольку участвуют в оценке его доступности и влияют на логику составления расписания.

Основной календарь устанавливает рабочее время, в которое ресурс доступен для проектных работ. Обычно в качестве основного календаря используется проектный календарь, в соответствии с которым выполняются все основные работы. Однако возможны ситуации, когда ресурсные календари могут иметь отличия, например, в силу технологических особенностей применяемого оборудования или индивидуальных планов сотрудников. В подобных случаях, фактический календарь выполнения работы строится путем теоретико-множественного пересечения ее собственного календаря с календарями всех

используемых ресурсов. Для поддержки основного календаря используется обязательная объектная ссылка на класс Calendar.

Дополнительный календарь реализуется аналогичным образом, но является необязательной объектной ссылкой. Его основное назначение — планирование работ в сверхурочное время, обычно оплачиваемое по повышенным тарифам. Если целью планирования является скорейшее завершение проекта за счет возможного увеличения бюджета, то расписание строится с учетом основного и дополнительного календарей.

Временные лаги ресурса определяются как два опциональных атрибута класса, имеющие тип продолжительности Duration. Если атрибуты не установлены, то они интерпретируются как имеющие нулевую продолжительность. Первый атрибут определяет время доставки, установки или наладки ресурса перед его использованием в ходе выполнения работы, а второй атрибут — время остановки, демонтажа или возврата ресурса, необходимое для его освобождения и последующего использования в других работах. Временные лаги имеют спецификатор зависимости от количества используемого ресурса. При его задании итоговый временной лаг получается домножением на количество ресурса, используемого в конкретной работе. Перечисленные временные параметры также применяются в случаях прерывания и возобновления работ аналогично тому, как это происходит при их начале и завершении.

Класс «Поставка» (Supply)

Для получения доступного количества простого ресурса на заданный момент времени часто необходимо реконструировать историю поставок. С это целью в состав каркаса включен специальный класс Supply, а в классе SimpleResource определяется множественная композиция объектов данного класса. Множество объектов класса Supply реализует понятие «Цепочки поставок» путем определения для каждого объекта двух наборов атрибутов, определяющих планируемые и фактические значения следующих параметров: дата поставки, объем поставки или списания ресурса, стоимость организации всей

поставки, стоимость за единицу ресурса, а также счет, если он отличается он единого счета ресурса. Атрибуты, связанные с планируемыми величинами, являются обязательными, а атрибуты, связанные с фактическими величинами — опциональными, поскольку актуализация поставок осуществляется уже в процессе проектной деятельности.

Объем поставки является ее ключевой характеристикой, поскольку предопределяет количество ресурса, доступное на начало проекта или на начало любой спланированной работы. В ходе выполнения проекта ресурсы могут захватываться, потребляться, освобождаться, генерироваться, в результате чего их доступное количество меняется.

При составлении расписания важно учесть, что на протяжении всего проекта доступное количество каждого ресурса не может быть отрицательным. Нарушение этого ограничения означает, что проектные работы спланированы неправильно и используют несуществующие объемы ресурса. Если проект не предусматривает пополнение или генерацию ресурса, то ни одна работа не может использовать ресурса больше, чем доступно на начало проекта. Это условие может проверяться уже на этапе постановки задачи.

Класс «Групповой ресурс» (GroupResource)

Класс группового ресурса GroupResource применяется для организации разнородных ресурсов в единое иерархическое многоуровневое представление в соответствии с требованиями пользователей. С этой целью в данном классе, наследуемом от абстрактного интерфейса Resource, определяется множественная композиция composedOf объектов типа Resource. Это позволяет в каждый групповой ресурс включить любое количество разнородных ресурсов, в том числе и групповые ресурсы более низких уровней.

Класс «Объединение ресурсов» (JointResource)

Класс объединения ресурсов JointResource во многом аналогичен классу GroupResource за исключением того, что вместо композиции composedOf определяется множественная ассоциация assembledFrom на объекты типа

Resource. Класс предназначен для задания альтернативных способов группирования разнородных ресурсов, например, при формировании и комплектовании бригад. Назначение объединенного ресурса на работу предполагает использование в работе всех его ассоциируемых ресурсов.

Класс «Семейство ресурсов» (FamilyResource)

Класс семейства ресурсов FamilyResource аналогичен классу JointResource и использует множественную ассоциацию associatedWith типа Resource, что позволяет в одном объекте группировать разные ресурсы. Однако принципиальным семантическим ограничением является требование, чтобы все ассоциируемые ресурсы были однородными. Например, если один из простых ресурсов является возобновимым, то и все остальные ресурсы, включенные в семейство, должны быть возобновимыми. Семейства ресурсов непосредственно назначаются на работы, однако любое такое назначение допускает использование любых комбинаций ассоциируемых родственных ресурсов. Например, если семейство ресурсов определяет группу сотрудников соответствующей специальности и квалификации, то назначение семейства на работу будет означать, что для выполнения работы может привлекаться любой свободный сотрудник или сотрудники, незанятые в период выполнения работы.

Класс «Использование ресурса» (ResourceUse)

Класс ResourceUse позволяет ассоциировать работу и используемый ей ресурс путем установки ссылок на соответствующие объекты (рис. 7) и задания значений атрибутов, определяющих условия привлечения ресурса, включая количество или лимиты использования ресурса, профиль потребления или генерации ресурса на протяжении работы.

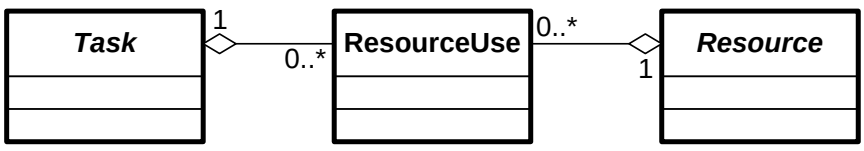

Рис. 7. Диаграмма классов UML для представления использования ресурсов.

Последний атрибут определяется как перечислимый тип ResourceProfile со значениями BackLoaded, BellShaped, FrontLoaded, Linear, OffsetTriangular, ThreeStep, Trapezoidal, TriangularDecrease, TriangularIncrease, DoublePeak, EarlyPeak, определяющими вид соответствующих скалярных функций одной переменной. Реальные профили потребления ресурса получаются путем масштабирования нормированных функций по оси абсцисс на период выполнения работы и по оси ординат на количество привлекаемого ресурса.

Следует иметь в виду, что установленное количество ресурса может быть отрицательным, что означает генерацию работой данного ресурса и возможность использования дополнительного количества другими работами. Примерами работ, генерирующих возобновимые и невозобновимые ресурсы, могут служить краткосрочная аренда дополнительного оборудования и производство вспомогательных материалов в ходе проектной деятельности.

3. 2. 6. Финансовое обеспечение

Важным аспектом проектной деятельности является бюджетно-финансовое планирование и обеспечение. С точки зрения дизайна каркаса (рис. 8) ключевыми элементами здесь являются бюджетный счет, а также различные правила исчисления стоимости работ и ресурсов.

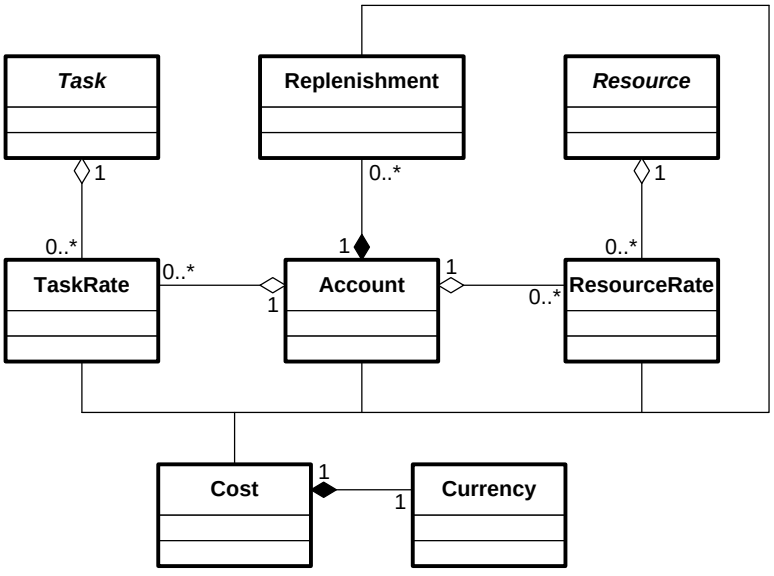

Рис. 8. Диаграмма классов UML для представления финансового обеспечения.

Классы «Стоимость» (Cost) и тип данных «Валюта» (Currency)

Вспомогательный класс Cost реализует понятие стоимости, выраженной в денежно-валютном эквиваленте. Объекты данного класса представляются парой значений: денежным номиналом и видом валюты, в которой данный номинал представлен. Вид валюты задается перечислимым типом данных Currency. Номинал может быть положительным, отрицательным или нулевым. Для объектов класса определены арифметические операции сложения, вычитания, умножения и деления на вещественное число и логические операции сравнения. В случаях, когда валюты двух стоимостных операндов не совпадают, значения номиналов приводятся к единой валюте по предопределенному обменному курсу. Для подобных целей класс предусматривает статические методы установки кросс-курсов и применения их к заданным номиналам.

Тип данных «Стоимость за…» (CostType)

Тип данных CostType предназначен для спецификации выставляемой стоимости, ставки или тарифа. Другими словами, атрибуты данного типа позволяют определить, за что указана стоимость. Данный тип представлен в каркасе как перечисляемый тип со следующими предопределёнными значениями: FixedCost (стоимость фиксирована и не зависит от какой-либо продолжительности или количества), CostPerMinute, CostPerHour, CostPerDay, CostPerWeek, CostPerMonth, CostPerYear (приведенная стоимость за соответствующую единицу времени), CostPerUnitMinute, CostPerUnitHour, CostPerUnitDay, CostPerUnitWeek, CostPerUnitMonth, CostPerUnitYear (приведенная стоимость за использование одной единицы ресурса в течении единицы времени), CostPerUnit (стоимость за использование одной единицы ресурса).

Класс «Счёт» (Account)

Класс Account реализует понятие банковского счёта или кошелька. Основным атрибутом класса являются начальный баланс счёта initBalance типа Cost. Все счета в проекте представлены плоским списком, любой счёт может

использоваться как для списания финансовых средств, так и для их пополнения. Счета могут быть ассоциированы с работами, ресурсами, календарями или проектом в целом.

Класс «Пополнение счёта» (Replenishment)

Для учёта доступности финансовых средств каркас предусматривает класс Replenishment, экземпляры которого описывают поступление финансовых средств на счёт. Каждый экземпляр класса Replenishment хранит объектную ссылку на счёт назначения (зачисления), сумму поступлений с указанием валюты (типы Cost и Currency), а также планируемую и актуальную даты поступления финансовых средств на счёт. При этом планируемая дата является обязательным атрибутом, а актуальная — опциональным. Используя приписанные объекты Replenishment можно реконструировать профиль доступных финансовых средств на счете. В этом смысле назначение и организация данного класса аналогична рассмотренному выше классу Supply.

Класс «Исчисление стоимости работы» (TaskRate)

Класс TaskRate позволяет ассоциировать работу и финансовый счет, откуда привлекаются средства для ее выполнения или куда зачисляются средства в случае ее прибыльности. Каждый экземпляр класса хранит объектные ссылки на работу и на финансовый счет, а также значения атрибутов, устанавливающих характер расходов или доходов (атрибут типа CostType), фиксированную или приведенную стоимость выполнения работы (атрибут типа Cost), стоимость прерываний работы (атрибут типа Cost) и профиль финансирования (атрибут типа CostProfile). Опциональными атрибутами класса являются актуальная стоимость и период действия применяемого тарифа. Данные атрибуты необходимы, чтобы оценить штрафные санкции в случае задержки или опережения работ относительно планируемых дат.

Приведенная стоимость может быть отнесена к единице рабочего или календарного времени, а также к единице трудозатрат работы. Профиль финансирования представляется перечислимым типом CostProfile со значениями

AtStart, AtEnd и Uniform, устанавливающими, что средства списываются со счета или зачисляются на счет в начале соответствующего временного периода, в его конце или расходуются равномерно на протяжении всего периода. Примечательно, что с одной и той же работой может быть ассоциировано несколько экземпляров данного класса. Кроме того, следует принимать во внимание, что в роли ассоциированной работы могут быть не только простые активности, но и любые другие виды работ (вехи, гамаки, структуры работ или мультимодальные работы).

Класс «Исчисление стоимости ресурса» (ResourceRate)

Организация класса ResourceRate «Исчисление стоимости ресурса» аналогична рассмотренному выше классу TaskRate за исключением того, что он определяет не стоимость выполнения работы, а стоимость привлечения использования ресурсов при выполнении работ. В данном классе определяются объектные ссылки на соответствующие экземпляры ресурса и счёта, опциональные даты начала и конца действия тарифа, планируемые и актуальные значения стоимости, характер стоимости и её выплаты. С одним ресурсом может быть ассоциировано несколько экземпляров данного класса. Итоговая стоимость использования ресурса в той или иной работе будет определяться как сумма затрат по каждому из тарифов с учётом временных показателей работы. Коллекция объектов ResourceRate с установленными датами действия тарифов позволяет реконструировать всю историю изменений стоимости ресурса в ходе проектной деятельности.

3. 3. Организация классов математических объектов и решателей

Пакеты классов математических объектов (Mathematics) и решателей (Solvers) в определённой степени изолированы от классов прикладных данных (Applications), рассмотренных выше. Данные классы реализуют математические понятия и алгоритмы теории расписаний (рис. 9). Для сведения прикладных задач к постановке обобщённого проектного планирования, формулируемой в

математически нейтральной форме, предусмотрены специальные классы редукции (Reductions), которые реализуют интерфейсы математических объектов с учетом особенностей прикладных задач и, тем самым, выполняют функции посредников между прикладными и математическими классами каркаса.

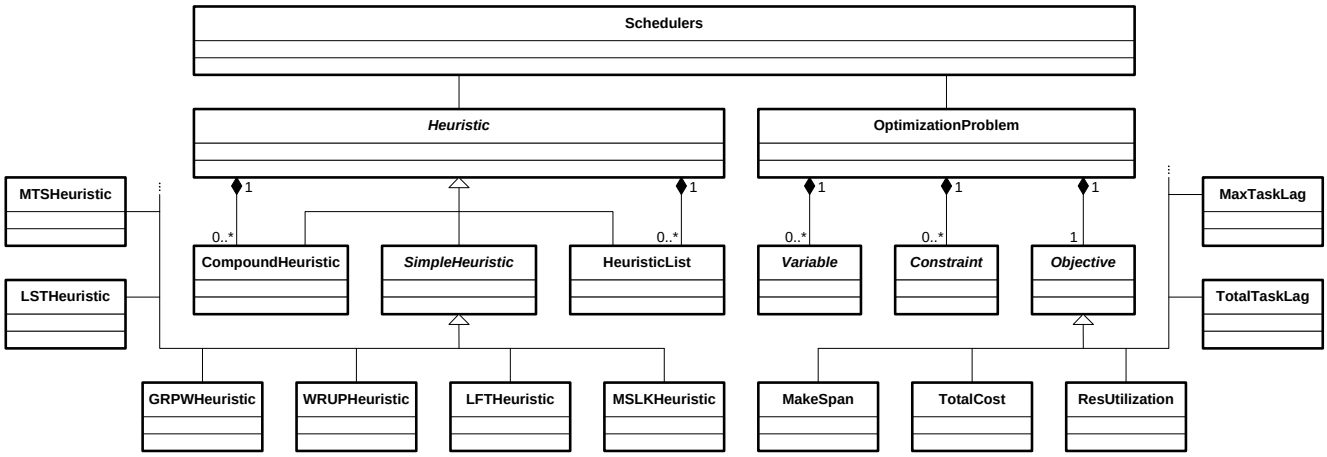

Рис. 9. Диаграмма классов UML математических объектов и решателей.

