ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ИРКУТСКИЙ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**На правах рукописи**

**ПАХОМОВ ДМИТРИЙ ВЯЧЕСЛАВОВИЧ**

# ПРОГРАММНЫЕ КОМПЛЕКСЫ ЧИСЛЕННОГО РЕШЕНИЯ РЕСУРСНЫХ ЗАДАЧ НА ОСНОВЕ ЛОГИКО-ЭВРИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ

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

**Диссертация на соискание ученой степени
кандидата технических наук**

**Научный руководитель:**
**д.ф.-м.н., профессор**
**В.И. Мартьянов**

Иркутск
2017

**Оглавление**

Реферат ..... 4
Введение ..... 5
1 Логико-эвристический подход к математическому моделированию и решению ресурсных задач ..... 15
1.1 Обзор методов поддержки организации функционирования сложных объектов ..... 15
1.2 Логико-эвристические методы сетевого планирования ..... 19
1.3 Общая схема детализации процедурной семантики логико-эвристических методов решения задач сетевого планирования ..... 25
1.4 Информационные технологии для реализации программных комплексов, решающих ресурсные задачи на основе логико-эвристического подхода ..... 28
1.4.1 Применение реляционных баз данных и SQL для математических моделей, допускающих решение задач логико-эвристическими методами ..... 28
1.4.2 Проектирование архитектуры современных программных комплексов ..... 31
1.4.3 Реализация численных методов решения ресурсных задач ..... 36
2 Математические модели для решения задач в дорожной отрасли и реализация программного комплекса мониторинга и поддержки управления сетью автомобильных дорог ..... 49
2.1 Математические модели для решения задач поддержки управления сетью автомобильных дорог ..... 49
2.1.1 Сетевое планирование работ по нормативному содержанию сети автодорог ..... 49
2.1.2 Нахождение областей обслуживания ресурсных баз сети автодорог ..... 57
2.1.3 Автоматизации создания проектов организации дорожного движения на основе применения правил, соответствующих положениям нормативных документов ..... 61
2.2 Реализация программного комплекса поддержки управления дорожной сетью ..... 69
2.2.1 Организация модульной структуры программного комплекса мониторинга и поддержки управления сети автодорог ..... 69
2.2.2 Организация хранилища данных и основные технические решения, используемые в проекте базы данных ..... 76
3 Применение логико-эвристического подхода для создания программного комплекса, обеспечивающего решение задач организации учебного процесса 81
3.1 Математические модели задач организации учебного процесса ..... 81
3.2 Этапы организации учебного процесса ..... 85
3.3 Реализация программного комплекса поддержки организации учебного процесса ..... 91
4 Применение логико-эвристического подхода при создании программного комплекса для определения параметров тепловых сетей ..... 94

4.1 Математическая модель для задачи определения параметров тепловых сетей на стадии схемы теплоснабжения и проектов тепловых сетей ..... 94
4.2 Реализация программного комплекса определения параметров тепловых сетей ..... 97
Заключение ..... 102
Список использованных источников ..... 103
Приложение А ..... 109
Копия справки о внедрении «Системы мониторинга и поддержки управления дорожной сетью» в ОГКУ «Дирекция по строительству и эксплуатации дорог Иркутской области» ..... 109
Приложение Б ..... 110
Копия справки о внедрении «Системы поддержки учебного процесса» в Иркутском государственном университете путей сообщения ..... 110
Приложение В ..... 111
Копия справки о внедрении «Системы поддержки учебного процесса» в Иркутском национальном исследовательском техническом университете ..... 111
Приложение Г ..... 112
Копия акта о внедрении программного комплекса «КЕДР» в АО «Сибирский ЭНТЦ» ..... 112
Приложение Д ..... 113
Копия сертификата соответствия программного комплекса «КЕДР» ..... 113

## Реферат

В диссертационной работе исследуются проблемы математического моделирования производственной деятельности и численного решения ресурсных задач на примере поддержки управления сетью автомобильных дорог Иркутской области, а также для 2-х других предметных областей: организации учебного процесса и проектировании тепловых сетей.

Разработанные математические методы и проекты баз данных реализованы в форме программных комплексов для следующих крупных организаций:

— ОГКУ «Дирекция по строительству и эксплуатации автомобильных дорог Иркутской области»;

— Департамента дорожной деятельности администрации г. Иркутска;

— ФГБОУ ВО «Иркутский национальный исследовательский технический университет»;

— ФГБОУ ВО «Иркутский государственный университет путей сообщения»;

— АО «Сибирский энергетический научно-технический центр».

## Введение

**Актуальность исследования.** Повышение эффективности управления любых крупных организаций в настоящее время трудно достижимо без внедрения комплексов программ, созданных с применением современных информационных технологий, обеспечивающих сетевое планирование этапов работы, поддержку принятия управленческих решений на основе анализа информации, включая решение ресурсных задач. Причем актуальность и достоверность этой информации обеспечивается взаимодействием на основе электронного документооборота структурных служб и отделов организации, интеграции данных, полученных высокотехнологическими системами сбора информации (например, передвижными дорожными лабораториями, оснащенными панорамной видеосъемкой, комплексами глобального позиционирования, георадарами, лазерными сканерами, ультразвуковыми датчиками и пр.).

В диссертационной работе рассматриваются вопросы построения математических моделей и методы численного решения расчетных задач ресурсного планирования этапов работы на основе логико-эвристического подхода [39, 40], а также создания программных комплексов, решающих эти задачи.

Математическое моделирование и создание на его основе программных комплексов поддержки функционирования сложных объектов и процессов, как правило, включает необходимость эффективного численного решения задач сетевого планирования (календарное, ресурсное и стоимостное планирование этапов производств, обслуживания и др.). По алгоритмической природе задачи сетевого планирования относятся к комбинаторным проблемам большой сложности (подкласс *NP* – полных проблем), которые по известным теоретическим результатам [34] не могут быть решены эффективно ни одним из известных современных методов (знаменитая проблема равенства алгоритмической сложности $NP =? P$, которую многие специалисты считают центральной проблемой современной теории алгоритмов, что подтверждает премия в миллион долларов за ее решение математического института Клэя, США).

Причем отметим, что, например, методы линейного программирования, видимо, не могут эффективно решать все задачи сетевого планирования по известному результату советского математика Хачияна [34], полученному в 1979г., о полиномиальной сложности решения задач линейного программирования.

Это определяет необходимость на современном этапе расширять спектр методов решения задач сетевого планирования, каждый востребованный из которых должен иметь свою «нишу» (область эффективной работы). В настоящее время классические методы решения комбинаторных проблем большой сложности, основанные на линейном, целочисленном, динамическом и других программированиях, классической теории расписаний и т. д., пополнились методами в той или иной степени идущих от логики (логическое, функциональное, семантическое программирование, позитивно – образованные логические исчисления, логико-эвристический подход, метод удовлетворения ограничениям в логическом программировании) и этими методами решены многие практические задачи большой размерности.

Результатом диссертационной работы является создание программных комплексов и их баз данных, реализованных с применением современных информационных технологий, для мониторинга и поддержки управления сетью автомобильных дорог, организации учебного процесса, проектировании тепловых сетей.

Задача планирования работ по нормативному содержанию сети автодорог, которая заключается в расчете стоимости и планировании сроков исполнения работ по содержанию сети автодорог, имеет два варианта решения.

Первый вариант утилитарный – по сформированному регламенту работ, где определен перечень работ по объектам и их периодичность, найти стоимостные, временные и ресурсные затраты. Если затраты не могут быть обеспечены имеющимся финансированием, то вносить вручную корректировки в регламент работ до «вписывания» затрат в имеющееся финансирование. Второй вариант состоит в минимизации необходимых затрат, что может быть сделано уменьшением затрат при выполнении самих конкретных работ из регламента.

В настоящее время комплексное решение такой задачи не определено в рекомендованных методиках или нормативных документах, в то время как для повышения эффективности организации работы дорожного хозяйства важное значение имеет рациональное расходование средств, которых всегда не хватает для обеспечения нормативного содержания.

В качестве близких подходов отметим результаты:

— Т.В. Бобровой [12] по проектно-ориентированному управлению производством работ на региональной сети автомобильных дорог;

— Н.С. Жилина [23] по информационно-справочным системам для органов управления региональным дорожным хозяйством;

— С.С. Близниченко [11] по теоретическим основам разработки комплексной автоматизации системы управления автомобильными дорогами Краснодарского края.