3. 3. 1. Класс «Оптимизационная задача» (OptimizationProblem)

Класс OptimizationProblem предназначен для постановки задачи условной нелинейной оптимизации путем задания множества переменных, целевой функции и системы нелинейных алгебраических ограничений. Класс реализуется как композиция всех математических объектов, участвующих в постановке задачи. Такими объектами являются переменные задачи класса Variable, целевая функция класса Objective и алгебраические ограничения класса Constraint.

Поскольку постановка оптимизационной задачи полностью определяется особенностями прикладной задачи проектного планирования, конструирование объектов OptimizationProblem и их наполнение условиями задачи осуществляется в классе Project с помощью вызова метода initOptimizationProblem(). Реализация данного метода предполагает обход всех экземпляров прикладных классов, ассоциируемых с проектом, и вызов соответствующих одноимённых методов для них. В результате каждый экземпляр прикладных классов, в том числе и сам проект, конструирует и инициализирует соответствующие математические объекты и включает их в композицию класса OptimizationProblem. Заметим, что

конструируемые математические объекты являются экземплярами редукционных классов, которые наследуют интерфейсы математических классов Variable, Objective, Constraint и реализуют их с учетом особенностей прикладной задачи. С этой целью они хранят ссылки на соответствующие прикладные данные. Например, для инициализации целевой функции оптимизационной задачи типа Objective создается соответствующий объект, который хранит ссылку на прикладной объект Project и через нее получает доступ ко всем проектным данным, участвующим в вычислении значений целевой функции. Аналогично задаются алгебраические ограничения. Опишем реализацию используемых вспомогательных классов более подробно.

3. 3. 2. Класс «Область допустимых значений» (ValueDomain)

Вспомогательный класс ValueDomain предназначен для задания области допустимых значений переменных в решаемой задаче. Поскольку переменные задачи могут выражать разные понятия и описываться разными типами, то наиболее рациональной представляется реализация класса ValueDomain в виде шаблона, параметризуемого типом переменной. В обсуждаемой постановке обобщённого проектного планирования все переменные задачи связаны с временными характеристиками, поэтому применяется конкретная реализация.

Область значений может быть представима отдельными точками, интервалами, бесконечными полуинтервалами, причем все элементы множества не пересекаются и упорядочены. Это обеспечивает эффективность операций проверки принадлежности заданного значения или интервала заданной области, пересечения и объединения заданных областей.

В интерфейсе данного класса для этих целей предусмотрены соответствующие методы:

- boolean isEmpty() — возвращает True, если множество допустимых значений пусто, и False в противном случае;

- boolean contains(ValueDomain&) — возвращает True, если множество допустимых значений полностью содержит заданную область, и False в противном случае;
- ValueDomain unite(ValueDomain&) — объединяет текущую область допустимых значений с другой, переданной в качестве параметра, и возвращает результат как новый экземпляр класса;
- ValueDomain intersect(ValueDomain&) — пересекает текущую область допустимых значений с другой, переданной в качестве параметра, и возвращает результат как новый экземпляр класса.

3. 3. 3. Интерфейс «Переменная» (Variable)

Интерфейс Variable определяет методы доступа к переменным задачи проектного планирования, получения, хранения и установки их значений. Используя интерфейс Variable и стандартные шаблоны коллекций, можно определить вектор переменных задачи, например, как ArrayOf<Variable*>.

Поскольку данные переменные связаны с прикладными данными, а именно с временными характеристиками проектных работ, то в составе каркаса предусмотрены конкретные классы TaskStart и TaskFinish, которые наследуя интерфейс Variable, предоставляют доступ к дате и времени начала и завершения работ типа DateAndTime. Кроме заданных значений даты и времени, каждая переменная может принимать неустановленное значение Unset и неизвестное значение Unknown. Например, неизвестное значение присваивается всем переменным перед решением задачи для указания необходимости выполнения соответствующих вычислений. Статус неустановленной переменной означает, что данная переменная была исключена из процесса решения в силу логических условий, связанных с выполнимостью соответствующих работ. Кроме методов доступа к переменным интерфейс Variable определяет метод получения ограничений задачи, в которых данная переменная участвует. Данный метод используется в ходе составления расписаний при анализе и согласованном разрешении наложенных ограничений.

3. 3. 4. Интерфейс «Целевая функция» (Objective)

Обсуждаемая постановка задач обобщённого проектного планирования допускает использование альтернативных целевых функций. Поэтому целевая функция в составе каркаса определяется как абстрактный класс с единым интерфейсом, от которого наследуются возможные конкретные реализации, а именно: MakeSpan (время выполнения всего проекта), TotalCost (суммарная стоимость всего проекта), ResUtilization (использование ресурсов), TotalTaskLag (суммарная задержка по всем работам относительно их директивных сроков), MaxTaskLag (максимальная задержка из всех работ относительно директивных сроков) и другие.

Единый интерфейс Objective определяет два основных метода:

- float getValue(ArrayOf<Variable*>&) — возвращает вещественное значение целевой функции по заданному вектору переменных задачи. Реализация метода в конкретных классах должна допускать корректную работу в тех случаях, когда не все переменные имеют предустановленные значения, но может быть вычислена, например, нижняя оценка целевой функции;
- DateAndTime getArgMin(ArrayOf<Variable*>&, int index, ValueDomain&) — возвращает значение заданной переменной, при которой достигается минимум целевой функции на заданной области при фиксированных значениях других переменных.

3. 3. 5. Интерфейс «Ограничение» (Constraint)

Принципы организации интерфейса Constraint и наследуемых от него конкретных классов прикладных ограничений во многом аналогичны рассмотренным выше. Интерфейс Constraint определяет следующие методы:

- bool isSatisfied() — возвращает True, если ограничение удовлетворено на ассоциируемом с ним множестве переменных и False в противном случае;
- getVariables(ArrayOf<Variable*>&) — возвращает множество ассоциируемых с ограничением переменных;

- DependencyType getDependency(Variable&) — возвращает одно из значений Independent, Dependent, Vague, определяющее характер зависимости переменной с указанным индексом;
- int getPriority() — возвращает целочисленный приоритет ограничения, используемый при разрешении переопределенных систем;
- ValueDomain resolveFor(ArrayOf<Variable*>&, int index) — возвращает множество значений для заданной переменной, при которых ограничение разрешается при фиксированных других переменных. Предполагается, что реализация данного метода в ограничениях регулярного алгебраического вида должна учитывать переменные с возможным неизвестным состоянием, которые в этом случае интерпретируются как отсутствующие.

Классы прикладных ограничений, наследуя интерфейс Constraint, реализуют данные методы в результате доступа к соответствующим прикладным данным и поэтому отнесены к пакету Reductions. Например, класс CalendarConstraint, ассоциируемый с классом Calendar, реализует ограничение допустимого рабочего времени начала или завершения работы. Классы TaskConstraint и TaskSnapping, ассоциируемые с классом Task, реализуют явные временные условия и правила выравнивания работ, атрибуты которых являются фактическими параметрами порождаемых алгебраических ограничений.

Аналогичным образом реализуются другие классы прикладных ограничений. Класс ActivityDuration определяет строгую алгебраическую зависимость между временами начала и завершения работы с учетом её продолжительности и применяемого календаря. Классы WBSStart и WBSFinish определяют соответствующие зависимости старта и завершения структуры работ от соответствующих параметров дочерних работ. Классы ShortHammockStart, ShortHammockFinish, LongHammockStart и LongHammockFinish определяют аналогичные зависимости «гамаков» от взаимосвязанных предшественников и последователей, классы LinkMinLag и LinkMaxLag — условия предшествования работ с учётом минимального и максимального временного лага,

ResourceConstraint — условие доступности ресурса, AccountConstraint — условие наличия средств на счете.

3. 3. 6. Интерфейс «Эвристика» (Heuristic)

Для решения задач проектного планирования в обобщённой постановке разработан приближенный алгоритм, основанный на эвристиках [4]. Каркас предоставляет реализацию данного алгоритма в виде соответствующего класса Scheduler. Однако использование последнего предполагает задание эвристик в виде упорядоченного множества объектов соответствующего типа Heuristic.

Интерфейс Heuristic определяет единственный метод:

- VariablePriority compare(Variable& var1, Variable& var2) — возвращает результат сравнения приоритетов пары переменных в виде одного из значений перечислимого типа VariablePriority: FirstOverSecond (первая переменная приоритетнее), SecondOverFirst (вторая переменная приоритетнее), Equal (приоритеты переменных равны). В качестве входных переменных метод принимает ссылки на сравниваемые переменные задачи.

Данный метод реализуется в конкретных классах, наследуемых от интерфейса Heuristic. В частности, каркас предоставляет готовые к использованию реализации следующих эвристик:

- MISHeuristic (Most Immediate Successors) — выбирает переменную с наибольшим количеством непосредственных переменных-последователей;
- LISHeuristic (Least Immediate Successors) — выбирает переменную с наименьшим количеством непосредственных переменных-последователей;
- MTSHeuristic (Most Total Successors) — выбирает переменную с наибольшим количеством всех переменных-последователей;
- LTSHeuristic (Least Total Successors) — выбирает переменную с наименьшим количеством всех переменных-последователей;
- SPTHeuristic (Shortest Process Time) — выбирает переменную с наименьшей разностью значений с переменными-последователями;

- LPTHeuristic (Longest Process Time) — выбирает переменную с наибольшей разностью значений с переменными-последователями.

Более строгое описание эвристик приводится в главе 2. Поскольку конкретные классы переменных имеют непосредственную связь с прикладными объектами, формирующими вектор переменных, то становится возможной реализация предметно-ориентированных эвристик с учетом особенностей прикладной задачи и более быстрых способов составления расписаний.

Поскольку применение отдельной эвристики не гарантирует вынесение окончательного вердикта о приоритете одной переменной над другими, на практике применяются иерархические стратегии, состоящие в последовательном применении нескольких эвристик. В каркасе такая возможность обеспечивается заданием упорядоченного множества разнотипных объектов типа Heuristic.

3. 3. 7. Класс «Решатель» (Scheduler)

Scheduler является конкретным классом, реализующий алгоритмы точного и приближенного решения оптимизационной задачи построения расписания. Интерфейс класса определяет пять основных метода:

- boolean state(OptimizationProblem&) — задаёт оптимизационную задачу составления расписания. Возвращает True в случае корректно заданных условий и False в противоположном случае;
- setHeuristics(ArrayOf<Heuristic*>&) — задать упорядоченное множество эвристик для приближенного решения;
- boolean solveApproximately() — ищет решение поставленной задачи, используя описанный приближенный алгоритм с предустановленными эвристиками. Возвращает True, если расписание было успешно составлено и False в противоположном случае;
- boolean solveExactly() — ищет решение заданной оптимизационной задачи, используя точный алгоритм границ и ветвей. Возвращает True в случае

успешного поиска решения и False в противоположном случае, например, при исчерпании отведенного процессорного времени;

- generateReport(Report&) — генерирует отчет о процессе работы алгоритмов и качестве найденного приближенного решения.

Рассмотрим листинг программы на языке Си++, иллюстрирующий типовую последовательность вызова методов, а также реализацию основных и вспомогательных методов класса. Основное внимание уделим методу реализации приближенного алгоритма solveApproximately().

```
void main()
  {
  …
  Project project;
  …
  OptimizationProblem problem;
  project.initOptimizationProblem( problem );
  ArrayOf<Heuristic*> heuristics;
  heuristics.push_back( new MISHeuristic() );
  heuristics.push_back( … );
  …
  Scheduler scheduler;
  if ( scheduler.state( problem ) )
    {
    scheduler.setHeuristics( heuristics );
    if ( scheduler.solveApproximately() )
      {
      ArrayOf<Variable*> &allVariables =
                                m_problem.getVariables();
      for ( int i = 0; i < allVariables.size(); ++i )
        allVariables[ i ].submit();
      }
    }
  Report report;
  scheduler.generateReport( report );
  …
  }

bool Scheduler::state( OptimizationProblem &problem )
  {
  m_problem = &problem;
  return true;
```

```
  }

void Scheduler::setHeuristics( ArrayOf<Heuristic*>
                                &heuristics )
  {
  m_heuristics = &heuristics;
  }

bool Scheduler::solveApproximately()
  {
  ArrayOf<Variable*> &allVariables =
                       m_problem.getVariables();
  checkAllVariablesAsUnknown( allVariables );
  ArrayOf<int> activeVariableIndices;
  computeIndependentVariableIndices( activeVariableIndices,
                                     allVariables );
  while ( activeVariableIndices.size() > 0 )
    {
    int selectIndex = selectVariable(
                                   activeVariableIndices,
                                   allVariables );
    if ( ! computeVariable( allVariables, selectIndex ) )
      return false;
    updateActiveVariableIndices( activeVariableIndices,
                                 allVariables, selectIndex );
    }
  return true;
  }
```

Вначале метод переводит переменные задачи в неизвестное состояние Unknown вызовом вспомогательного метода setAllVariablesAsUnknown(). Далее определяется множество независимых переменных с помощью метода computeIndependentVariableIndices() и инициализируется множество активных переменных activeVariableIndices. Дальнейшая работа алгоритма предполагает циклическую обработку данного множества. На каждом шаге цикла алгоритм выбирает одну из переменных на основе предустановленных эвристик (метод selectVariable()) и пытается найти её допустимое значение, удовлетворяющее всем ассоциированным с ней ограничениям (метод computeVariable()). Если допустимое значение найти не удаётся, то работа алгоритма прерывается с вердиктом о некорректно заданных условиях поставленной задачи. В случае