В диссертационной работе рассмотрен подход к автоматизации создания проектов организации дорожного движения на основе применения правил, соответствующих положениям нормативных документов (ГОСТы, СНиПы и др.). Разработано унифицированное представление для данных по автомобильным дорогам и нормам проектирования, что позволило решить вопрос автоматизации применения правил проектирования к конкретным данным по автомобильной дороге и ее обустройству.

В диссертации уточняется формализация схемы численного решения задач сетевого планирования логико-эвристическими методами с позиции итерационного подхода к построению алгебраических систем, удовлетворяющих ограничениям. Возможность применения подхода к математическому моделированию и решению задач показана на примере решения задач еще 2-х предметных областей: организация учебного процесса и проектирование схем теплоснабжения.

Построенные методы численного решения задач сетевого планирования используют логико-эвристический подход, предложенный В.И. Мартьяновым [39, 40].

**Целью представленных в диссертационной работе исследований** является разработка математических моделей и программная реализация решения ресурсных задач на основе логико-эвристического подхода. Для достижения этой цели последовательно решены следующие задачи:

Разработаны численные алгоритмы решения задач сетевого планирования, как итерационное построение многоосновных алгебраических систем, удовлетворяющих ограничениям.

Предложены математические модели, позволяющие применять логико-эвристических подход для решения задач.

Детализированы описания математических моделей для каждой из выделенных задач сетевого планирования, как построение многоосновных алгебраических систем, удовлетворяющих ограничениям.

Разработаны и внедрены программные комплексы поддержки организации управления производственными процессами для нескольких крупных организаций г. Иркутска.

**Объектом исследования** является решение ресурсных задач, возникающих в производственной деятельности следующих организаций

— ОГКУ «Дирекция по строительству и управлению автомобильных дорог Иркутской области»;

— Иркутского национального исследовательского технического университета (ИРНИТУ);

— Иркутского государственного университета путей сообщения (ИрГУПС):

— АО «Сибирский энергетический научно-технический центр» («Сибирский ЭНТЦ»).

**Предметом исследования** является

- Построение математических моделей для решения задач сетевого планирования, где требуется удовлетворение ограничениям, соответствующим нормативным требованиям.

- Алгоритмизация решения задач, формирование проектов баз данных на основании построенных моделей, создание автоматизированных рабочих мест (АРМ) специалистов предприятий различной направленности, поддержка актуальности и достоверности информационного наполнения баз данных на основе электронного документооборота структурных служб и отделов предприятий, решение базовых прикладных задач.

**Методы исследований:** математическая логика, теория моделей, теория программирования, включая многоосновные алгебраические системы, реляционные модели данных, объектно-ориентированное программирование, логико-эвристический подход к решению комбинаторных задач высокой сложности.

**Научная новизна** состоит в том, что в диссертационной работе впервые:

— применён логико-эвристический подход для построения математических моделей, представляемых многоосновными алгебраическими системами;

— на основе подхода построены математические модели для планирования нормативного содержания сети автодорог, организации учебного процесса и проектирования схем теплоснабжения;

— получены численные алгоритмы решения задач сетевого планирования производственной деятельности, указанных выше предметных областей, как итерационное построение многоосновных алгебраических систем, удовлетворяющих ограничениям и их программные реализации.

Достоверность и обоснованность результатов, представленных в диссертации, обусловлена использованием теоретически обоснованных методов алгебры, математической логики, теории моделей, современной теории программирования, логико-эвристического подхода к решению комбинаторных задач высокой сложности, а также, подтверждается сопоставлением данных численного и натурного экспериментов.

Положения, выносимые на защиту:

1. Методика проектирования реляционных баз данных для представления задач сетевого планирования, как построение многоосновных алгебраических систем, удовлетворяющих ограничениям.

2. Численные алгоритмы решения задач сетевого планирования на основе логико-эвристического подхода к решению комбинаторных задач высокой сложности.

3. Математические модели задач организации мониторинга, поддержки планирования нормативного содержания сети автомобильных дорог.

4. Математические модели для задач организации учебного процесса.

5. Математическая модель задачи определения параметров тепловых сетей на стадии проектирования.

6. Комплекс программ по мониторингу и поддержке управления сети автомобильных дорог Иркутской области.

7. Комплекс программ по поддержке организации учебного процесса для ИРНИТУ и ИрГУПС.

8. Комплекс программ для определения оптимальных параметров тепловых сетей на стадии схем теплоснабжения и проектов тепловых сетей.

**Практическая значимость исследования.** На основе полученных в диссертационной работе результатов разработаны программные комплексы, используемые, в частности, для организации деятельности нескольких крупных учреждений г. Иркутска.

А именно, программный комплекс мониторинга и поддержки управления сетью региональных автомобильных дорог используется для:

— организации нормативного содержания сети региональных автомобильных дорог;

— формирования базы данных сети автодорог на основе интеграции: инвентаризационных и технических паспортов, а также, видеопаспортизации;

— проектирования и корректировки дислокации дорожных знаков и схем разметки, а также в целом проектов организации дорожного движения (ПОДД);

— формирования информационных отчетов;

— объективного обоснования стоимости проведения работ по содержанию участков сети автомобильных дорог на конкурсных торгах.

Программный комплекс поддержки организации учебного процесса высших учебных заведений используется для:

— формирования и редактирования учебных планов специальностей по всем формам обучения (дневной, вечерней, заочной);

— формирования и редактирования рабочих учебных планов специальностей по всем формам обучения (дневной, вечерней, заочной);

— формирования и редактирования типового и рабочего графика учебного процесса по всем формам обучения (дневной, вечерней, заочной);

— формирования и редактирования контингента учащихся;

— формирования и редактирования норм времени по видам работ;

— расчета нагрузки по рабочему учебному плану и контингенту учащихся на основании норм времени по видам работ;

— формирования и редактирования баз данных основных объектов (профессорско-преподавательский состав, аудиторный фонд и др.);

— формирования и редактирования учебных поручений кафедрам и преподавателям;

— распределения аудиторного фонда по занятиям;

— автоматизированного расчета расписания по всем формам обучения (дневной, вечерней, заочной).

Программный комплекс определения параметров тепловых сетей используется для:

— формирования схемы сети;

— определения способа реконструкции для существующих участков сетей и расчета диаметров для новых;

— формирования пьезометрических графиков;

— формирования отчетов по ресурсным затратам.

Результаты диссертационной работы внедрены в учебный процесс ИРНИТУ для подготовки специалистов по направлению 270100 «Строительство» специальности 270205 «Автомобильные дороги и аэродромы».

Апробация результатов исследования

Результаты диссертационной работы докладывались на следующих конференциях:

— Винеровские чтения, ИРНИТУ, 2005г.;

— Новые технологии в инвестиционно-строительной сфере и ЖКХ, Иркутск, 2005г.;

— Научно-практическая конференция «Дальний Восток. Автомобильные дороги и безопасность движения», Хабаровск, 2006г.;

— 3-ей Российской школе – семинаре «Синтаксис и семантика логических систем», Иркутск, 2010.

— 5-ой Открытой конференции молодых специалистов ЗАО «Сибирский ЭНТЦ», Новосибирск, 2012.

— на семинарах кафедр «Автомобильные дороги», «Вычислительная техника» ИРНИТУ, института «Строительства и архитектуры» ИРНИТУ, ИДСТУ СО РАН и ИГУ.

**Публикации.** Основные результаты диссертации опубликованы в 15 работах [31, 32, 41 – 46, 51 – 56, 58], в том числе семь статей в рецензируемых изданиях, рекомендованных ВАК РФ для опубликования диссертационных исследований [31, 32, 43, 44, 46, 53, 55]. В работах с соавторами соискателю принадлежат от 25 до 50% результатов. Положения, составляющие новизну и выносимые на защиту, получены лично автором.

Основные программы, используемые в диссертационной работе, имеют свидетельства о государственной регистрации программы для ЭВМ в Федеральном институте промышленной собственности [51, 52, 54, 56].

Структура и объем работы. Диссертация включает введение, четыре главы основного текста, заключение, библиографический список из 71 наименования, приложения. Общий объем диссертации 116 страниц, в тексте содержится 11 рисунков и 16 таблиц. В приложении приведены материалы о внедрении результатов диссертационной работы.

**Во введении** обоснована актуальность проблемы и приведена общая характеристика диссертации с изложением основных положений и результатов работы, обозначена цель и задачи исследований.