успешного поиска множество активных переменных обновляется путем исключения найденной переменной и добавления новых переменных, зависящих только от уже обработанных (метод updateActiveVariableIndices()). Ниже приведён листинг перечисленных вспомогательных методов.

```
void Scheduler::checkAllVariablesAsUnknown(
                ArrayOf<Variable*> &variables ) const
  {
  for ( int i = 0; i < variables.size(); ++i )
    variables[ i ]->checkAsUnknown();
  }

void Scheduler::computeIndependentVariableIndices(
                ArrayOf<int> &activeVariableIndices, const
                ArrayOf<Variable*> &allVariables ) const
  {
  activeVariableIndices.clear();
  for ( int i = 0; i < allVariables.size(); ++i )
    if ( isNonDependentVariable( allVariables[ i ]->
         getConstraints(), allVariables[ i ] ) )
      activeVariableIndices.push_back( i );
  }

bool Scheduler::isNonDependentVariable( const
                ArrayOf<Constraint*> constraints,
                const Variable* variable ) const
  {
  for ( int i = 0; i < constraints.size(); ++i )
    if ( constraints[ i ]->getDependency( variable ) ==
         DEPENDENT )
    return false;
  return true;
  }

int Scheduler::selectVariable( const ArrayOf<int>
            &activeVariableIndices,
            const ArrayOf<Variable*> &allVariables ) const
  {
  ArrayOf<int> candidates = activeVariableIndices;
  ArrayOf<int> result;
  for ( int i = 0; i < ( *m_heuristics ).size(); ++i )
    {
```

```
    getPriorityVariableIndices( result, candidates,
                    allVariables, ( *m_heuristics )[ i ] );
    if ( result.size() == 1 )
      return ( result[ 0 ] );
    candidatIndices = result;
    }
  return ( result[ 0 ] );
  }

void Scheduler::getPriorityVariableIndices( ArrayOf<int>
                &result,
                const ArrayOf<int> &candidates,
                const ArrayOf<Variable*> &allVariables,
                Heuristic* heuristic ) const
  {
  result.clear();
  result.push_back( candidates [ 0 ] );
  for ( int i = 1; i < candidates.size(); ++i )
    {
    VariablePriority priority = heuristic->compare(
                    ( *( allVariables[ result[ 0 ] ] ) ),
                    ( *( allVariables[ candidates[ i ] ] )
                    ) );
    if ( priority == FIRST_OVER_SECOND )
      continue;
    if ( priority == SECOND_ OVER_FIRST )
      result.clear();
    result.push_back( candidates[ i ] );
    }
  }

bool Scheduler::computeVariable( ArrayOf<Variable*>
                        &allVariables,
                        const int &selectedIndex ) const
  {
  ArrayOf<Constraint*> drivingConstraints;
  getDrivingConstraints( drivingConstraints,
                allVariables[ selectedIndex ]->
                getConstraints(),
                allVariables[ selectedIndex ] );
  sortConstraintsByPriority( drivingConstraints );
  if ( drivingConstraints[ 0 ]->canSkipVariable(
                allVariables[ selectedIndex ] ) )
    {
    allVariables[ selectedIndex ]->unset();
```

```
    return true;
    }
  ValueDomain resultD;
  ValueDomain tempD;
  for ( int i = 0; i < drivingConstraints.size(); ++i )
    {
    tempD = drivingConstraints[ i ]->resolveFor(
    allVariables, selectedIndex );
    if ( i == 0 )
      {
      if ( tempD.isEmpty() )
        {
        reportError( drivingConstraints[ i ],
                     allVariables[ selectedIndex ] );
        return false;
        }
      else
        resultD = tempD;
      }
    else
      {
      tempD = tempD.intersect( resultD );
      if ( tempD.isEmpty() )
        reportUnresolvedConstraint( drivingConstraints[ i ],
                              allVariables[ selectedIndex ] );
      else
        resultD = tempD;
      }
    }
  allVariables[ selectedIndex ] = ( m_problem->
                                getObjective() ).
                                getArgMin( allVariables,
                                selectedIndex, resultD );
  return true;
  }

void Scheduler::getDrivingConstraints( ArrayOf<Constraint*>
                    &drivingConstraints,
                    const ArrayOf<Constraint*> constraints,
                    const Variable* variable ) const
  {
  drivingConstraints.clear();
  for ( int i = 0; i < constraints.size(); ++i )
    if ( constraints[ i ]->getDependency( *variable ) !=
                                                INDEPENDENT )
```

```
      drivingConstraints.push_back( constraints[ i ] );
  }

void Scheduler::updateActiveVariableIndices( ArrayOf<int>
                    &activeVariableIndices,
                    const ArrayOf<Variable*> &allVariables,
                    const int &selectedIndex ) const
  {
  activeVariableIndices.erase( activeVariableIndices.
                          find( selectedIndex ) );
  const ArrayOf<Constraint*> &constraints =
          allVariables[ selectedIndex ]->getConstraints();
  for ( int i = 0; i < constraints.size(); ++i )
    if ( constraints[ i ]->getDependency(
                  *( allVariables[ selectedIndex ] ) ) ==
                  INDEPENDENT )
      {
      const ArrayOf<Variable*> &assVars =
                            constraints[ i ]->getVariables();
      for ( int j = 0; j < assVars.size(); ++j )
        if ( constraints[ i ]->getDependency(
                           *( assVars[ j ] ) ) == DEPENDENT )
          if ( isActiveVariable( assVars[ j ] ) )
            activeVariableIndices.push_back(
                      allVariables.find( assVars[ j ] ) );
      }
  }

bool Scheduler::isActiveVariable( const Variable* variable
                                  ) const
  {
  const ArrayOf<Constraint*> &constraints =
                                  variable->getConstraints();
  for ( int i = 0; i < constraints.size(); ++i )
    if ( constraints[ i ]->getDependency( *variable ) ==
                                                DEPENDENT )
      {
      const ArrayOf<Variable*> &assVars =
                            constraints[ i ]->getVariables();
      for ( int j = 0; j < assVars.size(); ++j )
        if ( constraints[ i ]->getDependency( assVars[ j ] )
                                          == INDEPENDENT )
          if ( assVars[ j ]->isUnknown() )
            return false;
      }
```

```
return true;
}
```

3. 4. Метод инкрементальной разработки приложений теории расписаний на основе каркаса

Одним из принципиальных требований, предъявляемых к каркасу, была возможность повторного использования имеющихся модулей при программной реализации новых моделей, методов и приложений теории расписаний при относительно низких затратах на доработку.

Во многом данное требование удается удовлетворить благодаря принятым в качестве методологической основы принципам объектно-ориентированного программирования и оригинальной многослойной архитектуре каркаса. Рассмотрим их более подробно на примере разработки программных приложений теории расписаний и, в частности, приложений календарно-сетевого планирования.

Каркасом предусматривается довольно развитый набор готовых к использованию модулей и поэтому разработка типового приложения календарно-сетевого планирования, главным образом, сводится к реализации графического интерфейса пользователя (GUI), подключению и конфигурированию имеющихся модулей. В приведенной многослойной архитектуре каркаса элементы GUI составляют самый внешний слой, имеют непосредственный доступ к прикладным данным и могут использовать средства стандартных графических библиотек. Например, популярные GUI библиотеки, такие как Qt, MFC, BCG, предоставляют средства для визуализации проектного плана в виде диаграммы Ганта, построения графиков и диаграмм общего вида, отображения календарей и других данных, характерных для календарно-сетевого планирования.

Для поддержки приложением требуемых функций управления прикладными данными достаточно воспользоваться пакетом Project Data, а для решения соответствующих задач составления расписаний — пакетами Reductions, Mathematics и Solvers. Поскольку они предоставляют все необходимые классы для

задания условий задач проектного планирования в расширенных постановках и их решения, то реализация данных функций сводится к использованию классов прикладных данных и решателей. Если известны математические особенности прикладной задачи, а к ее решению предъявляются повышенные требования эффективности, то можно сконфигурировать классы решателей соответствующими целевыми функциями и эвристиками, реализации которых также включены в состав каркаса. При этом следует учесть, что выбор эвристик и порядок их применения во многом диктуется целевой функцией оптимизационной задачи.

Таким образом, разработка типового приложения календарно-сетевого планирования с функциями управления данными и решения задач проектного планирования RCPSP в расширенных постановках требует относительно низких затрат, обусловленных, главным образом, следующими работами:

- разработка GUI целевого приложения на основе графических библиотек общего назначения;
- использование имеющихся классов прикладных данных для задания условий задачи планирования;
- использование и конфигурирование имеющих классов решателей соответствующими целевыми функциями и эвристиками для эффективного приближенного решения поставленной задачи планирования.

В других, более специальных случаях целевых приложений могут потребоваться дополнительные усилия, связанные с выполнением некоторых из перечисленных ниже работ:

- развитие пакета классов Project Data для редукции задач теории расписаний к постановке RCPSP;
- развитие пакета классов Project Data для представления условий задач RCPSP в расширенных постановках;
- развитие пакета классов Reductions для редукции прикладной задачи к математической постановке условной оптимизации GCPSP;

- развитие пакета классов Solvers для реализации новых точных и приближенных алгоритмов, а также новых эвристик для них.

Обсудим перечисленные возможности разработки целевых приложений теории расписаний в результате развития и конфигурирования классов каркаса.

3. 4. 1. Развитие пакета Project Data для редукции задач теории расписаний к постановке RCPSP

Обычно задачи теории расписаний не формулируются как задачи проектного планирования, хотя в большинстве случаев могут быть проинтерпретированы в их терминах или сведены к ним за полиномиальное время. В подобных случаях требуется разработка специальных классов прикладных данных, выражающих соответствующие понятия рассматриваемой предметной области и реализующих их, например, с помощью классов пакета Applications.

В качестве примера рассмотрим задачу составления школьного расписания. Будем считать, что в школе занятия проводятся в соответствии с единым регулярным расписанием уроков и перемен, повторяющимся каждый день с понедельника по пятницу. Исключение составляют дни каникул и праздничные дни. В терминах проектного планирования временной аспект функционирования школы описывается с помощью понятия рабочего календаря. Поскольку каркас предоставляет обобщённую реализацию рабочего календаря с регулярными и исключительными правилами, для представления школьного календаря можно воспользоваться соответствующим классом Calendar. Школьный календарь может быть реализован в целевом приложении с помощью конструирования соответствующего объекта и инициализации его атрибутов. Альтернативная реализация состоит в определении класса-наследника и в уточнении рабочих интервалов, регулярных правил и исключений непосредственно в его конструкторе.

У каждого класса учеников есть свой предопределённый образовательными стандартами план занятий на год и на каждый семестр. В терминах проектного

планирования циклы занятий представляются вложенными структурами работ, а каждое занятие или урок в отдельности — активностью с продолжительностью 45 минут. При реализации целевого приложения можно воспользоваться классами WBS и Activity, входящими в состав пакета каркаса Project Data, а в целевом приложении построить годовой план школьных занятий. Для упрощенного задания циклов занятий по отдельным предметам и количествам учебных часов можно определить специальные классы-наследники Cycle и Lesson и реализовать в них необходимые вспомогательные методы.

Для проведения занятия в определенное время необходимо, чтобы были свободны ученики класса, а также были доступны учитель и учебный класс. В терминах задачи RCPSP ученики класса, учитель и учебный класс следует рассматривать как возобновимые ресурсы соответствующих типов с доступным единичным количеством. При этом учитель обычно проводит весь цикл занятий для одного класса по одному из своих предметов. А занятия по некоторым предметам требуют специально оборудованных классов. Данные условия в рамках задачи RCPSP задаются путем назначения индивидуальных ресурсов на соответствующие работы. Обычно для организации циклов занятий по отдельным предметам необходимо предусмотреть временные интервалы между ними, например, для подготовки домашних занятий или отдыха учащихся. Данные условия естественным образом интерпретируются в терминах отношений предшествования между работами с соответствующими задержками. Для этих целей может быть определен класс Rule, который наследуя класс каркаса Link, уточняет способ параметризации данных условий.

Таким образом, рассмотренный пример задачи составления школьного расписания может быть сведён к задаче проектного планирования, а разработка программного приложения — к непосредственному использованию классов каркаса или к их наследованию с доопределением вспомогательных методов инициализации прикладных данных. В первом случае может оказаться полезным использование алиасов или предкомпиляторных директив для переименования классов каркаса. Тогда разработка GUI целевого приложения может

осуществляться в терминах предметной области, а не проектного планирования. Вместо классов Project, WBS, Activity, Resource, Link можно использовать более естественные для целевого приложения названия School, Cycle, Lesson, Teacher, Room, Class, Rule. Во втором случае реализуются новые предметно-ориентированные классы путем наследования от классов каркаса или в результате их использования с определением нового интерфейса, характерного для условий решаемой прикладной задачи.

3. 4. 2. Развитие пакета Project Data для представления условий задач RCPSP в расширенных постановках

Хотя пакет каркаса Project Data предоставляет довольно развитый набор классов прикладных данных для задания условий RCPSP задач в расширенных постановках, вполне допустимы ситуации, когда требуется еще расширить данный набор, например, для поддержки новых моделей исполнения работ, привлечения ресурсов, специфических типов ограничений и т.п. С математической точки зрения класс задач остается неизменным, но появляется дополнительная специфика, связанная с новыми типами целевых функций и наложенных ограничений. В главе 2 обосновывается возможность математической постановки и решения, так называемых, задач GCPSP с целевыми функциями и наложенными алгебраическими ограничениями общего характера.

В соответствии с объектно-ориентированным подходом развитие пакета Project Data может осуществляться различными способами. Например, новые классы прикладных данных можно наследовать от существующих классов с определением новых свойств объектов и уточнением способов параметризации условий прикладных задач. Можно создавать новые классы прикладных данных, определяя их свойства и устанавливая отношения ассоциации, композиции, агрегации с существующими классами каркаса. В отличие от простого переименования классов, упоминаемого в предыдущем разделе, становится возможной реализация обобщённых моделей прикладных данных для задания специальных условий RCPSP задач в расширенных постановках.

В качестве примера, иллюстрирующего необходимость развития пакета Project Data, приведем приложение визуального пространственно-временного моделирования проектов. В отличие от традиционных информационных систем календарно-сетевого планирования и управления проектами, составление расписаний в подобных приложениях осуществляется также с учетом пространственных ограничений, регламентирующих условия выполнимости планируемых работ на проектной площадке. Для поддержки пространственных ограничений потребуются дополнительные классы, чтобы представить проектные данные в виде трехмерных геометрических моделей, связать их с проектными работами или порождаемыми ими событиями, а также определить характер пространственных коллизий для составления согласованных расписаний. Примечательно, что реализация классов для задания других видов ограничений, таких как временные условия, отношения предшествования, ресурсные лимиты, рабочие календари, уже предусмотрена типовой конфигурацией пакета Project Data и не потребует дополнительных усилий. Поэтому развитие каркаса для обсуждаемых случаев не представляется сложным.

3. 4. 3. Развитие пакета Reductions для редукции прикладных задач к постановке GCPSP

Рассмотренные выше способы развития пакета Project Data сами по себе не влияют на ход составления расписания алгоритмами, реализации которых уже включены в состав каркаса. Чтобы учесть новые условия задачи проектного планирования, необходимо реализовать соответствующие классы пакета Reductions, с помощью которых данные условия могут быть выражены и проинтерпретированы в математически нейтральных терминах постановки условной оптимизации GCPSP. Программисту, прежде всего, требуется решить, порождают ли введенные или производные прикладные данные новые переменные, меняют ли они вид целевой функции, а также приводят ли они к новым типам алгебраических ограничений. Если это имеет место, то требуется реализовать заново или унаследовать классы, реализующие интерфейсы Variable,

ValueDomain, Objective, Constraint, и соответствующим образом модифицировать тело методов формирования условий математической задачи OptimizationProblem. С учетом того, что пакет Reductions содержит реализации всех классов, необходимых для математически корректной редукции задач RCPSP к постановке GCPSP, разработка нескольких дополнительных классов близкой функциональности не потребует значительных усилий.

3. 4. 4. Развитие пакета Solvers для реализации новых алгоритмов и эвристик

Пакет Solvers предоставляет готовые к использованию реализации точного и приближенного алгоритмов решения задач GCPSP. В главе 2 формулируются достаточные условия существования решения в данной постановке, а также эквивалентность обобщённого приближенного алгоритма популярному алгоритму последовательной диспетчеризации в случае классической постановки RCPSP. Однако каркас не предоставляет средств анализа существования решений в случае произвольно заданных целевых функций и систем ограничений, а также средств оценки качества найденных приближенных решений. Отчасти вторая проблема нивелируется возможностью поиска точного решения, по крайней мере, для задач низкой размерности. Однако вся ответственность целиком ложится на разработчиков приложений, которые должны обеспечить согласованность задаваемых условий прикладных задач и применяемых алгоритмов.

Одним из способов настройки обобщённого приближенного алгоритма является задание эвристик. Поскольку эвристики реализуются как объекты с общим интерфейсом Heuristic, то разработчики могут предоставить собственные реализации эвристик с учетом прикладных особенностей решаемых задач и сконфигурировать соответствующим образом решатель. В случаях, когда в приложении требуются новые алгоритмы, разработчики могут их реализовать, основываясь на уже имеющихся в каркасе классах математических объектов и типовых алгоритмов. Примечательно, что используя интерфейсы математических объектов, разработчики, тем самым, предоставляют обобщённые реализации алгоритмов, которые могут быть использованы при решении прикладных задач с

произвольными условиями, представимыми математическими объектами наследуемых классов. Однако вопросы конструктивности применения отдельных алгоритмов к конкретным задачам опять же относятся к компетенции разработчиков приложения.

Глава 4. Экспериментальное исследование объектно-ориентированной среды

Создание программных приложений теории расписаний представляет собой серьезную проблему, поскольку требует со стороны разработчиков как математической квалификации, необходимой для формализации прикладных задач и построения эффективных алгоритмов решения, так и знаний и опыта в области программной инженерии, необходимых для проектирования, реализации и интеграции сложных программных систем.

Подход, предложенный в предыдущей главе, предполагает создание и всестороннее применение единой программно-инструментальной среды для реализации моделей, методов и приложений теории расписаний. Данная среда сочетает в себе функции математической библиотеки и программного инструментария.

С одной стороны, такое сочетание предполагает наличие готовых к использованию программных компонентов для задания условий и решения типовых задач теории расписаний и, в частности, индустриально значимых задач высокой размерности в постановке проектного планирования GCPSP. Поскольку многие задачи теории расписаний редуцируются к данной обобщённой постановке, среда предоставляет необходимые точные и приближенные средства решения.

С другой стороны, организация инструментальной среды в виде объектно-ориентированного каркаса обеспечивает повторное использование имеющихся компонентов при программной реализации новых моделей, методов и приложений теории расписаний при относительно низких затратах на доработку. При этом возможности развития, адаптации и конфигурации компонентов не

препятствуют построению эффективных приложений, релевантных условиям и сложности решаемых прикладных задач.

Данные факторы предопределили выбор программно-инструментальной среды в качестве основного средства для построения системы визуального планирования проектов на основе ранее существовавшего приложения визуального моделирования Synchro [201]. Функции исходного приложения, главным образом, ограничивались возможностями визуализации проектных планов и календарно-сетевых графиков на диаграмме Ганта и в окнах просмотра трехмерных сцен. Также приложение предоставляло средства верификации графиков, выявления пространственно-временных конфликтов и генерации сопутствующих отчетов и видеоматериалов.

Принцип работы исходного приложения состоял в консолидации календарно-сетевых графиков, импортируемых из традиционных систем управления проектами, таких как Oracle Primavera, Microsoft Project, Asta Powerproject, и трехмерных моделей, подготовленных в популярных CAD-системах, таких как AutoCAD, Revit, Sketchup, Microstation. В результате подобной консолидации формируется единая пространственно-временная модель проекта, которая затем может использоваться для визуализации, анализа и верификации. При обнаружении пространственно-временных конфликтов календарно-сетевой график проекта может быть скорректирован средствами приложения или с использованием сторонних систем в результате экспорта и импорта проектных данных.

Отсутствие средств построения расписаний являлось принципиальным недостатком исходного приложения, поскольку любая коррекция календарно-сетевого графика при обнаружении конфликтов могла потенциально нарушить его согласованность. В этом случае требовалось экспортировать календарно-сетевой график в стороннее приложение соответствующей функциональности, строить в нем согласованное расписание, а затем импортировать график обратно. Поскольку такая последовательность не гарантировала отсутствие новых пространственно-временных конфликтов в результирующем графике,

требовались многократные действия. Реализация в составе целевого приложения на основе объектно-ориентированной среды развитых средств построения расписаний позволило превратить его в полноценную систему управления проектами, причем с возможностями многофакторного планирования и визуального анализа.

Настоящая глава посвящена оценке универсальности, эффективности и гибкости разработанной объектно-ориентированной среды на основе описанного в главе практического опыта эволюционной разработки перспективной системы визуального планирования проектов на базе существующего приложения Synchro, который может быть востребован при создании других приложений. В разделе 4. 1 описывается процесс разработки целевой системы визуального планирования проектов. Результаты вычислительных экспериментов представлены в разделе 4. 2, в котором проводится сравнение показателей эффективности построенной системы с популярными системами управления проектами.

4. 1. Разработка и развитие системы визуального планирования проектов

Разработанный каркас и связанный с ним метод инкрементальной разработки программных приложений теории расписаний были успешно применены и исследованы в ходе построения системы визуального планирования индустриальных проектов. Система строилась на основе ранее существовавшего приложения визуального моделирования с основными функциями, перечисленными во введении. На рис. 10 приведен снимок экрана, иллюстрирующий главное окно и основные элементы графического интерфейса пользователя исходного приложения, включая диаграмму Ганта и окна просмотра трехмерных сцен.

Принцип работы приложения состоял в консолидации календарно-сетевых графиков, импортируемых из традиционных систем управления проектами, таких как Oracle Primavera, Microsoft Project, Asta Powerproject, и трехмерных моделей,

подготовленных в популярных CAD-системах, таких как AutoCAD, Revit, Sketchup, Microstation. В результате подобной консолидации формируется единая пространственно-временная модель проекта, которая затем может использоваться для визуализации, анализа и верификации. При обнаружении пространственно-временных конфликтов календарно-сетевой график проекта может быть скорректирован средствами приложения или с использованием сторонних систем в результате экспорта и импорта проектных данных. Как отмечалось выше, отсутствие собственных средств построения расписаний являлось одним из главных недостатков целевого приложения, поскольку ограничивало пользователя в разрешении обнаруженных конфликтов в результате согласованной коррекции календарно-сетевого графика. Наличие подобных средств превратило целевое приложение в полноценную систему управления проектами, причем с возможностями визуального моделирования и многофакторного планирования.

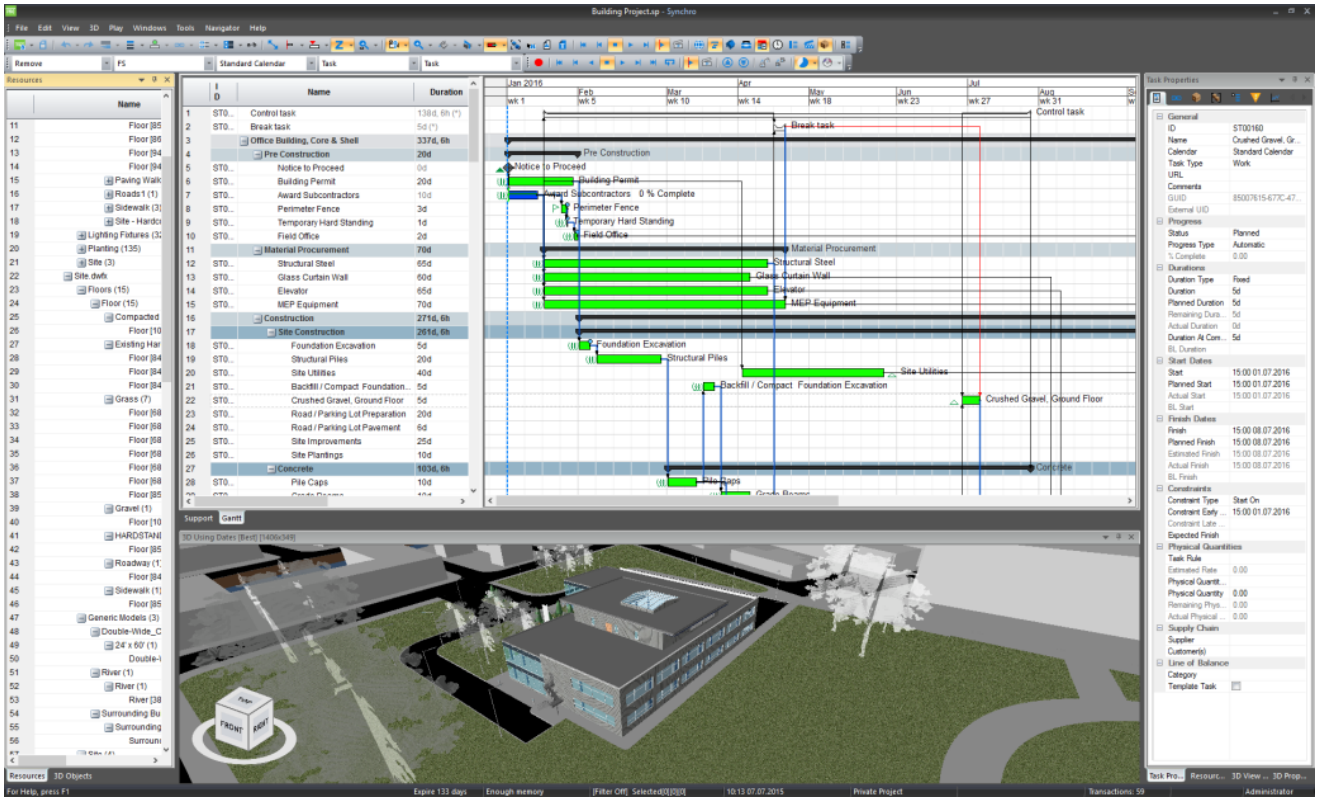

Рис. 10. Главное окно графического интерфейса пользователя исходного приложения.

Учет различных факторов осуществления проектной деятельности также являлся одним из принципиальных требований, предъявляемым к развиваемому приложению. При составлении расписаний должны учитываться не только временные условия, отношения предшествования, ресурсные ограничения и календарные правила, характерные для задач проектного планирования RCPSP, но и специфические требования пространственно-временной согласованности проектных работ, их финансового и логистического обеспечения. Данные требования важны для масштабных индустриальных программ, в которых риски технологических и организационных ошибок чрезвычайно высоки, а сроки и бюджеты жестко ограничены. Примерами специфических требований могут служить условия привлечения инвестиционных средств, ограничения по поставкам материалов, правила размещения и использования оборудования, особенности монтажа элементов конструкций возводимого сооружения, условия резервирования рабочих зон при организации проектных работ.

Заметим, что по мере развития целевой системы набор ограничений может пересматриваться в сторону обобщения и расширения, поэтому средства составления расписаний должны быть реализованы надлежащим образом, допуская поддержку новых типов ограничений без каких-либо серьезных изменений существующих программных компонентов.

Данные требования удалось удовлетворить с помощью представленного каркаса для построения приложений теории расписаний. Каркас не только предоставил готовые к использованию программные компоненты для задания условий и решения задач проектного планирования, но и обеспечил возможность добавления и поддержки новых видов ограничений в рамках описанной выше методологии.

Рассмотрим процесс эволюционной разработки целевого приложения визуального планирования индустриальных проектов более подробно.

Так как каркас внедрялся в существующее приложение, а не использовался для создания нового, возникла необходимость развития унаследованной модели данных. Заметим, что исходное приложение реализовано в рамках объектно-

ориентированной парадигмы на языке Си++ и использует объектно-ориентированную модель данных для представления проекта, работ, связей, календарей, ресурсов, трехмерных объектов сцены. Упрощенная реализация данных концептов с использованием ограниченного набора атрибутов диктовалась функциями визуального моделирования, для которых достаточно иметь данные, подлежащие отрисовке на диаграмме Ганта или в окнах просмотра трехмерных сцен. В частности, в классе работ определялись атрибуты имени, длительности, времен начала и конца, однако отсутствовали структуры индивидуального использования ресурсов. Класс календарей реализовывал простую модель рабочей недели с исключениями, но не применял более общую модель регулярных и исключительных правил. Класс ресурсов использовался, главным образом, в качестве делегата трехмерных объектов сцены, однако не предусматривал задание ресурсных атрибутов, таких как тип, статус возобновимости и количество доступных единиц.

В силу указанных причин было принято решение заменить соответствующие классы приложения каркасными реализациями. Такая замена не являлась трудоемкой и в большинстве случаев заключалась в коррекции некоторых названий методов доступа к атрибутам классов. Для поддержания прежней функциональности визуального моделирования и реализации новых функций проектного планирования, класс проектов Project, агрегирующий трехмерные данные (объекты, текстуры, камеры, анимации), был унаследован от соответствующего класса каркаса, агрегирующего необходимые проектные данные. Аналогичная схема консолидации применена в классе ресурсов, что позволило объединить данные, необходимые для ресурсного планирования, и данные, необходимые для их связи с трехмерными объектами.

При описанной схеме наследования в целевом приложении стало возможным составлять расписания непосредственно средствами каркаса. В частности, появилась возможность расчета критического пути и оценки временных резервов в постановке задачи проектного планирования без ресурсов. Также стали доступны алгоритмы составления расписаний в расширенных

постановках. Важно, что для подобных целей не потребовалась разработка дополнительных конвертеров исходных данных и результатов расчетов, поскольку алгоритмические реализации Solvers рассчитаны для работы с проектными данными при надлежащей конфигурации компонентов Project Data, Reductions и Mathematics.

Для задания дополнительных параметров планирования, выполнения алгоритмов и визуализации расчетов потребовалось расширить графический интерфейс пользователя. В частности, в окно параметров проекта добавлены элементы для установки планируемых дат начала и завершения проекта, необходимых для составления расписания и расчета временных резервов работ. В окно параметров работ добавлены элементы для установки временных ограничений, количеств используемых ресурсов, задержек между взаимосвязанными работами. В диалоге задания опций предусмотрен раздел настроек для алгоритмов составления расписаний, который позволяет, например, установить правила выравнивания дат работ, пороговые значения временных резервов для интерпретации работ как критических и субкритических, способы разрешения нарушенных ограничений в процессе актуализации проектных данных. Наконец, в панель инструментов добавлена кнопка для запуска пересчета расписания. На рис. 11 представлен экранный снимок целевого приложения с описанными развитыми функциями проектного планирования.

Заметим, что ревизии подверглись и средства визуализации результатов расчетов. В диаграмме Ганта появилась возможность отобразить критический путь, наложенные временные ограничения, а также фактические взаимосвязи, определяющие даты планируемых работ. В таблицу проектных работ (слева от диаграммы Ганта) добавлены столбцы со значениями временных резервов, которые в сочетании с универсальными средствами сортировки и фильтрации позволяют пользователю выделить относительно небольшое, но исключительно важное подмножество работ, влияющих на ход и успешное завершение всего проекта к планируемой дате. Данная возможность особенно важна для управления масштабными индустриальными программами, в которых детальный

контроль выполнения каждой индивидуальной работы не представляется возможным.

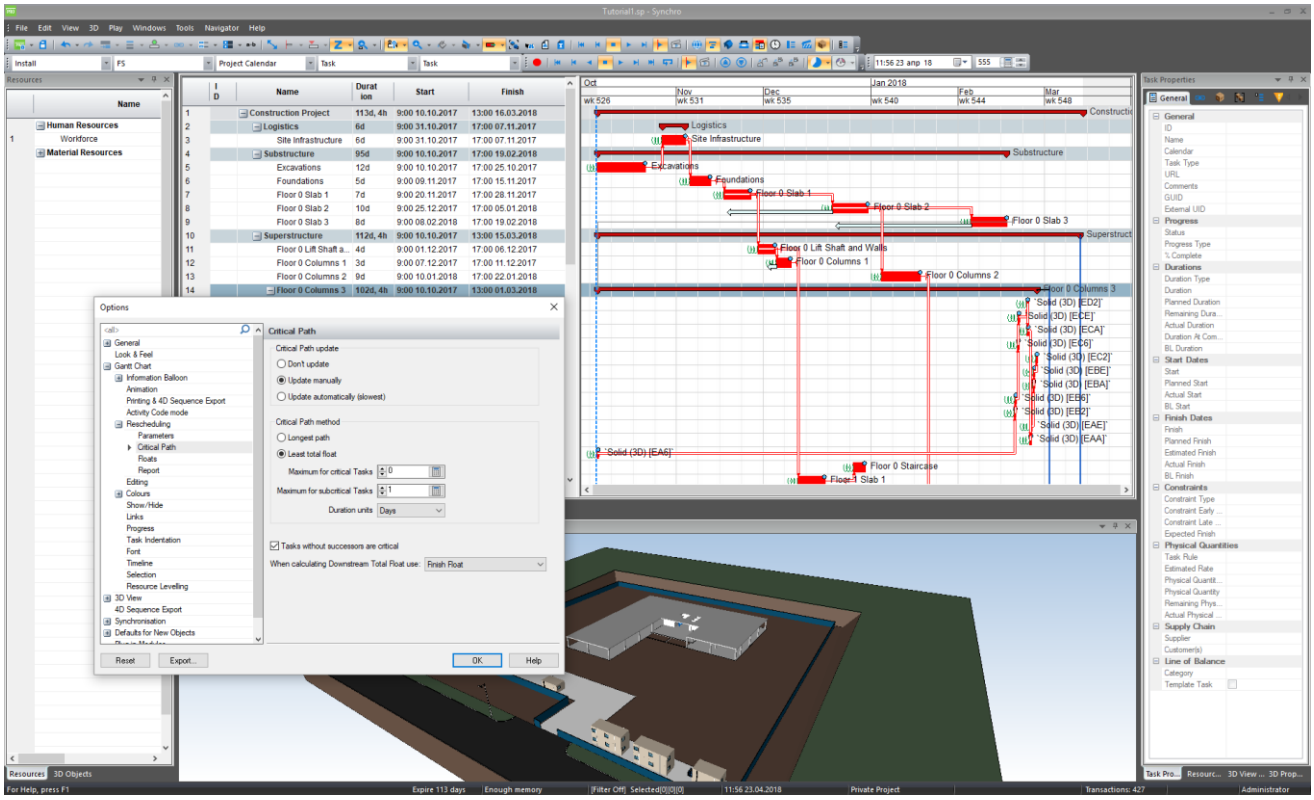

Рис. 11. Главное окно графического интерфейса пользователя целевого приложения с развитыми возможностями проектного планирования.