**Первая глава** включает обзор методов математического моделирования работы сложных объектов, включая экономико-математические методы. Основное внимание уделено логико-эвристическому подходу к решению задач сетевого планирования, которые наиболее адекватно моделируют рассмотренные этапы производственных процессов, включая решения ресурсных задач. Приведены математические постановки задач и оценки алгоритмической сложности их решения.

Также в главе рассмотрены информационные технологии для программных комплексов. Отмечается важное значение логико-эвристического подхода для построения баз данных, которые сами обеспечивают значительную часть программной реализации. Рассмотрена реализация численных методов, позволяющая применить логико-эвристический подход для решения практических задач.

**Вторая глава** посвящена применению логико-эвристического подхода для задач в дорожной отрасли. Рассмотрена задача сетевого планирования работ по нормативному содержанию сети автомобильных дорог, как расчет стоимости и планирование сроков исполнения работ по содержанию сети автомобильных дорог с минимальными затратами (временными, ресурсными и стоимостными). Также рассмотрены задачи по обслуживанию баз противогололедных материалов и формирования проектов организации дорожного движения. Описана реализация программного комплекса мониторинга и поддержки управления сетью автомобильных дорог.

**В третей главе** рассмотрена организация учебного процесса, как решение ряда задач сетевого планирования: от формирования учебных планов по стандартам до проектирования расписания занятий. Для решения данных задач применен логико-эвристический подход, позволивший реализовать программный комплекс поддержки организации учебного процесса.

**Четвертая глава** посвящена применению логико-эвристического подхода к созданию математических моделей и реализации программного комплекса определения параметров тепловых сетей на стадии проектов теплоснабжения.

# 1 Логико-эвристический подход к математическому моделированию и решению ресурсных задач

## 1.1 Обзор методов поддержки организации функционирования сложных объектов

При решении ресурсных задач широко используются ряд формализованных методов, которые в литературе иногда называются экономико-математическими, то есть методами, позволяющими находить оптимальные или близкие к ним решения. Часть из них нашла применение в современных системах организации производственной деятельности. Постановка задачи должна отражать существующие ограничения, которые возникают из ограниченности ресурсов или внешних условий, в которых осуществляется хозяйственная деятельность предприятий. Критерий оптимизации обычно формализуется в виде целевой функции, которую требуется, исходя из поставленной задачи, минимизировать или максимизировать.

Критериями оптимизации могут являться, например, объемы продаж, затраты на производство, прибыль, время выпуска, загрузка оборудования и т. п. Переменными в экономико-математических моделях являются управляемые параметры. При решении задач оптимизации переменными могут быть количество выпускаемых изделий, время запуска/выпуска, размеры партий, уровень запасов, время начала и окончания операций. Важной особенностью экономико-математических методов является то, что они могут способствовать анализу изучаемой модели. С их помощью, например, можно определить, что при заданных ограничениях допустимого решения не существует. Некоторые методы не ограничиваются получением оптимального решения: они позволяют оценить чувствительность построенного решения к изменению начальных условий или ограничений, налагаемых в деятельности предприятия внешними условиями или внутренними характеристиками.

Многообразие экономико-математических методов достаточно велико.

*Линейное программирование* заключается в поиске оптимального решения для линейной целевой функции при линейных ограничениях и ограничениях на не отрицательность переменных.

В терминах линейного программирования формулируется широкий круг задач технико-экономического планирования. Следует отметить, что методы линейного программирования успешно работают на задачах с полиномиальной сложностью [34]. Однако, с помощью этих методов можно получить не только оптимальное решение, но и исследовать чувствительность решения к изменениям исходных данных.

Частным случаем линейного программирования является транспортная модель. Она получается естественным образом при формализации задачи планирования перевозок, однако с ее помощью можно решать и другие задачи поддержки организации производства (назначение кадров на рабочие места, составление сменных графиков и др.). Специфическая структура ограничений задачи позволила разработать эффективные методы решения.

Важное место в системах поддержки организации производства принадлежит методам дискретного программирования, которые ориентированы на решение задач оптимизации с целочисленными (частично или полностью) переменными [21].

Задачи с булевыми переменными являются частным случаем задач дискретного программирования. Примерами таких задач можно указать задачи о размещении производств, задачи о назначении, задачи синтеза структуры транспортных и информационных сетей и т. п. Для решения задач дискретного программирования разработаны различные алгоритмы, в том числе методы ветвей и границ, отсечения и др.

Модели *стохастического программирования* описывают ситуации, в которых элементы модели являются случайными величинами с известными функциями распределения.

Сетевые модели и методы применяются там, где есть возможность четко структурировать управляемый процесс в виде графа, описывающего взаимосвязи работ, ресурсов, временных затрат и т. п. Разработан ряд методов решения задач на сетевых моделях по определению критического пути, распределению ресурсов.

*Динамическое программирование* представляет собой многошаговый процесс получения решения оптимальной задачи [13]. Наиболее естественной выглядит формализация динамических задач, однако этот метод успешно может применяться и для статических задач, если удается разбить решение исходной задачи на этапы. Серьезным ограничением применения метода динамического программирования является размерность задач. Если размерность велика, то необходимо запоминать большой объем промежуточной информации. Практически, решение задач оптимизации возможно для систем, имеющих размерность не выше трех.

*Математическая статистика* применяется для решения задач анализа и прогнозирования экономических и социальных процессов на предприятиях, создания и корректировки нормативной базы. Наиболее часто применяются методы: расчета статических характеристик, корреляционного, регрессионного и дисперсионного анализа.

*Теория управления запасами* позволяет определять уровни запасов материалов, полуфабрикатов, производственных мощностей и других ресурсов в зависимости от спроса на них.

*Теория расписаний* представляет собой методологическую основу для решения задач упорядочения последовательности работ. При этом учитываются структура и параметры технологического процесса. Для решения задач, сформулированных в терминах теории расписаний, используют методы моделирования на основе приоритетов.

*Эвристические методы* (к которым необходимо отнести и используемые в данной диссертационной работе) получили в системах поддержки организации производства достаточно широкое распространение, и дальнейший прогресс в этом направлении связан с разработкой и внедрением экспертных систем. Экспертные системы позволяют накапливать базы знаний о производственном процессе, об эффективных управляющих решениях и на этой основе предлагать рациональные решения задач, слабо поддающихся формализации.

Круг экономико-математических моделей и методов чрезвычайно широк. Их применение сдерживается затрудненностью адекватного описания производ-

ственного процесса, получения решений в условиях высокой размерности задач, а также отсутствием необходимой для этого случая квалификации управленческого персонала.

Ниже перечислены модели и методы решения частных задач управления предприятиями, включаемые в базовые программные системы:

- Для решения задач стратегического планирования применяются модели линейного программирования.
- Оперативное планирование построено, как правило, на базе сетевых моделей. В этом случае используются методы расчета критического пути.
- Для решения задач прогнозирования спроса и других экономических процессов применяются методы регрессионного анализа, анализа временных рядов, процедуры обработки экспертных оценок.
- При решении задач планирования объемов продаж и производства используются методы линейного программирования.
- Задача формирования графика выпуска продукции может быть сформулирована как задача минимизации совокупного производственного цикла при ограничениях по мощностям, где в качестве переменных выступают сроки запуска (выпуска). В базовых программных системах имеются процедуры, позволяющие решить эту задачу путем генерирования, анализа и отсеивания вариантов с одновременным сокращением числа переменных на каждой итерации.
- Задача расчета материальных потребностей на обеспечение графика выпуска продукции решается на основе модели разузлования, в ходе которого, выполняется обсчет сетевой структуры, описывающей состав изделия.

Оперативное управление производством базируется на применении приоритетов и эвристических методов для построения сетевых планов работ [66]. И действительно, данные методы представляются наиболее удобными для автоматизации управления работой предприятия, что подтверждается, в частности, и результатами данной работы.

## 1.2 Логико-эвристические методы сетевого планирования

Управление сложными объектами и процессами часто включает задачи сетевого планирования (календарного, ресурсного и стоимостного), как, отмечалось выше, это имеет прямое отношение ко всем рассматриваемым в диссертации наукоемким задачам. По алгоритмической природе данные задачи, как правило, относятся к комбинаторным проблемам большой сложности (подкласс NP – полных проблем).

В настоящее время классические методы решения комбинаторных проблем большой сложности, основанные на линейном [16, 22, 68], целочисленном [60], динамическом и других программированиях [24, 59], классической теории расписаний [21, 27, 30, 33, 62, 63] и т. д., пополнились методами в той или иной степени идущих от логики (логическое, функциональное, семантическое программирование [19], позитивно – образованные логические исчисления [13 – 15], логико-эвристические методы [9, 37 – 39]), метод удовлетворения ограничениям в логическом программировании [73] и этими методами решены многие практические задачи большой размерности.

В диссертации формализация схемы решения задач сетевого планирования логико-эвристическими методами [39] рассматривается с позиции итерационного подхода к построению алгебраических систем, удовлетворяющих ограничениям [39], причем для проверки выполнимости ограничений на модифицированной алгебраической системе (полученной после применения оператора преобразования) будут использоваться методы распознавания ситуаций [38]. Для иллюстрации полученных результатов будут использоваться постановки задач для метода удовлетворения ограничениям [73].

Как отмечалось во введении, в логико-эвристическом подходе математические модели решения задач сетевого планирования интерпретируются как построение многоосновных алгебраических систем, удовлетворяющих ограничениям.

Более точно, исходной точкой решения является многоосновная алгебраическая система, называемая инициальной, которая последовательно преобразуется

в финальную многоосновную алгебраическую систему, удовлетворяющую ограничениям.

Пусть в логико-эвристическом подходе [39] инициальной моделью является многоосновная алгебраическая система (а. с.)

$$\boldsymbol{M = < A_1 , \ldots, A_s ; f_1 , \ldots, f_n ; p_1 , \ldots, p_k >} \qquad \boldsymbol{(1.1)}$$

где $\boldsymbol{A_i}$ – основные множества, $\boldsymbol{f_i}$ – операции (функции) на основных множествах, $\boldsymbol{p_i}$ – предикаты (отношения) на основных множествах. В дальнейшем этого раздела будем рассматривать только а. с. данной сигнатуры

$$\boldsymbol{\sigma = < f_1 , \ldots, f_n ; p_1 , \ldots, p_k >.}$$

Для решения комбинаторной (сетевой) задачи требуется перестройка отношений и функций такая, что получившаяся в результате а. с. удовлетворяла ограничениям. В рамках классического логического подхода такого рода перестройки ограничиваются формульным заданием каких-либо отношений и функций (через базовые, вычислимые), либо через расширения основных множеств термами (например, списочная надстройка в семантическом программировании [19]), реализуемость (вычислимость) поддерживается автоматическим доказательством теорем (АДТ) [14] или его частным случаем – логическим программированием. Проверка ограничений сводится к интерпретации формул на а. с. (проверке истинности).

При решении комбинаторных проблем подобного рода «уточнений» (через ввод формульных отношений и функций) недостаточно. Схемы решения задач в методе удовлетворения ограничений [73] или в логико-эвристическом подходе [39] требуют определения операторов преобразования функций и отношений исходной а. с. $\boldsymbol{M}$ (1.1), позволяющих получать последовательности а. с. (пространство состояний исходной а. с. $\boldsymbol{M}$), а сама проблема состоит в поиске а. с., удовлетворяющей заданным ограничениям. Подобная формализация напоминает подход Месаровича и Такахары [48] (наиболее близки операторам преобразований – функции перехода состояний многоуровневых систем в теории реализации), но не

требует ввода параметра времени и будет приведена только для случая дискретных конечномерных задач.

Для формализации процесса решения комбинаторных проблем определим основу процесса (ОП), как тройку

$$\Omega = < M,\ \Psi,\ R>, \qquad (1.2)$$

где $M$ – а. с. сигнатуры $\sigma$ (1.1), которая будет рассматриваться как начальные условия процесса решения, множество операторов преобразования а. с. $\Psi = \{ \psi_1, \ldots, \psi_k \}$, множество ограничений $\mathbf{R} = \{ \mu_1, \ldots, \mu_m \}$, определенных на а. с. данной сигнатуры $\sigma$, и принимающих значение истина или ложь.

Слово $\mathbf{W}=\mathbf{w}_1 \ \ldots \ \mathbf{w}_s$ в алфавите $\Psi$ назовем решением комбинаторной проблемы, заданной ОП (1.2), если все ограничения $\mathbf{R}$ выполнены на а. с. $\mathbf{W}(M)=\mathbf{w}_l(\ldots (\mathbf{w}_s(M)) \ \ldots )$. Таким образом, поиск последовательностей операторов преобразований, являющихся решением, и является основным содержанием всех методов логического подхода.

Если искать аналогии с классическими методами, такими как динамическое программирование, то решение $\mathbf{W}$ следует называть допустимым управлением. Основное отличие методов решения комбинаторных проблем логического подхода и традиционных методов (динамическое программирование [10, 18, 28, 71]) состоит в том, что здесь во главу угла ставится удовлетворение ограничениям (поиск допустимого управления), а не вопрос оптимизации по тому или иному критерию целевой функции. Хотя, конечно, относительная оптимизация (выбор лучшего из совокупности найденных допустимых решений) делается практически всегда.

Рассмотрим важнейшие технические приемы для повышения эффективности решения комбинаторных проблем в рамках логического подхода. Пространство поиска решения, очевидно, порождается вариантами выбора операторов преобразований $\{\psi_1, \ldots, \psi_k \}$ в каждой точке пространства. Точку пространства поиска решения назовем тупиковой, если проведенная последовательность преобразований не может быть дополнена до решения комбинаторной проблемы. Стратегия

«смотри вперед» состоит в анализе текущей точки пространства поиска решения и уменьшении вариантов выбора операторов преобразований { $\psi_1$, …, $\psi_k$ }, т. е. убираем из этого множества операторы заведомо ведущие в тупиковые состояния. Стратегия стандартного и глубокого возврата (backtracing) состоит в откате назад при получении тупикового состояния.

При стандартном возврате откат назад делается на один шаг (т. е. берется предыдущая точка пространства состояний) и вместо оператора $\psi_i$ применяется оператор $\psi_{i+1}$ (может быть и большего номера, если используется стратегия «смотри вперед»). При глубоком возврате осуществляется анализ причин тупикового состояния и откат может осуществляться на много шагов назад, что, конечно, очень сильно сокращает перебор. Стратегия «чем хуже, тем лучше» состоит в выборе самой левой точки с признаком влияния на попадание в тупиковое состояния. Понятно, что данная стратегия не полна, но наиболее сильно сокращает пространство поиска. Для обеспечения эффективности реализации возврата необходимо уметь быстро восстанавливать среду точки возврата (восстанавливать а. с., соответствующую этой точке пространства состояний).

В логико-эвристическом подходе [38] глубокий возврат дополняется неполным восстановлением среды точки возврата, что обеспечивает сохранение части алгоритмического ресурса, потраченного между точками возврата и тупика (т. е. некоторые вычисления остаются). Более точно, пусть применение слова $\mathbf{w}_1 \ldots \mathbf{w}_s$ $\mathbf{w}_{s+1} \ldots \mathbf{w}_v$ в алфавите { $\psi_1$, …, $\psi_k$ } приводит к тупиковой точке, а точке возврата соответствует применение слова $\mathbf{w} = \mathbf{w}_1 \ldots \mathbf{w}_s$. В рамках стратегии неполного восстановления среды точки возврата откат может осуществляться к точке пространства состояний, соответствующей применению слова $\mathbf{w}_1 \ldots \mathbf{w}_s \, \mathbf{u}_1 \ldots \mathbf{u}_n$, где слово $\mathbf{u}_1 \ldots \mathbf{u}_n$ входит с точностью до вычеркивания букв в слово $\mathbf{w}_{s+1} \ldots \mathbf{w}_v$. Такой подход значительно усложняет многие технические моменты программной реализации метода (как в плане обеспечения корректности решения, так и в организации данных, фиксирующих траекторию решения, сбор мусора и т. п.), но отключение

этой возможности (неполное восстановление среды точки возврата) не позволяет в реальное время решать, например, задачу планирования расписания занятий.

Рассмотрим конкретизации данной схемы для выше указанных задач. В методе удовлетворения ограничениям для переменных $\mathbf{x}_1$, …, $\mathbf{x}_n$ необходимо найти значения $\mathbf{a}_1$, …, $\mathbf{a}_n$ из конечных множеств $\mathbf{A}_1$, …, $\mathbf{A}_n$, соответственно, удовлетворяющие ограничениям. Нетрудно заметить, что данная постановка может быть трансформирована в задачу построения а. с., удовлетворяющей ограничениям. Операторами преобразований будет выбор элементов из множеств $\mathbf{A}_1$, …, $\mathbf{A}_n$.