Поскольку планирование ресурсов является одним из ключевых элементов проектного управления, в графический интерфейс пользователя были добавлены графики потребления ресурсов, которые позволяют пользователю идентифицировать конфликты в календарно-сетевом графике, связанные с превышением доступного количества единиц того или иного ресурса. Сгенерированные приложением расписания являются согласованными в том смысле, что при корректно заданных условиях наложенные ограничения, в том числе и ресурсные, оказываются удовлетворенными. Однако при задании или коррекции календарно-сетевого графика непосредственно пользователем часть ограничений может быть нарушена и необходимы средства их идентификации. На рис. 12 приведён пример графиков потребления ресурсов по времени, причем слева представлены кривые, устанавливающие превышение доступного

количества ресурсов в графике, а справа — кривые полностью согласованного графика.

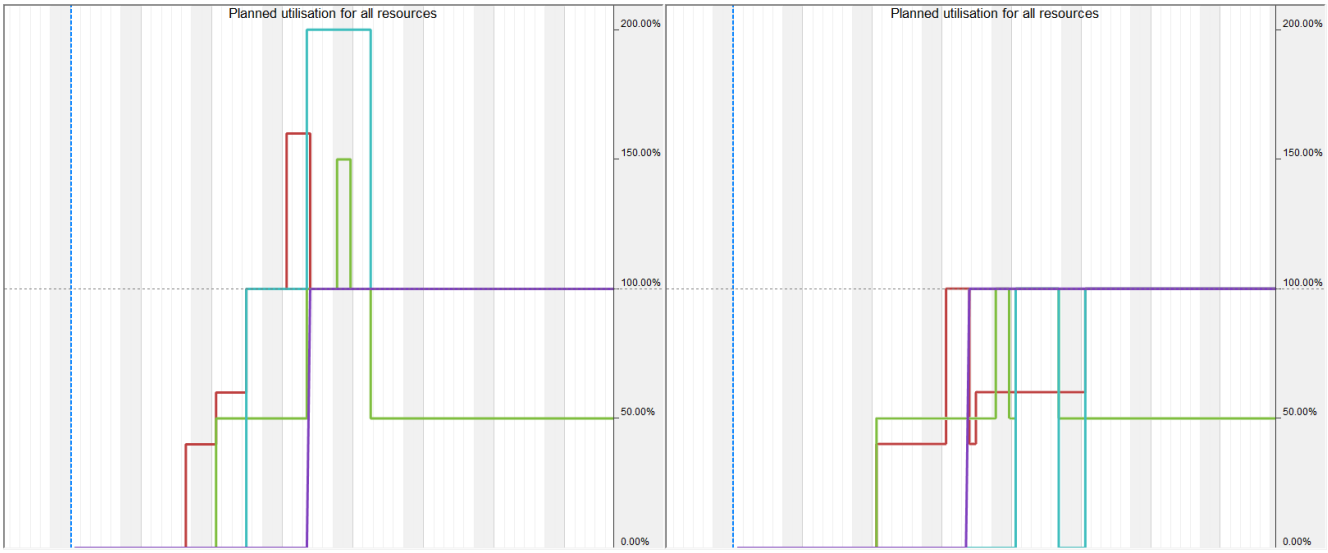

Рис. 12. Примеры графиков потребления ресурсов в несогласованном (слева) и согласованном (справа) календарно-сетевом графике.

Наконец, для верификации условий задач проектного планирования и генерации отчетов был разработан и добавлен в систему соответствующий инструмент. На рис. 13 показано окно для просмотра текстовых отчетов, в которое выводятся общие статистические данные о проекте (количество работ, ресурсов, взаимосвязей, календарей и т.п.), параметры проектного планирования, а также некорректно заданные условия, связанные с наличием циклических взаимосвязей между работами, превышением потребляемых ресурсов индивидуальными работами, переопределенными временными условиями.

Перечисленные выше функции календарно-сетевого планирования являются традиционными для систем управления проектами. Использование компонентов каркаса лишь упростило и ускорило их реализацию в составе целевого приложения, которое в результате объединило в себе функционал визуального моделирования и проектного планирования.

Вместе с тем, с консолидацией проектных данных появились качественно новые возможности для решения задач планирования в расширенных постановках с учетом пространственных факторов. Такими факторами могут быть рабочие

зоны, которые в случае пересечения или перегруженности порождают конфликты на проектной площадке и мешают осуществлению проектных работ [202]. Другими пространственными факторами, влияющими на реализуемость графика, могут быть коллизии размещаемого оборудования, конфликтность планируемых путей доставки материалов, отсутствие опор или подвесов при монтаже конструкций [2]. Математическая формализация задачи проектного планирования с учетом перечисленных пространственных факторов, а также метод решения приводились в главе 2.

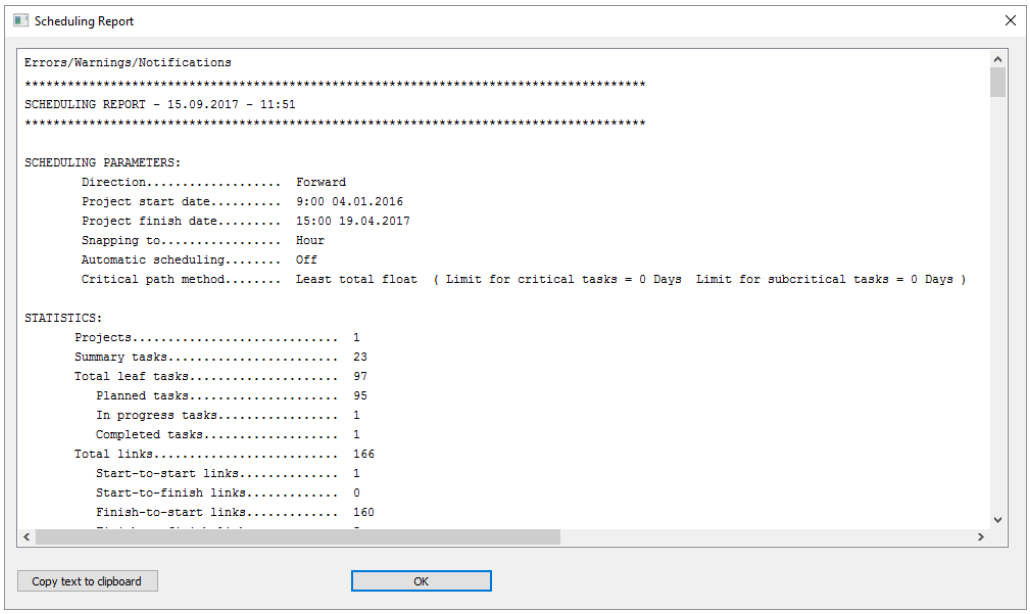

Рис. 13. Пример отчёта о построении календарно-сетевого графика.

Важно, что данные пространственные факторы должны учитываться вместе с другими видами ограничений в рамках общей вычислительной схемы составления расписания. Введенный класс задач проектного планирования, а также применяемая схема последовательной диспетчеризации позволяют осуществить это в результате задания специальных условий и приведения их к обобщённой математической постановке GCPSP. С этой целью в пакет каркаса Project Data были добавлены необходимые классы представления рабочих зон и трехмерных объектов, а в реализации класса Project учтены новые пространственные условия. Для данных условий были разработаны и включены в пакет Reductions соответствующие классы приведения к алгебраическим

ограничениям общего вида в рамках постановки GCPSP. Для приоритезации работ с назначенными рабочими зонами была разработана новая эвристика, учитывающая требуемый объем рабочей зоны, а ее реализация включена в состав пакета Solvers. Заметим, что для подобных целей можно было бы использовать и стандартную эвристику, основанную на уровнях потребления ресурсов, однако интерпретация объема рабочей зоны в качестве ресурса не всегда корректна.

Рассматривались возможности использования других вычислительных схем и алгоритмов проектного планирования, в частности, параллельной схемы генерации расписаний, одно- и много проходных схем, схем сэмплирования [172]. Несмотря на их поддержку каркасом, в целевой системе был использован компонент, реализующий алгоритм последовательной диспетчеризации с расширенным набором эвристик. В силу невысокой вычислительной сложности данный алгоритм хорошо себя зарекомендовал при решении широкого класса задач проектного планирования высокой размерности, что являлось принципиальным требованием к разрабатываемой целевой системе. Поддержка возможностей выбора специализированных алгоритмов для решения частных классов задач была принята нецелесообразной.

Заметим, что использование объектно-ориентированного каркаса значительно упростило развитие приложения. Сам каркас предоставил готовые классы проектных данных и решатели задач построения расписаний в обобщённой постановке GCPSP. Адаптация каркаса к соответствующим прикладным задачам свелась лишь к реализации классов в пакете Reductions для поддержки специальных типов условий, возникающих в задачах проектного планирования. Трудозатраты (в строках программного кода) реализации данных классов приведены в таб. 1.

Таким образом, реализация каждого из типов ограничений заключалась в создании одного нового класса, описываемого в среднем 530 строками программного кода, что составляет 0,39% от общего количества строк кода инструментальной среды.

Таб. 1. Оценка трудозатрат развития целевого приложения.

Тип ограничения	**Кол-во строк кода**	**Доля от общего кол-ва строк**
Учёт предшествования работ	211	0,16%
Учёт календарей	374	0,28%
Явные временные ограничения	170	0,13%
Обязательные временные ограничения	153	0,11%
Выравнивание начала или конца работ	97	0,07%
Учёт использования ресурсов	669	0,49%
Учёт использования рабочих пространств	2033	1,5%

4. 2. Сравнительный анализ производительности системы

Достигнутая общность в программной реализации алгоритмов составления расписаний, а также принятая параметризация условий задач проектного планирования в расширенных постановках могут существенно влиять на производительность целевой системы. Поэтому важным было убедиться, что негативный эффект от обобщённой реализации программных компонентов каркаса не столь существенен и разработанное приложение визуального планирования может конкурировать с популярными системами управления проектами при решении индустриально значимых задач высокой размерности.

Оценка производительности систем производилась по критерию затраченного процессорного времени на составление расписания в зависимости от размерности задачи (числа переменных и ограничений). В качестве тестов использовались синтезированные наборы проектных данных с разным количеством работ и связей между ними. Поскольку синтезированные планы имели фиксированные соотношения глубины и кратности связей, размерность задач в каждом тестовом наборе определялась лишь числом работ. Планы реальных индустриальных проектов использовались лишь для подтверждения полученных результатов. Тестирование проводилось на одном и том же

компьютере с типовой конфигурацией (процессор: Intel Core i7, количество ядер: 4, объём оперативной памяти: 16 ГБ, операционная система: MS Windows 10 x64), поэтому для проводимого сравнительного анализа систем значение имели лишь относительные показатели производительности.

Первая серия испытаний предназначалась для анализа производительности систем в зависимости от размерности задач планирования в постановке расчета критического пути (CPM). Для каждого теста в наборе составлялось расписание с помощью пяти различных систем, включая целевое приложение, и замерялось затраченное процессорное время. Поскольку используемые в тестах системы управления проектами являются коммерческими, и их упоминание в результатах сравнительного анализа запрещается в соответствии с лицензионными соглашениями, мы присвоили данным системам символические имена A, B, C, D. Результаты испытаний качественно не менялись при переходе к новому тестовому набору, поэтому на рис. 14 приведены типовые графики зависимости процессорного времени от количества работ в проектном плане.

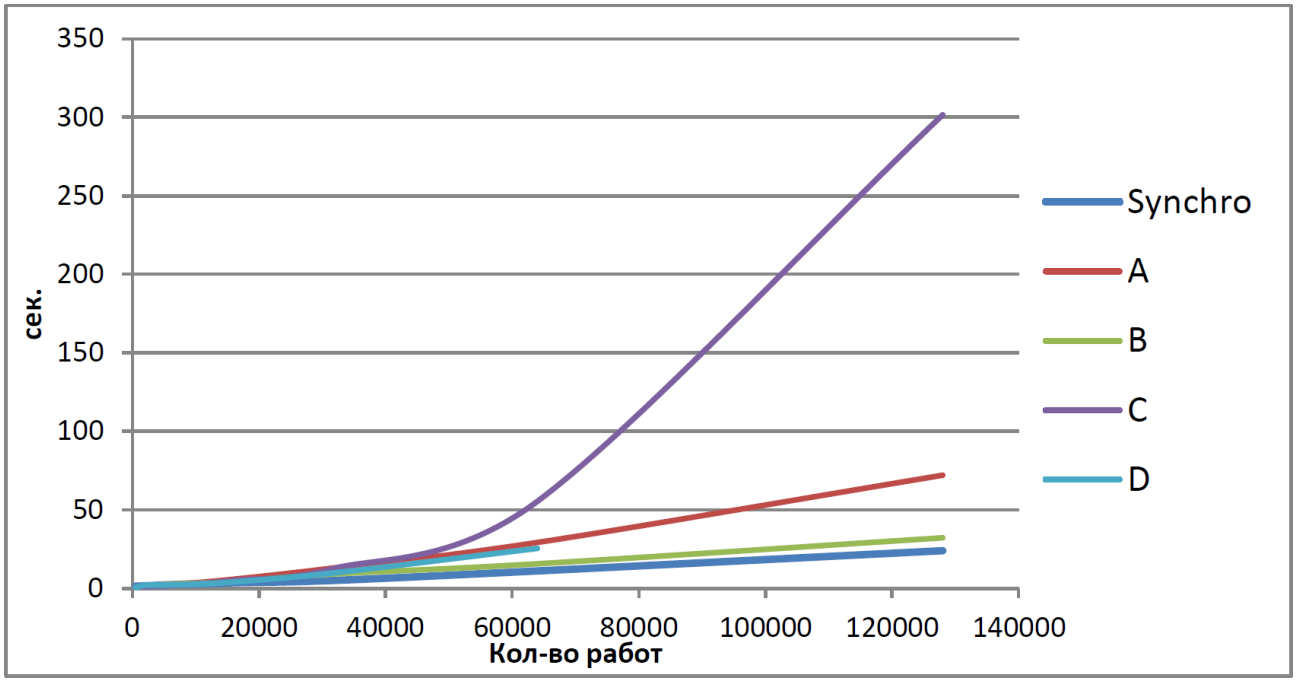

Рис. 14. Зависимость процессорного времени составления расписания в постановке CPM от размерности задачи.

Заметим, что кривая производительности для системы D обрывается при достижении размерности тестовых данных в 64000 работ. Это связано с невозможностью данной системы обрабатывать проектные планы большей размерности. С ростом размерности задач планирования целевое приложение демонстрировало наилучшие результаты.