В динамическом программировании системе **X** необходимо сопоставить а. с. *M* с основными множествами, соответствующими областям изменения параметров из **X**, если в сигнатуру а. с. *M* ввести отношения, позволяющие присваивать параметрам из **X** любые значения (конечно, еще надо ввести все остальные необходимые функции и отношения, например, сложение, умножение и др.). Тогда в качестве операторов преобразований можно взять совокупность $\mathbf{f(X, y}_1\mathbf{)}$, …, $\mathbf{f(X, y}_m\mathbf{)}$, …, где **f** – функция перехода системы в новое состояние, $\mathbf{y}_i$ – управления.

В работах [39, 40] основные рассмотрения посвящены алгоритмам построения функций и отношений, удовлетворяющих ограничениям. Поэтому нижним уровнем таких алгоритмов является перебор наборов (кортежей) элементов, которые могут быть элементами графика строящихся функций и отношений. Эффективность работы алгоритма построения таких функций и отношений определяется весьма многими аспектами (и, как правило, плохая реализация хотя бы одного аспекта сводит на нет всю работу), но в этом разделе будут рассмотрены только ключевые проблемы проверки выполнимости ограничений на а. с., полученной после применения одного оператора преобразования (хотя иногда проверка ограничений осуществляется после ряда применений операторов преобразований).

Во избежание рассмотрения «всеобщего случая» рассмотрим, следуя работам [39, 40], следующую переборную задачу ресурсного планирования. Пусть задана совокупность объектов планирования $\mathbf{Ob} = \{ \mathbf{O}_1, \ldots, \mathbf{O}_n\}$ и их ресурсов $\mathbf{Res} = \{\mathbf{R}_1, \ldots, \mathbf{R}_n\}$, а процесс решения состоит в последовательной выборке объектов планирования и выделения элемента соответствующего ресурса (в этой последо-

вательности один и тот же объект может встречаться несколько раз, так как ему необходим более чем один элемент своего ресурса).

Таким образом, процесс решения для i-го шага может быть представлен последовательностью $\{\mathbf{O}_{j1}, \mathbf{r}_1\}, \ldots, \{\mathbf{O}_{ji}, \mathbf{r}_i\}$, где $\mathbf{r}_k \in \mathbf{R}_{jk}$, а i+1-ый шаг состоит в выборе:

1) $\mathbf{O}_{ji+1}$ – го объекта и элемента его ресурса $\mathbf{r}i+1$;

2) если это сделать невозможно, то необходимо вернуться на один (или более) шаг назад и осуществить сдвиг в выборе элемента ресурса.

При данном подходе явно выделяются следующие типы данных, структуры управления решением и базовые алгоритмы:

1) данные, представляющие объекты планирования и их ресурсы;

2) структуры управления, фиксирующие траекторию решения и обеспечивающие откат назад на любой шаг решения (т. е. обеспечивающие полное восстановление среды точки возврата);

3) алгоритмы проверки выполнимости ограничений на каждом шаге решения (демоны);

4) алгоритмы восстановления среды точки возврата;

5) алгоритмы сдвига в выборе следующего элемента ресурса;

6) алгоритмы распознавания тупика и определения точки возврата (обеспечивает глубокий возврат);

7) алгоритмы редактирования шагов решения между тупиковой точкой и точкой возврата (обеспечивает глубокий возврат без полного восстановления среды точки возврата).

Алгоритмы пунктов 3) – 7) должны иметь очень эффективную реализацию, отсутствие алгоритма 7) ограничивает область реального применения этих методов.

## 1.3 Общая схема детализации процедурной семантики логико-эвристических методов решения задач сетевого планирования

***Процедурная семантика логико-эвристических методов*** решения конкретной прикладной задачи состоит в том, что многоосновные алгебраические системы преобразуются в реляционные базы данных, где основные множества преобразуются в домены, а функции и отношения преобразуются в реляционные таблицы. Отметим, что проект реляционной базы данных для современных СУБД (SQL Server, Oracle и др.) в значительной мере определяет программные решения для всего программного комплекса, т. е. определяет процедурную семантику метода.

Рассматриваемые задачи в общей постановке, как правило, NP-полны, поэтому нельзя утверждать, что логико-эвристический подход реализуется всюду одинаково. Тем не менее, есть две очень важные вычислительные процедуры выполнение которых должно иметь линейную или квадратичную сложность либо от числа аргументов, либо от числа основных множеств модели.

Прежде всего, при представлении в оперативной памяти необходимо обеспечить линейную вычислимость отношений от их арности (количества аргументных мест). В случае, когда оперативной памяти не хватает для представления всех кортежей отношений многоосновной модели (1.1), необходимо использовать какие-либо процедуры сжатия информации, так чтобы с преобразованными данными можно было бы работать не выходя из квадратичной оценки сложности от арности отношений. Если это по каким-то причинам оказывается недостижимым, то вероятность эффективного решения данного класса комбинаторных задач сомнительна (конечно, это касается только логико-эвристического подхода).

Следующей задачей является обеспечение быстрой проверки ограничений $R_1, R_2, \ldots, R_m$, где идеальной является ситуация когда сложность их проверки не зависит от m и является так же линейной от числа основных множеств s. Достижение такой эффективности вычислений возможно далеко не всегда, но служит хорошим ориентиром для обеспечения этого свойства хотя бы для части отношений.

Эффективная проверка ограничений на модифицированной многоосновной алгебраической системе после применения преобразования во многом обеспечивается фундаментальным свойством логико-эвристических методов комбинаторных задач высокой сложности сформулированным в теореме [38]: Сложность проверки ограничений не зависит от количества ограничений. Верхняя граница сложности проверки истинности ограничений имеет порядок O(n), где n – число показателей ситуаций.

Отдельной проблемой является необходимость «попасть» под условия применения данной теоремы, что бывает весьма и весьма сложно, а иногда вообще невозможно. Но проверка выполнимости большей части ограничений «вписывается» в условия применимости данной теоремы, что обосновывает эффективность предлагаемых в диссертационной работе методов решения задач сетевого планирования.

Разработка алгоритмов численного решения задач (включая сетевое планирование) на основе логико-эвристического подхода имеет две основных особенности:

Организация данных, структур управления, способов оптимизации переборов в общих чертах одинакова для практически всех задач рассматриваемых предметных областей.

Проверка ограничений, генерация множества вариантов переборов, выбор типов стратегий управления решением достаточно индивидуальны и всегда специально настраиваются под конкретный класс задач.

В настоящем разделе рассмотрена идеология общих технических решений по организации данных, структур управления, способов оптимизации переборов и по некоторым другим аспектам реализации численных методов решения задач сетевого планирования на основе логико-эвристического подхода.

Прежде всего, это относится к программной реализации следующих конструкций, определенных выше:

- основных множеств многоосновных алгебраических систем (доменов реляционной БД);

- отношений многоосновных алгебраических систем (таблиц реляционной БД);
- задания ограничений, которым должны удовлетворять многоосновные алгебраические системы, и методов проверки выполнимости этих ограничений;
- задание операторов преобразований многоосновных алгебраических систем и организации перебора последовательностей преобразований;
- стратегии оптимизации перебора последовательностей преобразований.

Важно отметить, что в программной реализации логико-эвристического подхода данные и структуры управления алгоритмами численного решения задач сетевого планирования представляются близкими типами данных, а именно, деревьями, что обеспечивает возможность применения «обратной связи», т. е. данные задачи могут легко преобразовываться в структуры управления и наоборот. Таким образом, особенности данных задачи могут «управлять» процессом численного решения.

Рассмотрим важнейшие технические приемы для повышения эффективности решения комбинаторных проблем в рамках логического подхода. Пространство поиска решения, очевидно, порождается вариантами выбора операторов преобразований $\{\delta_1,\dots\delta_k\}$ в каждой точке пространства. Точку пространства поиска решения назовем тупиковой, если проведенная последовательность преобразований не может быть дополнена до решения комбинаторной проблемы. Стратегия «смотри вперед» состоит в анализе текущей точки пространства поиска решения и уменьшении вариантов выбора операторов преобразований $\{\delta_1,\dots\delta_k\}$ т. е. убираем из этого множества операторы заведомо ведущие в тупиковые состояния. Стратегия стандартного (standart backtracing) и глубокого возврата (intelligent backtracing) состоит в откате назад при получении тупикового состояния.

## 1.4 Информационные технологии для реализации программных комплексов, решающих ресурсные задачи на основе логико-эвристического подхода

### 1.4.1 Применение реляционных баз данных и SQL для математических моделей, допускающих решение задач логико-эвристическими методами

Как отмечалось во введении основой формализации (построения математических моделей) предметной области являются многоосновные алгебраические системы [36], которые для конкретной прикладной задачи реализуются в виде реляционных баз данных, т.к. домены можно трактовать как основные множества, а реляционные таблицы, как отношения на основных множествах.

Имея формализацию производственного процесса в виде математической модели, использующей логико-эвристический подход, можно построить базу данных, которая уже обеспечивает значительную часть программной реализации.

Опишем кратко расширенный начальный вариант алгебры, который был предложен Коддом [70]. В этом варианте набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса – теоретико-множественные операции и специальные реляционные операции. В состав теоретико-множественных операций входят операции:

— объединения отношений;

— пересечения отношений;

— взятия разности отношений;

— прямого произведения отношений.

Специальные реляционные операции включают:

— ограничение отношения;

— проекцию отношения;

— соединение отношений;

— деление отношений.

Кроме того, в состав алгебры включается операция присваивания, позволяющая сохранить в базе данных результаты вычисления алгебраических выражений, и операция переименования атрибутов, дающая возможность корректно сформировать заголовок (схему) результирующего отношения.

Реляционная модель данных представляется достаточно гибкой при формализации технологических процессов. Следует обратить внимание на основной инструмент получения информации из современных систем управления базами данных (СУБД) – язык SQL (Structured Query Language).

SQL является инструментом, предназначенным для обработки и чтения данных, содержащихся в компьютерной базе данных. Как следует из названия, SQL является языком программирования, который применяется для организации взаимодействия пользователя с базой данных. На самом деле SQL работает только с базами данных одного определенного типа, называемых реляционными. Если пользователю необходимо прочитать данные из базы данных, он запрашивает их у СУБД с помощью SQL. СУБД обрабатывает запрос, находит требуемые данные и посылает их пользователю.

Несмотря на то, что чтение данных по-прежнему остается одной из наиболее важных функций SQL, сейчас этот язык используется для реализации всех функциональных возможностей, которые СУБД предоставляет пользователю, а именно:

*Организация данных.* SQL дает пользователю возможность изменять структуру представления данных, а также устанавливать отношения между элементами базы данных.

*Чтение данных.* SQL дает пользователю или приложению возможность читать из базы данных содержащиеся в ней данные и пользоваться ими.

*Обработка данных.* SQL дает пользователю или приложению возможность изменять базу данных, т. е. добавлять в нее новые данные, а также удалять или обновлять уже имеющиеся в ней данные.

*Управление доступом.* С помощью SQL можно ограничить возможности пользователя по чтению и изменению данных и защитить их от несанкционированного доступа.

*Совместное использование данных.* SQL координирует совместное использование данных пользователями, работающими параллельно, чтобы они не мешали друг другу.

*Целостность данных.* SQL позволяет обеспечить целостность базы данных, защищая ее от разрушения из-за несогласованных изменений или отказа системы.

Таким образом, SQL является достаточно мощным языком для взаимодействия с СУБД.

SQL — это не полноценный компьютерный язык типа COBOL, FORTRAN или C. В SQL нет оператора IF для проверки условий, нет оператора GOTO для организации переходов и нет операторов DO или FOR для создания циклов. SQL является *подъязыком* баз данных, в который входит около тридцати операторов, предназначенных для управления базами данных. Операторы SQL *встраиваются* в базовый язык, например COBOL, FORTRAN или C, и дают возможность получать доступ к базам данных. Кроме того, из такого языка, как C, операторы SQL можно посылать СУБД в явном виде, используя *интерфейс вызовов функций.*

Наконец, SQL — это слабо структурированный язык, особенно по сравнению с такими сильно структурированными языками, как C или Pascal. Операторы SQL напоминают английские предложения и содержат «слова-пустышки», не влияющие на смысл оператора, но облегчающие его чтение. В SQL почти нет нелогичностей, к тому же имеется ряд специальных правил, предотвращающих создание операторов SQL, которые выглядят как абсолютно правильные, но не имеют смысла.

SQL — это легкий для понимания язык и в то же время универсальное программное средство управления данными.

Успех языку SQL принесли следующие его особенности:

— независимость от конкретных СУБД;
— переносимость с одной вычислительной системы на другую;
— наличие стандартов;
— одобрение компанией IBM (СУБД DB2);
— поддержка со стороны компании Microsoft (протокол ODBC);
— реляционная основа;
— высокоуровневая структура, напоминающая английский язык;
— возможность выполнения специальных интерактивных запросов;

— обеспечение программного доступа к базам данных;

— возможность различного представления данных;

— полноценность как языка, предназначенного для работы с базами данных;

— возможность динамического определения данных;

— поддержка архитектуры клиент/сервер.

Все перечисленные выше факторы явились причиной того, что SQL стал стандартным инструментом для управления данными на персональных компьютерах, мини-компьютерах и больших ЭВМ.

Таким образом, SQL обеспечивает максимальную гибкость, так как дает базе данных возможность адаптироваться к изменяющимся требованиям, не прерывая работу приложения, выполняющегося в реальном масштабе времени.

В настоящее время разработка базы данных это не просто проектирование таблиц и отношений между ними, но и написание SQL-запросов, обеспечивающих доступ к данным в нужной форме. За счет того, что математические методы, обеспечивающие выполнение запросов в современных СУБД (на основе индексов), хорошо проработаны, программисту не нужно задумываться об оптимизации выборки нужного набора данных. Обычно таким набором данных является денормализованная таблица, позволяющая применять логико-эвристические методы.

Итак, язык SQL является удобным стандартизированным средством общения с системой управления базой данных. Имея такое средство, можно рассмотреть способы использования его в рамках программных приложений на основе реляционных баз данных.

### 1.4.2 Проектирование архитектуры современных программных комплексов

Разработка программных систем не может основываться только лишь на организации хранения данных. Важно умение правильно спроектировать систему в целом, чтобы она была не слишком чувствительна к различным изменениям. При обсуждении способов проектирования программных комплексов и вариантов

выбора типовых решений важно понимать, что двух одинаковых приложений просто не существует и различия в проблемах обусловливают необходимость применения различных средств достижения поставленных целей.

Если задача сводится к простым операциям по отображению информации из базы данных и ее незначительному обновлению, то для ее решения обычно используется архитектура *клиент/сервер (client/server)*. Такие системы содержат два слоя: клиент несет ответственность за отображение пользовательского интерфейса и выполнение кода приложения, а в роли сервера обычно выступает СУБД. Зачастую разработчики пренебрегают простотой архитектуры там, где это нецелесообразно.

Проблемы возникают с усложнением логики предметной области, расширением функциональных возможностей программных комплексов, усложнением алгоритмов вычислений, условий проверок и т. д. Чем сложнее становится логика, тем сложнее поддерживать созданный код: он дублировался, становился громоздким и трудным для восприятия. Как следствие для внесения простейшего изменения приходится внимательно изучать всю программу.

Таблица 1.1

Слои программной системы

| Слой | Функции |
|---|---|
| Представление | Предоставление услуг, отображение данных, обработка событий пользовательского интерфейса (щелчков кнопками мыши и нажатий клавиш), обслуживание запросов HTTP, поддержка функций командной строки и API пакетного выполнения |
| Домен | Бизнес-логика приложения |
| Источник данных | Обращение к базе данных, обмен сообщениями, управление транзакциями и т. д. |

По мере роста популярности систем клиент/сервер набирала силу и парадигма объектно-ориентированного программирования, способствовавшая переносу бизнес-логики на свою платформу. Она же обусловила переход к системной архитектуре с *тремя* слоями (таблица 1.1), в которой слой *представления* отводится пользовательскому интерфейсу, слой *предметной области* предназначен

для описания бизнес-логики, а третий слой представляет *источник данных.* В этом случае удается разнести интерфейс и логику, поместив последнюю на отдельный уровень, где она может быть структурирована с помощью соответствующих объектов.