Вторая серия испытаний предназначалась для аналогичных целей, но расписание строилось в постановке проектного планирования с учетом ресурсов RCPSP. Поскольку системы реализуют приближенные алгоритмы, имеющие разную вычислительную сложность и обеспечивающие разное качество найденных решений, сравнительный анализ производительности здесь не всегда корректен.

На рис. 15 представлены характерные результаты сравнения производительности популярных систем управления проектами. Системы C и D не участвовали в данном испытании по причине отсутствия в них соответствующих алгоритмических средств. Более тщательный анализ показывает, что алгоритм, реализованный в системе A, демонстрирует линейную сложность от размерности задачи, а в двух других системах, включая целевую — квадратичную. Это объясняется тем, что первый алгоритм строит согласованное расписание путем решения задачи в постановке расчета критического пути, а затем — последовательно выравнивает уровни потребления ресурсов. Это может быть выполнено за линейное время, однако качество найденного решения в общем случае будет хуже вследствие возможного увеличения длительности проектов за счет выравнивания ресурсов. Более качественный результат (меньшая длительность планируемого проекта) достигается при поиске оптимального расписания приближенными методами, например, с помощью реализованного в объектно-ориентированном каркасе классического алгоритма последовательной диспетчеризации, который имеет квадратичную сложность, поскольку выбор приоритетной работы на каждом шаге требует применения эвристик для всех работ в активном списке. По сравнению с системой B, также реализующей

алгоритм поиска расписания квадратичной сложности, разработанная система демонстрирует лучшую производительность.

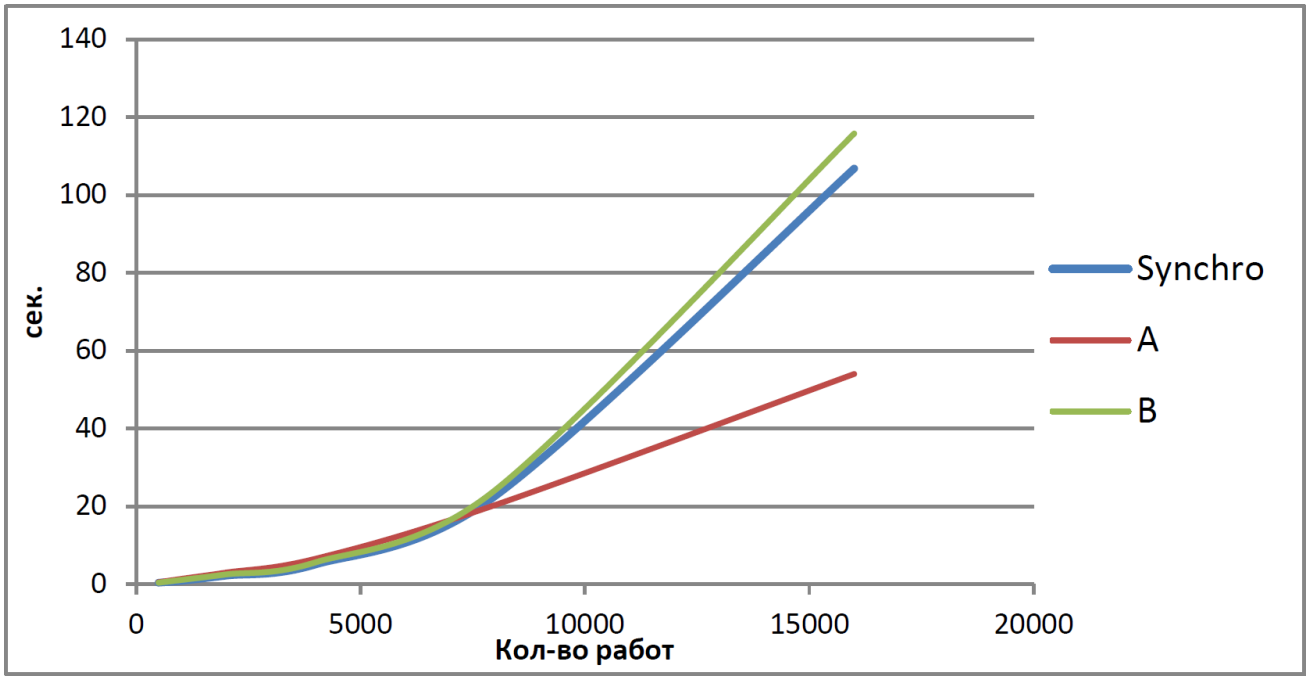

Рис. 15. Зависимость времени поиска расписания от размерности плана (с учётом ресурсов).

Тем самым, предоставляя широкие функциональные возможности, разработанное целевое приложение не уступает по производительности популярным системам управления проектами при решении задач планирования в аналогичных постановках и аналогичными методами.

Заключение

Основные результаты диссертационной работы состоят в следующем:

- предложен и математически формализован класс задач обобщённого проектного планирования GCPSP, охватывающий разнообразные задачи теории расписаний и проектного планирования в расширенных постановках. GCPSP задача ставится как оптимизационная задача на множестве решений, локально согласованных с эквивалентной системой ограничений с приоритетами;
- доказаны конструктивные теоремы о существовании решения задач в классе GCPSP, о сводимости классических постановок теории расписаний к задачам данного класса, а также о возможности их точного и приближённого решения на основе предложенного обобщённого алгоритма;
- на языке Си++ спроектирована и разработана объектно-ориентированная среда, предусматривающая развитые инструментальные возможности для программной реализации моделей, методов и приложений теории расписаний, а также предоставляющая эффективные средства решения индустриальных задач высокой размерности;
- разработан метод построения и инкрементального развития приложений теории расписаний на основе предлагаемой объектно-ориентированной среды;
- разработанная среда и предложенный метод успешно прошли экспериментальное исследование в ходе создания, сопровождения и развития коммерческой системы визуального моделирования и планирования проектов, используемой в 36 странах мира.

Список литературы

[1] Semenov V.A., Anichkin A.S., Morozov S.V., Tarlapan O.A., Zolotov V.A. Visual Planning and Scheduling of Industrial Projects with Spatial Factors. // Proceedings of the 20th ISPE International Conference on Concurrent Engineering (под ред. Bil C., Mo J., Stjepandić J.), Мельбурн, Австралия, 2013, стр. 343–352.

[2] Семенов В.А., Аничкин А.С., Морозов С.В., Тарлапан О.А., Золотов В.А. Комплексный метод составления расписаний для сложных индустриальных программ с учетом пространственно-временных ограничений. // Труды ИСП РАН (под ред. В. П. Иванникова), том 26, вып. 1, 2014 г., стр. 457–482, DOI: 10.15514/ISPRAS-2014-26(1)-20.

[3] Аничкин А.С., Семенов В.А. Современные модели и методы теории расписаний. // Труды ИСП РАН (под ред. В. П. Иванникова), том 26, вып. 3, 2014 г., стр. 5–50, DOI: 10.15514/ISPRAS-2014-26(3)-1.

[4] Аничкин А.С., Семенов В.А. Математическая формализация задач проектного планирования в расширенной постановке. // Труды ИСП РАН, том 29, вып. 2, 2017 г., стр. 231–256, DOI: 10.15514/ISPRAS-2017-29(2)-9.

[5] Аничкин А.С., Семенов В.А. Объектно-ориентированный каркас для программной реализации приложений теории расписаний. // Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 247–296, DOI: 10.15514/ISPRAS-2017-29(3)-14.

[6] Аничкин А.С., Морозов С.В., Семенов В.А., Тарлапан О.А. Эволюционная разработка системы визуального планирования проектов на основе объектно-ориентированного каркаса. // Труды ИСП РАН, том 29, вып. 5, 2017 г., стр. 239–256, DOI: 10.15514/ISPRAS-2017-29(5)-12.

[7] Аничкин А.С., Семенов В.А. Объектно-ориентированный каркас для разработки приложений теории расписания. // Информационные технологии в науке, образовании и управлении: труды международной конференции IT + S&E`15 (под ред. Глориозова Е.Л.), весенняя сессия, 22 мая – 01 июня 2015 г., Ялта-Гурзуф, Крым, Россия, стр. 460–464.

[8] Аничкин А.С., Семенов В.А. Об обобщённой математической постановке задач проектного планирования. // ИТНОУ: Информационные технологии в науке, образовании и управлении (под ред. Глориозова Е.Л.), вып. 2, 2017 г., стр. 74–86.

[9] Теория расписаний [Электронный ресурс] // Википедия. URL: https://ru.wikipedia.org/wiki/Теория_расписаний

[10] Лазарев А.А., Гафаров Е.Р. Теория расписаний. Задачи и алгоритмы. // МГУ им. М. В. Ломоносова, Москва, Россия, 2011 г., 222 с.

[11] Коваленко Ю.В. Сложность некоторых задач теории расписаний и эволюционные алгоритмы их решения. // Диссертационная работа на соискание учёной степени кандидата физико-математических наук, Омск, Россия, 2013 г., 129 с.

[12] Sprecher A. Resource-constrained project scheduling: Exact methods for the multi-mode case. Том 409. // Springer, Берлин, Германия, 1994 г., 142 с., Серия книг "Lecture Notes in Economics and Mathematical Systems".

[13] Задача об упаковке в контейнеры [Электронный ресурс] // Википедия. URL: https://ru.wikipedia.org/wiki/Задача_об_упаковке_в_контейнеры

[14] Задача о рюкзаке [Электронный ресурс] // Википедия. URL: https://ru.wikipedia.org/wiki/Задача_о_рюкзаке

[15] Stadtler H. Multilevel capacitated lot-sizing and resource-constrained project scheduling: An integrating perspective. // International Journal of Production Research, том 43, вып. 24, 2005 г., стр. 5253–5270.

[16] Задача коммивояжёра [Электронный ресурс] // Википедия. URL: https://

ru.wikipedia.org/wiki/Задача_коммивояжёра

[17] Brucker P., Knust S. Resource-constrained project scheduling and timetabling. // Lecture Notes in Computer Science, том 2079, 2001 г., стр. 277–293.

[18] Drexl A., Salewski F. Distribution requirements and compactness constraints in school timetabling. // European Journal of Operational Research, том 102, вып. 1, 1997 г., стр. 193–214.

[19] Pritsker A.A.B., Watters L.J., Wolfe P.M. Multiproject scheduling with limited resources: A zero-one programming approach. // Management Science, том 16, вып. 1, 1969 г., стр. 93–107.

[20] Blazewicz J., Lenstra J.K., Rinnooy K.A.H.G. Scheduling subject to resource constraints: Classification and complexity. // Discrete Applied Mathematics, том 5, 1983 г., стр. 11–24.

[21] Ahuja H.N. Construction performance control by networks. // John Wiley & Sons, Нью-Йорк, США, 1976 г., 636 с.

[22] Davis E.W., Patterson J.H. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. // Management Science, том 21, вып. 8, 1975 г., стр. 944–955.

[23] Hegazy T. Optimization of resource allocation and leveling using genetic algorithms. // Journal of Construction Engineering and Management, том 126, вып. 3, 1999 г., стр. 167–175.

[24] Brooks G.N., White C.R. An algorithm for finding optimal or near optimal solutions to the production scheduling problem. // Journal of Industrial Engineering, том 17, вып. 2, 1966 г., стр. 173–186.

[25] Gomory R.E. An all integer programming algorithm. // Из книги: "Industrial Scheduling.", Prentice-Hall, Нью-Джесси, США, 1963 г., стр. 193–206.

[26] Szwarg W. Solutions of the akers-friedman scheduling problem. // Operations Research, том 8, вып. 6, 1960 г., стр. 782–788.

[27] Ashour S. A Decomposition Approach for the Machine Scheduling Problem. //

University of Iowa, Айова-сити, США, 1967 г., 416 с.

[28] Giffler B., Thompson G.L., Van N.V. Numerical experience with the linear and Monte-Carlo algorithms for solving production scheduling problems. // Из книги: "Industrial Scheduling.", Prentice-Hall, Нью-Джесси, США, 1963 г., стр. 21–29.

[29] Page E.S. An approach to the scheduling of the N jobs on M machines. // Journal of the Royal Statistical Society, том 23, вып. 2, 1961 г., стр. 484–492.

[30] Kolisch R. Efficient priority rules for the resource-constrained project scheduling problem. // Journal of Operations Management, том 14, вып. 3, 1996 г., стр. 179–192.

[31] Boctor F.F. Some efficient multi-heuristic procedures for resource constrained project scheduling. // European Journal of Operational Research, том 49, вып. 1, 1990 г., стр. 3–13.

[32] Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy K.A.H.G. Optimization and approximation in deterministic sequencing and scheduling: A survey. // Annals of Discrete Mathematics, том 5, 1979 г., стр. 287–326.

[33] Brucker P., Drexl A., Möhring R., Neumann K., Pesch E. Resource-constrained project scheduling: Notation, classification, models, and methods. // European Journal of Operational Research, том 112, вып. 1, 1999 г., стр. 3–41.

[34] Słowin´ski R. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. // European Journal of Operational Research, том 7, вып. 3, 1981 г., стр. 265–273.

[35] Weglarz J. On certain models of resource allocation problems. // Kybernetes, том 9, вып. 1, 1980 г., стр. 61–66.

[36] Böttcher J., Drexl A., Kolisch R., Salewski F. Project scheduling under partially renewable resource constraints. // Management Science, том 45, вып. 4, 1999 г., стр. 543–559.

[37] Zhu G., Bard J.F., Yu G. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. // INFORMS Journal on Computing, том

18, вып. 3, 2006 г., стр. 377–390.

[38] Nudtasomboon N., Randhawa S.U. Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs. // Computers and Industrial Engineering, том 32, вып. 1, 1997 г., стр. 227–242.

[39] Neumann K., Schwindt C. Project scheduling with inventory constraints. // Mathematical Methods of Operations Research, том 56, вып. 3, 2003 г., стр. 513–533.

[40] Bartels J.H., Zimmermann J. Scheduling tests in automotive R&D projects. // European Journal of Operational Research, том 193, вып. 3, 2009 г., стр. 805–819.

[41] Schwindt C., Trautmann N. Batch scheduling in process industries: An application of resource-constrained project scheduling. // OR Spectrum, том 22, вып. 4, 2000 г., стр. 501–524.

[42] Neumann K., Schwindt C., Trautmann N. Scheduling of continuous and discontinuous material flows with intermediate storage restrictions. // European Journal of Operational Research, том 165, вып. 2, 2005 г., стр. 495–509.

[43] Weglarz J., Blazewicz J., Cellary W., Słowin´ski R. Algorithm 520: An automatic revised simplex method for constrained resource network scheduling. // ACM Transactions on Mathematical Software, том 3, вып. 3, 1977 г., стр. 295–300.

[44] Blazewicz J., Ecker K., Pesch E., Schmidt G., Weglarz J. Handbook on Scheduling: From Theory to Applications. // Springer, Берлин, Германия, 2007 г., 647 с., Серия книг "International Handbooks on Information Systems".

[45] Kis T. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. // Mathematical Programming, том 103, вып. 3, 2005 г., стр. 515–539.

[46] Waligora G. Discrete-continuous project scheduling with discounted cash flows – A tabu search approach. // Computers and Operations Research, том 35, вып. 7, 2008 г., стр. 2141–2153.

[47] Kis T. RCPS with Variable intensity activities and feeding precedence constraints. // Из книги: "Perspectives in Modern Project Scheduling", Springer Science & Business Media, Нью-Йорк, США, 2006 г., стр. 105–129.

[48] Jozefowska J., Mika M., Rozycki R., Waligora G., Weglarz J. Solving the discrete-continuous project scheduling problem via its discretization. // Mathematical Methods of Operations Research, том 52, вып. 3, 2000 г., стр. 489–499.