Концепция *слоев (layers)* — одна из общеупотребительных моделей, используемых разработчиками программного обеспечения для разделения сложных систем на более простые части [64].

Расчленение системы на слои предоставляет целый ряд преимуществ.

- Можно выбирать альтернативную реализацию базовых слоев.
- Каждый слой является удачным кандидатом на стандартизацию.
- Отдельный слой можно воспринимать как единое самодостаточное целое, не особенно заботясь о наличии других слоев.
- Сведение зависимости между слоями позволит вести параллельную разработку программного комплекса несколькими независимыми командами разработчиков.
- Созданный слой может служить основой для нескольких различных слоев более высокого уровня.
- Каждый последующий слой, базирующийся на предыдущем будет проще разрабатывать.

Схема расслоения обладает и определенными недостатками.

- Слои способны удачно инкапсулировать многое, но не все: модификация одного слоя подчас связана с необходимостью внесения каскадных изменений в остальные слои. Классический пример из области корпоративных программных приложений: поле, добавленное в таблицу базы данных, подлежит воспроизведению в пользовательском интерфейсе (в слое представления) и потребует изменений каждом промежуточном слое.
- Производительность системы снижается при наличии избыточных слоев. Несмотря на это, инкапсуляция нижележащих функций зачастую позволяет достичь весьма существенного преимущества. Например,

оптимизация слоя транзакций обычно приводит к повышению производительности всех вышележащих слоев.

Однако самое трудное при использовании архитектурных слоев — это определение содержимого и границ ответственности каждого слоя.

При обсуждении вопросов расслоения программных систем нередко путают понятия *слоя (layer)* и *уровня (tier)*. Часто их употребляют как синонимы, но в большинстве случаев термин *уровень* трактуют, подразумевая *физическое* разделение. Поэтому системы клиент/сервер обычно описывают как двухуровневые (в общем случае «клиент» действительно отделен от сервера физически): клиент — это приложение для компьютера пользователя, а сервер — программа, выполняемая на сетевом сервере. Слои не обязательно должны располагаться на разных машинах. Один слой бизнес-логики может функционировать как на компьютере вместе со слоем пользовательского интерфейса, так и на сервере СУБД. Если база данных локальна, все три слоя находятся в одном приложении, но даже в этом случае они должны оставаться независимыми.

Слой *представления* охватывает все, что касается общения пользователя с системой, то есть пользовательский интерфейс. К главным функциям слоя представления относятся отображение информации и интерпретация вводимых пользователем команд с преобразованием их в соответствующие операции в контексте домена (бизнес-логики) и источника данных.

*Источник данных* — это подмножество функций, обеспечивающих взаимодействие со сторонними системами, которые выполняют задания в интересах приложения. Чаще всего источником данных выступает СУБД. Код в этой части отвечает за обеспечение транзакций, обмен сообщениями, а также управление другими.

*Бизнес-логика* описывает основные функции приложения, предназначенные для достижения поставленной перед ним цели. К таким функциям относятся обработка команд, поступающих от слоя представления, вычисления на основе вводимых и хранимых данных, верификация данных, а также передача информации источнику данных.

Желательно, чтобы бизнес-логика полностью скрывала источник данных от представления. Однако, иногда код представления может обращаться к источнику данных непосредственно. С теоретической точки зрения такой вариант менее безупречен, но с точки зрения практики он более удобен и даже иногда целесообразен: код представления может интерпретировать команду пользователя, активизировать функции источника данных для извлечения информации из базы данных, обратиться к средствам бизнес-логики для анализа этой информации и осуществления необходимых расчетов и только затем вывести пользователю соответствующее сообщение.

Существует также правило, касающееся взаимодействия слоев: не допускается зависимость бизнес-логики и источника данных от уровня представления. Другими словами, в тексте приложения не должно быть вызовов функций представления из кода бизнес-логики или источника данных. Правило позволяет упростить возможность адаптации слоя представления или замены его альтернативным вариантом с сохранением основы приложения. Связь между бизнес-логикой и источником данных, однако, не столь однозначна и во многом определяется выбором типовых решений для архитектуры источника данных.

В настоящее время концептуальная база, а также типовые шаблоны проектирования архитектуры программных комплексов достаточно хорошо описаны в работах М. Фаулера [64, 65], что позволяет создавать достаточно гибкое для поддержки и использования программное обеспечение. Основными критериями хорошей архитектуры являются:

- Эффективность системы (программный комплекс должен выполнять свои функции).
- Гибкость системы (удобство внесения изменений в функциональные возможности программного комплекса).
- Расширяемость системы (возможность добавления новых сущностей и функций в программный комплекс).

- Масштабируемость процесса разработки (возможность сократить срок разработки за счет привлечения разработчиков).
- Тестируемость (возможность обеспечения тестирования модулей программного комплекса).
- Повторное использование (возможность использования разработанных модулей в других программных комплексах).
- Сопровождаемость (удобство и простота понимания программного кода).

Чтобы обеспечить выполнение этих критериев следует стремиться к ослаблению связей между модулями (слоями). Этого можно достичь при помощи шаблонов проектирования: фасад, фабрика, локатор сервисов [65]. Следует заменять прямые зависимости на обмен сообщениями (шаблоны наблюдатель, посредник, команда) или замену прямых зависимостей на синхронизацию через общее ядро.

Все эти приемы достаточно известны и успешно применяются – они позволяют создать архитектуру программного комплекса в целом. В следующем параграфе будет рассмотрен вопрос численных методов решения ресурсных задач.

### 1.4.3 Реализация численных методов решения ресурсных задач

Разработка алгоритмов численного решения задач организации производственных процессов (включая сетевое планирование) на основе логико-эвристического подхода имеет две основных особенности.

1. Организация данных, структур управления, способов оптимизации переборов в общих чертах одинакова для практически всех задач рассматриваемых предметных областей.

2. Проверка ограничений, генерация множества вариантов переборов, выбор типов стратегий управления решением достаточно индивидуальны и всегда специально настраиваются под конкретный класс задач.

В настоящем параграфе будут рассмотрены именно общие технические

решения по организации данных, структур управления, способов оптимизации переборов и по некоторым другим аспектам реализации численных методов решения задач сетевого планирования на основе логико-эвристического подхода.

Прежде всего, это относится к программной реализации следующих конструкций, определенных в параграфе **1.3.**:

— основных множеств многоосновных алгебраических систем (доменов реляционной БД);

— отношений многоосновных алгебраических систем (таблиц реляционной БД);

— задания ограничений, которым должны удовлетворять многоосновные алгебраические системы, и методов проверки выполнимости этих ограничений;

— задание операторов преобразований многоосновных алгебраических систем и организации перебора последовательностей преобразований;

— стратегии оптимизации перебора последовательностей преобразований.

**Организация данных и структур управления.** Отметим сразу, что в программной реализации логико-эвристического подхода данные и структуры управления алгоритмами численного решения задач сетевого планирования представляются близкими типами данных, а именно, **деревьями**, что обеспечивает возможность применения «обратной связи», т. е. данные задачи могут легко преобразовываться в структуры управления и наоборот.

Таким образом, особенности данных задачи могут «управлять» процессом численного решения.

При программировании методов решения задач сетевого планирования приходится работать, в том числе, и с низкоорганизованными структурами данных. Для оперативной памяти – это прямое выделение ее некоторых областей (**куч**, на английском языке ***heap***), а для дисковой памяти – файловые системы, которые являются практически неизбежной расплатой из-за невозможности хранить

в СУБД сложно организованные структуры управления алгоритмами численного решения задач сетевого планирования на основе логико-эвристического подхода.

При работе с оперативной памятью используются команды языка программирования ***C++ calloc*** или ***new***, выделяющие по указателю ***pointer*** область памяти – ***кучу*** (***heap***).

При работе с дисковой памятью файлы создаются командой языка программирования ***C++ Discrip = _creat(const char *File, 0);,*** где файл с именем ***File*** имеет дескриптор или указатель файла ***Discrip.*** Операторы чтения – ***_read***, записи ***_write,*** позиционирования указателя ***lseek***.

Фрагменты программ, иллюстрирующие реализацию представления данных, структур управления и алгоритмов логико-эвристических методов решения сетевого планирования, будут приведены на подмножестве ***C++*** – подобного алгоритмического языка (который и использовался при реализации рассматриваемых здесь программных комплексов).