[49] Weglarz J. Project scheduling with continuously divisible, doubly-constrained resources. // Management Science, том 27, вып. 9, 1981 г., стр. 1040–1053.

[50] Bianco L., Dell'olmo P., Speranza M.G. Heuristics for multimode scheduling problems with dedicated resources. // European Journal of Operational Research, том 107, вып. 2, 1998 г., стр. 260–271.

[51] Bianco L., Caramia M., Dell'olmo P. Solving a preemptive project scheduling problem with coloring techniques. // Из книги: "Project Scheduling: Recent Models, Algorithms and Applications.", Springer Science & Business Media, Нью-Йорк, США, 1999 г., стр. 135–146.

[52] Dorndorf U., Phan-Huy T., Pesch E. A survey of interval capacity consistency tests for time- and resource constrained scheduling. // Из книги: "Project Scheduling: Recent Models, Algorithms and Applications.", Springer Science & Business Media, Нью-Йорк, США, 1999 г., стр. 213–238.

[53] Dorndorf U., Pesch E., Phan-Huy T. Constraint propagation techniques for the disjunctive scheduling problem. // Artificial Intelligence, том 122, вып. 1-2, 2000 г., стр. 189–240.

[54] Bomsdorf F., Derigs U. A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. // OR-Spectrum, том 30, вып. 4, 2008 г., стр. 751–772.

[55] Klein R., Scholl A. Scattered branch and bound – An adaptive search strategy applied to resource-constrained project scheduling. // Central European Journal of

Operations Research, том 7, 1999 г., стр. 177–201.

[56] Nonobe K., Ibaraki T. Formulation and tabu search algorithm for the resource constrained project scheduling problem. Том 15. // Из книги: "Essays and Surveys in Metaheuristics.", Springer, Берлин, Германия, 2002 г., стр. 557-588, Серия книг "Operations Research/Computer Science Interfaces Series".

[57] Pesch E. Lower bounds in different problem classes of project schedules with resource constraints. // Из книги: "Project Scheduling: Recent Models, Algorithms and Applications.", Springer Science & Business Media, Нью-Йорк, США, 1999 г., стр. 53–76.

[58] Klein R., Scholl A. Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. // European Journal of Operational Research, том 112, вып. 2, 1999 г., стр. 322–346.

[59] Klein R. Project scheduling with time-varying resource constraints. // International Journal of Production Research, том 38, вып. 16, 2000 г., стр. 3937–3952.

[60] Hartmann S. Project scheduling under limited resources: Models, methods, and applications. Том 478. // Springer, Берлин, Германия, 1999 г., 221 с., Серия книг "Lecture Notes in Economics and Mathematical Systems".

[61] Icmeli O., Rom W.O. Solving the resource constrained project scheduling problem with optimization subroutine library. // Computers and Operations Research, том 23, вып. 8, 1996 г., стр. 801–817.

[62] Bartusch M., Möhring R.H., Radermacher F.J. Scheduling project networks with resource constraints and time windows. // Annals of Operations Research, том 16, вып. 1, 1988 г., стр. 201–240.

[63] Debels D., Vanhoucke M. The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects. // Computers and Industrial Engineering, том 54, вып. 1, 2008 г., стр. 140–154.

[64] Demeulemeester E.L., Herroelen W.S. An efficient optimal solution procedure for

the preemptive resource-constrained project scheduling problem. // European Journal of Operational Research, том 90, вып. 2, 1996 г., стр. 334–348.

[65] Ballestin F., Valls V., Quintanilla S. Pre-emption in resource-constrained project scheduling. // European Journal of Operational Research, том 189, вып. 3, 2008 г., стр. 1136–1152.

[66] Buddhakulsomsiria J., Kim D.S. Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. // European Journal of Operational Research, том 175, вып. 1, 2006 г., стр. 279–295.

[67] Buddhakulsomsiria J., Kim D.S. Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. // European Journal of Operational Research, том 178, вып. 2, 2007 г., стр. 374–390.

[68] Franck B., Neumann K., Schwindt C. Project scheduling with calendars. // OR Spektrum, том 23, вып. 3, 2001 г., стр. 325–334.

[69] Drezet L.E., Billaut J.C. A project scheduling problem with labour constraints and time-dependent activities requirements. // International Journal of Production Economics, том 112, вып. 1, 2008 г., стр. 217–225.

[70] Cavalcante C.C.B., Souza C., Savelsbergh M.W.P., Wang Y., Wolsey L.A. Scheduling projects with labor constraints. // Discrete Applied Mathematics, том 112, вып. 1-3, 2001 г., стр. 27–52.

[71] Mika M., Waligora G., Weglarz J. Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. // European Journal of Operational Research, том 187, вып. 3, 2008 г., стр. 1238–1250.

[72] Mika M., Waligora G., Weglarz J. Modelling setup times in project scheduling. // Из книги: "Perspectives in Modern Project Scheduling.", Springer Science & Business Media, Нью-Йорк, США, 2006 г., стр. 131–165.

[73] Drexl A., Nissen R., Patterson J.H., Salewski F. Progen/πx – An instance

generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. // European Journal of Operational Research, том 125, вып. 1, 2000 г., стр. 59–72.

[74] Vanhoucke M. Setup times and fast tracking in resource-constrained project scheduling. // Computers and Industrial Engineering, том 54, вып. 4, 2008 г., стр. 1062–1070.

[75] Elmaghraby S.E. Activity networks: Project planning and control by network models. // Wiley, Нью-Йорк, США, 1977 г., 443 с.

[76] Alcaraz J., Maroto C., Ruiz R. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. // Journal of the Operational Research Society, том 54, вып. 6, 2003 г., стр. 614–626.

[77] Bouleimen K., Lecocq H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. // European Journal of Operational Research, том 149, вып. 2, 2003 г., стр. 268–281.

[78] Hartmann S. Project scheduling with multiple modes: A genetic algorithm. // Annals of Operations Research, том 102, вып. 1-4, 2001 г., стр. 111–135.

[79] Jarboui B., Damak N., Siarry P., Rebai A. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. // Applied Mathematics and Computation, том 195, вып. 1, 2008 г., стр. 299–308.

[80] Jozefowska J., Mika M., Rozycki R., Waligora G., Weglarz J. Simulated annealing for multi-mode resource-constrained project scheduling. // Annals of Operations Research, том 102, вып. 1-4, 2001 г., стр. 137–155.

[81] Özdamar L. A genetic algorithm approach to a general category project scheduling problem. // IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, том 29, вып. 1, 1999 г., стр. 44–59.

[82] Calhoun K.M., Deckro R.F., Moore J.T., Chrissis J.W., Hove J.C.V. Planning and

re-planning in project and production scheduling. // Omega, том 30, вып. 3, 2002 г., стр. 155–170.

[83] Reyck B., Herroelen W.S. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. // European Journal of Operational Research, том 119, вып. 2, 1999 г., стр. 538–556.

[84] Heilmann R. Resource-constrained project scheduling: A heuristic for the multi-mode case. // OR Spektrum, том 23, вып. 3, 2001 г., стр. 335–357.

[85] Heilmann R. A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. // European Journal of Operational Research, том 144, вып. 2, 2003 г., стр. 348–365.

[86] Li H., Womer K. Modeling the supply chain configuration problem with resource constraints. // International Journal of Project Management, том 26, вып. 6, 2008 г., стр. 646–654.

[87] Tiwari V., Patterson J.H., Mabert V.A. Scheduling projects with heterogeneous resources to meet time and quality objectives. // European Journal of Operational Research, том 193, вып. 3, 2009 г., стр. 780–790.

[88] Tareghian H.R., Taheri S.H. A solution procedure for the discrete time, cost and quality tradeoff problem using electromagnetic scatter search. // Applied Mathematics and Computation, том 190, вып. 2, 2007 г., стр. 1136–1145.

[89] Salewski F., Schirmer A., Drexl A. Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application. // European Journal of Operational Research, том 102, вып. 1, 1997 г., стр. 88–110.

[90] Demeulemeester E.L., Reyck B., Herroelen W.S. The discrete time/resource trade-off problem in project networks – A branch-and-bound approach. // IIE Transactions, том 32, вып. 11, 2000 г., стр. 1059–1069.

[91] Ranjbar M., Kianfar F. Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms. // Applied Mathematics and

Computation, том 191, вып. 2, 2007 г., стр. 451–456.

[92] Ranjbar M., Reyck B., Kianfar F. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. // European Journal of Operational Research, том 193, вып. 1, 2009 г., стр. 35–48.

[93] Akkan C., Drexl A., Kimms A. Network decomposition-based benchmark results for the discrete time-cost tradeoff problem. // European Journal of Operational Research, том 165, вып. 2, 2005 г., стр. 339–358.

[94] Demeulemeester E.L., Reyck B., Foubert B., Herroelen W.S., Vanhoucke M. New computational results on the discrete time/cost trade-off problem in project networks. // Journal of the Operational Research Society, том 49, вып. 11, 1998 г., стр. 1153–1163.

[95] Deckro R.F., Hebert J.E., Verdini W.A., Grimsrud P.H., Venkateshwar S. Nonlinear time/cost tradeoff models in project management. // Computers and Industrial Engineering, том 28, вып. 2, 1995 г., стр. 219–229.

[96] Chassiakos A.P., Sakellaropoulos S.P. Time-cost optimization of construction projects with generalized activity constraints. // Journal of Construction Engineering and Management, том 131, вып. 10, 2005 г., стр. 1115–1124.

[97] Klein R., Scholl A. PROGRESS: Optimally solving the generalized resource-constrained project scheduling problem. // Mathematical Methods of Operations Research, том 52, вып. 3, 2000 г., стр. 467–488.

[98] Kolisch R. Integrated scheduling, assembly area- and part-assignment for large-scale, make-to-order assemblies. // International Journal of Production Economics, том 64, вып. 1-3, 2000 г., стр. 127–141.

[99] Vanhoucke M. Work continuity constraints in project scheduling. // Journal of Construction Engineering and Management, том 132, вып. 1, 2006 г., стр. 14–25.

[100] Demeulemeester E.L., Herroelen W.S. Modelling setup times, process batches and transfer batches using activity network logic. // European Journal of Operational Research, том 89, вып. 2, 1996 г., стр. 355–365.

[101] Cesta A., Oddi A., Smith S.F. A constraint-based method for project scheduling with time windows. // Journal of Heuristics, том 8, вып. 1, 2002 г., стр. 109–136.

[102] Dorndorf U., Pesch E., Phan H.T. A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalized precedence constraints. // Management Science, том 46, вып. 10, 2000 г., стр. 1365–1384.

[103] Schwindt C. Generation of resource-constrained project scheduling problems subject to temporal constraints. // Технический отчёт WIOR-543, Universität Karlsruhe, Баден-Вюртемберг, Германия, 1998 г.

[104] Neumann K., Zimmermann J. Exact and truncated branch-and-bound procedures for resource-constrained project scheduling with discounted cash flows and general temporal constraints. // Central European Journal of Operations Research, том 10, вып. 4, 2002 г., стр. 357–380.

[105] Neumann K., Zimmermann J. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. // European Journal of Operational Research, том 127, вып. 2, 2000 г., стр. 425–443.

[106] Neumann K., Schwindt C., Zimmermann J. Recent results on resource-constrained project scheduling with time windows: Models, solution methods, and applications. // Central European Journal of Operations Research, том 10, вып. 2, 2002 г., стр. 113–148.

[107] Sabzehparvar M., Seyed-Hosseini S.M. A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. // Journal of Supercomputing, том 44, вып. 3, 2008 г., стр. 257–273.

[108] Franck B., Neumann K. Resource-constrained project scheduling with time windows: Structural questions and priority rule methods. // Технический отчёт WIOR-492, Universität Karlsruhe, Баден-Вюртемберг, Германия, 1997 г.

[109] Ballestin F., Valls V., Quintanilla S. Due dates and RCPSP. // Из книги: "Perspectives in Modern Project Scheduling.", Springer Science & Business

Media, Нбю-Йорк, США, 2006 г., стр. 79–104.

[110] Brânzei R., Ferrari G., Fragnelli V., Tijs S. Two approaches to the problem of sharing delay costs in joint projects. // Annals of Operations Research, том 109, вып. 1-4, 2002 г., стр. 359–374.

[111] Chiu H.N., Tsai D.M. An efficient search procedure for the resource-constrained multi-project scheduling problem with discounted cash flows. // Construction Management and Economics, том 20, вып. 1, 2002 г., стр. 55–66.

[112] Özdamar L., Ulusoy G., Bayyigit M. A heuristic treatment of tardiness and net present value criteria in resource constrained project scheduling. // International Journal of Physical Distribution and Logistics Management, том 28, вып. 9/10, 1998 г., стр. 805–824.

[113] Baptiste P., Pape C.L., Nuijten W. Satisfiability tests and time-bound adjustments for cumulative scheduling problems. // Annals of Operations Research, том 92, 1999 г., стр. 305–333.

[114] Yang H.H., Chen Y.L. Finding the critical path in an activity network with time-switch constraints. // European Journal of Operational Research, том 120, вып. 3, 2000 г., стр. 603–613.

[115] Vanhoucke M., Demeulemeester E.L., Herroelen W.S. Discrete time/cost trade-offs in project scheduling with time-switch constraints. // Journal of the Operational Research Society, том 53, вып. 7, 2002 г., стр. 741–751.

[116] Krüger D., Scholl A. Managing and modelling general resource transfers in (multi-)project scheduling. // OR Spektrum, том 32, вып. 2, 2010 г., стр. 369–394.

[117] Elmaghraby S.E. An algebra for the analysis of generalized activity networks. // Management Science, том 10, вып. 3, 1964 г., стр. 494–514.

[118] Belhe U., Kusiak A. Resource-constrained scheduling of hierarchically structured design activity networks. // IEEE Transactions on Engineering Management, том 42, вып. 2, 1995 г., стр. 150–158.

[119] Neumann K. Stochastic project networks: Temporal analysis, scheduling and cost minimization. Том 344. // Springer, Берлин, Германия, 1990 г., 237 с., Серия книг "Lecture Notes in Economics and Mathematical Systems".

[120] Kuster J., Jannach D. Handling airport ground processes based on resource-constrained project scheduling. // Lecture Notes in Computer Science, том 4031, 2006 г., стр. 166–176.

[121] Viana A., Sousa J. Using metaheuristics in multiobjective resource constrained project scheduling. // European Journal of Operational Research, том 120, вып. 2, 2000 г., стр. 359–374.

[122] Vanhoucke M., Demeulemeester E.L., Herroelen W.S. An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem. // Annals of Operations Research, том 102, вып. 1-4, 2001 г., стр. 179–196.

[123] Franck B., Schwindt C. Different resource-constrained project scheduling models with minimal and maximal time-lags. // Технический отчёт WIOR-450, Universität Karlsruhe, Баден-Вюртемберг, Германия, 1995 г.

[124] Lorenzoni L.L., Ahonen H., Alvarenga G.A. A multi-mode resource-constrained scheduling problem in the context of port operations. // Computers and Industrial Engineering, том 50, вып. 1-2, 2006 г., стр. 55–65.

[125] Vanhoucke M. Scheduling an R&D project with quality-dependent time slots. // Lecture Notes in Computer Science, том 3982, 2006 г., стр. 621–630.

[126] Rom W.O., Tukel O.I., Muscatello J.R. MRP in a job shop environment using a resource constrained project scheduling model. // Omega, том 30, вып. 4, 2002 г., стр. 275–286.

[127] Nazareth T., Verma S., Bhattacharya S., Bagchi A. The multiple resource constrained project scheduling problem: A breadth-first approach. // European Journal of Operational Research, том 112, вып. 2, 1999 г., стр. 347–366.

[128] Al-Fawzan M., Haouari M. A bi-objective model for robust resource-constrained project scheduling. // International Journal of Production Economics, том 96,

вып. 2, 2005 г., стр. 175–187.

[129] Abbasi B., Shadrokh S., Arkat J. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. // Applied Mathematics and Computation, том 180, вып. 1, 2006 г., стр. 146–152.

[130] Kobylanski P., Kuchta D. A note on the paper by M. A. Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling. // International Journal of Production Economics, том 107, вып. 2, 2007 г., стр. 496–501.

[131] Chtourou H., Haouari M. A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. // Computers and Industrial Engineering, том 55, вып. 1, 2008 г., стр. 183–194.

[132] Icmeli-Tukel O., Rom W.O. Ensuring quality in resource constrained project scheduling. // European Journal of Operational Research, том 103, вып. 3, 1997 г., стр. 483–496.

[133] Van S., Demeulemeester E.L., Herroelen W.S. A classification of predictive-reactive project scheduling procedures. // Journal of Scheduling, том 10, вып. 3, 2007 г., стр. 195–207.

[134] Sakkout H., Wallace M. Probe backtrack search for minimal perturbation in dynamic scheduling. // Constraints, том 5, вып. 4, 2000 г., стр. 359–388.

[135] Zhu G., Bard J.F., Yu G. Disruption management for resource-constrained project scheduling. // Journal of the Operational Research Society, том 56, 2005 г., стр. 365–381.

[136] Neumann K., Schwindt C., Zimmermann J. Resource-constrained project scheduling with time windows: Recent developments and new applications. // Из книги: "Perspectives in Modern Project Scheduling.", Springer Science & Business Media, Нью-Йорк, США, 2006 г., стр. 375–407.

[137] Drexl A., Kimms A. Optimization guided lower and upper bounds for the resource investment problem. // Journal of the Operational Research Society, том

52, вып. 3, 2001 г., стр. 340–351.

[138] Ranjbar M., Kianfar F., Shadrokh S. Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. // Applied Mathematics and Computation, том 196, вып. 2, 2008 г., стр. 879–888.

[139] Yamashita D.S., Armentano V.A., Laguna M. Robust optimization models for project scheduling with resource availability cost. // Journal of Scheduling, том 10, вып. 1, 2007 г., стр. 67–76.

[140] Shadrokh S., Kianfar F. A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. // European Journal of Operational Research, том 181, вып. 1, 2007 г., стр. 86–101.

[141] Nübel H. The resource renting problem subject to temporal constraints. // OR Spektrum, том 23, вып. 3, 2001 г., стр. 359–381.

[142] Ballestin F. A genetic algorithm for the resource renting problem with minimum and maximum time lags. // Lecture Notes in Computer Science, том 4446, 2007 г., стр. 25–35.

[143] Bandelloni M., Tucci M., Rinaldi R. Optimal resource leveling using non-serial dynamic programming. // European Journal of Operational Research, том 78, вып. 2, 1994 г., стр. 162–177.

[144] Davis K.R., Stam A., Grzybowski R.A. Resource constrained project scheduling with multiple objectives: A decision support approach. // Computers and Operations Research, том 19, вып. 7, 1992 г., стр. 657–669.

[145] Maniezzo V., Mingozzi A. The project scheduling problem with irregular starting time costs. // Operations Research Letters, том 25, вып. 4, 1999 г., стр. 175–182.

[146] Möhring R.H., Schulz A.S., Stork F., Uetz M. On project scheduling with irregular starting time costs. // Operations Research Letters, том 28, вып. 4, 2001 г., стр. 149–154.

[147] Möhring R.H., Schulz A.S., Stork F., Uetz M. Solving project scheduling problems by minimum cut computations. // Management Science, том 49, вып. 3,

2003 г., стр. 330–350.

[148] Achuthan N., Hardjawidjaja A. Project scheduling under time dependent costs – A branch and bound algorithm. // Annals of Operations Research, том 108, вып. 1-4, 2001 г., стр. 55–74.

[149] Dodin B., Elimam A.A. Integrated project scheduling and material planning with variable activity duration and rewards. // IIE Transactions, том 33, вып. 11, 2001 г., стр. 1005–1018.

[150] Nonobe K., Ibaraki T. A metaheuristic approach to the resource constrained project scheduling with variable activity durations and convex cost functions. // Из книги: "Perspectives in Modern Project Scheduling.", Springer Science & Business Media, Нью-Йорк, США, 2006 г., стр. 225–248.

[151] Rummel J.L., Walter Z., Dewan R., Seidmann A. Activity consolidation to improve responsiveness. // European Journal of Operational Research, том 161, вып. 3, 2005 г., стр. 683–703.

[152] Kimms A. Maximizing the net present value of a project under resource constraints using a lagrangian relaxation based heuristic with tight upper bounds. // Annals of Operations Research, том 102, вып. 1-4, 2001 г., стр. 221–236.

[153] Mika M., Waligora G., Weglarz J. Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. // European Journal of Operational Research, том 164, вып. 3, 2005 г., стр. 639–668.

[154] Padman R., Zhu D. Knowledge integration using problem spaces: A study in resource-constrained project scheduling. // Journal of Scheduling, том 9, вып. 2, 2006 г., стр. 133–152.

[155] Vanhoucke M., Demeulemeester E.L., Herroelen W.S. On maximizing the net present value of a project under renewable resource constraints. // Management Science, том 47, вып. 8, 2001 г., стр. 1113–1121.

[156] Varma V.A., Uzsoy R., Pekny J., Blau G. Lagrangian heuristics for scheduling

new product development projects in the pharmaceutical industry. // Journal of Heuristics, том 13, вып. 5, 2007 г., стр. 403–433.

[157] Najafi A.A., Niaki S.T.A. A genetic algorithm for resource investment problem with discounted cash flows. // Applied Mathematics and Computation, том 183, вып. 2, 2006 г., стр. 1057–1070.

[158] Herroelen W.S., Dommelen P., Demeulemeester E.L. Project network models with discounted cash flows: A guided tour through recent developments. // European Journal of Operational Research, том 100, вып. 1, 1997 г., стр. 97–121.

[159] Dayanand N., Padman R. On modelling payments in projects. // Journal of the Operational Research Society, том 48, вып. 9, 1997 г., стр. 906–918.

[160] Etgar R., Shtub A., LeBlanc L.J. Scheduling projects to maximize net present value – the case of time-dependent, contingent cash flows. // European Journal of Operational Research, том 96, вып. 1, 1997 г., стр. 90–96.

[161] Hapke M., Jaszkiewicz A., Slowinski R. Interactive analysis of multiple-criteria project scheduling problems. // European Journal of Operational Research, том 107, вып. 2, 1998 г., стр. 315–324.

[162] Slowinski R., Soniewicki B., Weglarz J. DSS for multiobjective project scheduling. // European Journal of Operational Research, том 79, вып. 2, 1994 г., стр. 220–229.

[163] Dörner K.F., Gutjahr W.J., Hartl R.F., Strauss C., Stummer C. Nature-inspired metaheuristics for multiobjective activity crashing. // Omega, том 36, вып. 6, 2008 г., стр. 1019–1037.

[164] Nabrzynski J., Weglarz J. Knowledge-based multiobjective project scheduling problems. // Из книги: "Project Scheduling: Recent Models, Algorithms and Applications.", Springer Science & Business Media, Нью-Йорк, США, 1999 г., стр. 383–411.

[165] Kolisch R. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. // European Journal of Operational Research,

том 90, 1996 г., стр. 320–333.

[166] Kolisch R., Sprecher A. PSPLIB — A project scheduling problem library. // European Journal of Operational Research, том 96, вып. 1, 1997 г., стр. 205-216, DOI: 10.1016/S0377-2217(96)00170-1.

[167] Kelley J.E.J., Walker M.R. Critical-Path Planning and Scheduling. // Proceedings of the eastern joint computer conference, 1959, стр. 160–173.

[168] Land A.H., Doig A.G. An automatic method of solving discrete programming problems. // Econometrica, том 28, вып. 3, 1960 г., стр. 497–520.

[169] Brucker P., Knust S. Complex scheduling. // Springer, Берлин, Германия, 2012 г., 350 с.

[170] Kolisch R. Project Scheduling under Resource Constraints: Efficient Heuristics for Several Problem Classes. // Springer, Берлин, Германия, 1995 г., 212 с.

[171] Kolisch R., Schwindt C., Sprecher A. Benchmark instances for project scheduling problems. // Из книги: "Handbook on recent advances in project scheduling", 1999 г., стр. 197–212.

[172] Kolisch R., Hartmann S. Heuristic algorithms for solving the resource-constrained project scheduling problem — Classification and computational analysis. // Из книги: "Handbook on recent advances in project scheduling", 1999 г., стр. 147–178.

[173] Hartmann S., Kolisch R. Experimental evaluation of state-of-the-art heuristics for resource constrained project scheduling. // European Journal for Operational Research, том 127, вып. 2, 2000 г., стр. 394–407.

[174] Kolisch R., Hartmann S. Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update // European Journal of Operational Research, том 174, вып. 1, 2006 г., стр. 23–37.

[175] R. L. Modeling Resource Alternatives in Project Scheduling. // Дипломная работа, Munich University of Applied Sciences, 2005 г.

[176] 43 полезных сервиса для управления проектами. Без эпитетов.

[Электронный ресурс] // URL: https://habrahabr.ru/post/276873/ (дата обращения: 9 февраля 2016).

[177] Creemers T., Giralt L.R., Riera J., Ferrarons C., Roca J., Corbella X. Constraint-based Maintenance Scheduling on an Electric Power Distribution Network // Proceedings of the 3rd International Conference and Exhibition on Practical Applications of Prolog, Париж, Франция, 1995, стр. 135–144.

[178] PLanning Activities on NETworkS [Электронный ресурс] // URL: http:// www.iri.upc.edu/research/webprojects/planets/

[179] Simonis H., Cornelissens T. Modelling producer/consumer constraints // Proceedings 1st International Conference on Principles and Practice of Constraint Programming (CP95), 1995, стр. 449–462.

[180] Atlas Venture [Электронный ресурс] // URL: https://atlasventure.com

[181] Aggoun A., Gloner Y., Simonis H. Global constraints for scheduling in CHIP // Invited Industrial Presentation, JFPLC'99, Лион, Франция, 1999.

[182] Glaisner F., Richard L.M. FORWARD-C: A refinery scheduling system. // Proceedings of the 3rd International Conference on Practical Applications of Constraint Technology (PACT'97), Лондон, Великобритания, 1997.

[183] Fromherz M., Gupta V., Saraswat V. Model-based computing: constructing constraint-based software for electro-mechanical systems. // Proceedings of the International Conference on Practical Applications of Constraint Technology (PACT'95), Париж, Франция, 1995, стр. 63–66.

[184] Baues G., Kay P., Charlier P. Constraint based resource allocation for airline crew management. // Proccedings of the Aviation, Transport and Travel Informations Systems Conference/Exhibition (ATTIS'94), Париж, Франция, 1994.

[185] Collignon C. Gestion optimisee de ressources humaines pour l’audiovisuel. // Proccedings of the CHIP Users' Club, Масси, Франция, 1996.

[186] COSYTEC [Электронный ресурс] // URL: http://www.cosytec.com/ constraint_programming/cases_studies/administration.htm

[187] Simonis H., Charlier P. Cobra — a system for train crew scheduling. // Proceedings of the DIMACS workshop on constraint programming and large scale combinatorial optimization, 1998.

[188] Chew T., David J.M. A constraint-based spreadsheet for cooperative production planning. // Proceedings of the AAAI SIGMAN workshop on knowledge-based production planning, scheduling and control, 1992.

[189] Shvetsov I., Kornienko V., Preis S. Interval spreadsheet for problems of financial planning. // Proceedings of the 3rd International Conference on Practical Applications of Constraint Technology (PACT'97), Лондон, Великобритания, 1997, стр. 373–385.

[190] Fruhwirth T., Brisset P. Optimal planning of digital cordless telecommunication systems. // Proceedings of the 3rd International Conference on Practical Applications of Constraint Technology (PACT'97), Лондон, Великобритания, 1997.

[191] Chen S.M., Griffis F.H., Chen P.H., Chang L.M. A framework for an automated and integrated project scheduling and management system. // Automation in Construction, том 35, 2013 г., стр. 89–110.

[192] Tulke J., Nour M., Beucke K. A Dynamic Framework for Construction Scheduling based on BIM using IFC. // IABSE Congress Report, 2008 г., стр. 158–159.

[193] ISO 16739:2013 [Электронный ресурс] // International Organization for Standardization. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=51622

[194] Фреймворк [Электронный ресурс] // Википедия. URL: https://ru.wikipedia.org/wiki/Фреймворк

[195] Лаврищева Е.М. Software Engineering компьютерных систем. Парадигмы, технологии и CASE-средства программирования. // Наукова думка, Киев, Украина, 2013 г., 283 с.

[196] Riehle D. Framework Design - A Role Modeling Approach. // Диссертационная работа на соискание учёной степени доктора технических наук, Щвейцарский Федеральный Технологический Институт, Цюрих, Швейцария, 2000 г., 230 с.

[197] Горбунов-Посадов М.М. Расширяемые программы. // Полиптих, Москва, Россия, 1999 г., 336 с.

[198] Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. // Addison-Wesley, 1994 г., 395 с., ISBN: 0-201-63361-2.

[199] Schmidt D.C. Applying Patterns and Frameworks to Develop Object-Oriented Communication Software. MacMillan Computer Publishing-е издание. Том 1. // Из книги: "Handbook of Programming Languages", 1997 г.

[200] Martin Fowler: InversionOfControl. [Электронный ресурс] // URL: https:// martinfowler.com/bliki/InversionOfControl.html

[201] Официальный Интернет-сайт продукта «Synchro» [Электронный ресурс] // «Synchro Software». URL: http://synchroltd.com

[202] Аничкин А.С., Казаков К.А., Семенов В.А. Календарно-сетевое планирование индустриальных проектов с учётом перегруженности рабочих зон. // Информационные и математические технологии в науке и управлении: Труды XX Байкальской Всероссийской конференции (30 июня – 07 июля 2015 г.), в 3 томах (под ред. Массель Л.В.), Иркутск, Байкал, Россия, 2015, том 1, стр. 7–14.

[203] Semenov V., Anichkin A., Morozov S., Tarlapan O., Zolotov V. Effective project scheduling under workspace congestion and workflow disturbance factors. // Australasian Journal of Construction Economics and Building Conference Series, том 2, вып. 1, 2014 г., стр. 35–50, ISSN: 2200-7679.

[204] Semenov V., Anichkin A., Morozov S., Tarlapan O., Zolotov V. Effective project scheduling under workspace congestion and workflow disturbance factors. //

Proceedings of the 13th International Conference on Construction Applications of Virtual Reality (под ред. Dawood N., Kassem M.), 30 – 31 октября 2013 г., Лондон, Великобритания, стр. 239–252.

[205] Семенов В.А., Аничкин А.С., Казаков К.А., Тарлапан О.А. Пространственно-временное моделирование и планирование индустриальных проектов. // Труды научной конференции, посвященной 80-летию со дня рождения академика В.А. Мельникова, 19 – 20 февраля 2009 г., Москва, Россия, стр. 111–113.

[206] Семенов В.А., Аничкин А.С., Казаков К.А. О некоторых актуальных задачах визуального моделирования проектных планов. // Материалы XXXVI международной конференции «Информационные технологии в науке, образовании, телекоммуникации и бизнесе» IT + SE`09, весенняя сессия, 20 – 30 мая 2009 г., Ялта–Гурзуф, Крым, Украина, стр. 64–65.

[207] Щербина О.А. Удовлетворение ограничений и программирование в ограничениях. // Интеллектуальные системы - теория и приложения, том 15, вып. 1-4, 2011 г., стр. 53–170, ISSN: 2411-4448.