Для обеспечения эффективной реализации работы с данными, конечно, приходится использовать и стандартные средства, такие как: сортировка, двоичный поиск, хеширование и др. Применение этих средств соответствует описаниям, приведенным в классических книгах Д. Кнута [25, 26].

Структура, задающая вершину дерева (по терминологии Д.Кнута [25]) – ***ссылками «левый сын – правый брат»***

```
struct Tree {
int value;    //значение, приписанное вершине (операция, число, имя и др.)
int down;     //ссылка на первого потомка (-1: отсутствие ссылки);
int right;    //ссылка на следующего брата (-1: отсутствие ссылки);
};
```

Отсутствие ссылки на следующую вершину для компонент ***down*** и ***right*** дерева определяется значением **-1**.

Выделение ***k*** элементов структуры ***Tree*** по указателю ***tree*** задается оператором

***tree = (struct Tree *) calloc (k, sizeof(struct Tree));*** ***(1.3)***

Функция ***sizeof*** задает объем памяти необходимый под размещение одного экземпляра данных типа **struct** ***Tree*** в байтах. Таким образом, оператор ***(1.3)*** задает по указателю ***tree*** область в ***k * sizeof(*****struct** ***Tree)*** байт.

Доступ к ***i***-му экземпляру структуры ***Tree*** осуществляется командой

****( tree + i)***. Например, если необходимо переменной ***val*** (типа ***int***) присвоить значение элемента ***value i***-го экземпляра структуры ***Tree***, определенной оператором ***(1.3)***, то это задается оператором

***val = (tree + i)-> value;*** ***(1.4)***

Эквивалентные формы записи оператора ***(1.4)*** следующие:

***val = * (tree + i).value;*** ***(1.5)***

***val = tree[ i]. value;*** ***(1.6)***

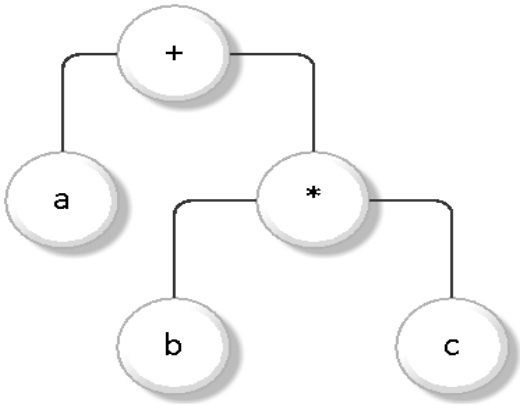
*Оператор* ***(1.6)*** *наиболее наглядно показывает эквивалентность понятий* ***указателей и массивов.***

Рассмотрим конкретный пример представления алгебраического выражения. Выражение ***a+(b*c)*** можно представить в виде следующей таблицы и рисунка, соответствующего графа (дерева) (см. таблица 1.2):

Таблица 1.2

Алгебраическое выражение a + (b * c)

| **Номер** | **Значение** | **Потомок** | **Брат** |
|---|---|---|---|
| **0** | + | **1** | **-1** |
| **1** | **a** | **-1** | **2** |
| **2** | * | **3** | **-1** |
| **3** | **b** | **-1** | **4** |
| **4** | **c** | **-1** | **-1** |



Для представления таблицы понадобится пять экземпляров структуры **struct** ***Tree***, определенной в (1.3).

Предположим, что эти пять экземпляров структуры **struct** ***Tree*** определены командой:

***tree = (struct Tree *) calloc (5, sizeof(struct Tree));*** ***(1.7)***

Тогда таблица, соответствующая выражению ***a+(b*c)***, будет иметь следующее представление:

***( tree + 0)-> value = Код(+);***

***( tree + 0)-> down = 1;***

***( tree + 0)-> right = -1;***

***( tree + 1)-> value = Код(a);***

***( tree + 1)-> down = -1;***

***( tree + 1)-> right = 2;***

***( tree + 2)-> value = Код(*);***

***( tree + 2)-> down = 3;***

***( tree + 2)-> right = -1;***

***( tree + 3)-> value = Код(b);***

***( tree + 3)-> down = -1;***

***( tree + 3)-> right = 4;***

***( tree + 4)-> value = Код(c);***

***( tree + 4)-> down = -1;***

***( tree + 4)-> right = -1;***

К основным алгоритмам обработки данных численного решения задач сетевого планирования на основе логико-эвристического подхода, представленных ***деревьями,*** относятся:

— обход дерева «***в глубину – направо***» (Д.Кнут указывает [25], что данный алгоритм соответствует системе передачи титулов у английской аристократии);

— проверка вхождения (или пересечения) одних ***деревьев*** в другие (с другими);

— перестройки ***деревьев*** (подстановки значений переменных, замены ***поддеревьев*** на другие, проведение сокращений, например, из-за вычисления значений термов, и по многим другим причинам, связанным в основном с реализацией основных технических приемов сокращения переборов, численного решения задач сетевого планирования на основе логико-эвристического подхода, таких как «**просмотр вперед**», «**глубокий возврат**», «**быстрая проверка ограничений**»;

— многие другие, включая (Д.Кнут [25]): прохождение в прямом, обратном и концевом порядке, прошивка ***дерева***, обработка соотношений эквивалентности, включение в прошитое ***дерево*** и пр.

К достоинствам вышеописанной структуры ***деревьев*** можно отнести (как уже указывалось выше и иллюстрации чего мы приступаем в этом и других разделах):

1) В решателях подобную структуру используют не только для организации данных, но и как структуру управления.

2) Перестройки и рекомбинации данных осуществляются с минимальными затратами алгоритмических ресурсов, а при определенных условиях, и требуемой оперативной памяти.

3) «Сборка мусора» для данных, организованных по типу ***деревьев,*** осуществляется чуть сложнее, чем в списковом случае (далее это будет проиллюстрировано на примере организации ***списка свободных вершин дерева***).

Приведем описание программной реализации конструкций, определенных в параграфе **1.3.** для многоосновных алгебраических систем, заданных в форме

$$M = < A_1 , A_2 ,\ldots, A_s ; p_1 , \ldots, p_k >,$$

где $A_1 , A_2 ,\ldots, A_s$ – основные множества и $p_1 , \ldots, p_k$ – отношения (предикаты), где, так же, определены ограничения $R_1, R_2,\ldots, R_m$.

**1. Задание основных множеств** $A_1$ **, …,** $A_s$ многоосновной алгебраической системы $M$ (доменов реляционной БД) в оперативной памяти осуществляется выделением, соответственно, областей (**куч**)

$Set_1$ = (struct $Set_1$ *) calloc ($k_1$, sizeof(struct $Set_1$));

$Set_2$ = (struct $Set_2$ *) calloc ($k_2$, sizeof(struct $Set_2$));

……..……..

$Set_s$ = (struct $Set_s$ *) calloc ($k_s$, sizeof(struct $Set_s$));

где, соответственно, структуры $Set_1$, $Set_2$, …, $Set_s$ задают необходимые свойства элементов основных множеств $A_1$ , $A_2$ ,…, $A_s$, соответственно,

$k_1$, $k_2$, …., $k_s$ – размерности основных множеств $A_1$ , $A_2$ ,…, $A_s$.

Доступ к $i$-му элементу основных множеств $A_1$ , $A_2$ ,…, $A_s$, соответственно, осуществляется операторами ($Set_1$ + $i$), ($Set_2$ + $i$),…, ($Set_s$ + $i$). Доступ к компоненте *value* $i$-го элемента основных множеств $A_1$ , $A_2$ ,…, $A_s$, соответственно, осуществляется операторами

($Set_1$ + $i$) -> value, ($Set_2$ + $i$) -> value,…, ($Set_s$ + $i$) -> value.

2. **Задание отношений** $p_1$ **, …,** $p_k$ **многоосновных алгебраических систем** (таблиц реляционной БД) в оперативной памяти осуществляется выделением, соответственно, областей (**куч**)

$Relat_1$ = (struct $Relat_1$ *) calloc ($m_1$, sizeof(struct $Relat_1$));

$Relat_2$ = (struct $Relat_2$ *) calloc ($m_2$, sizeof(struct $Relat_2$));

……..

$Relat_k$ = (struct $Relat_k$ *) calloc ($m_k$, sizeof(struct $Relat_k$));

где, соответственно, структуры $Relat_1$, $Relat_2$, …, $Relat_k$ служат для задания кортежей элементов основных множеств $A_1$ , $A_2$ ,…, $A_s$, составляющих отношения $p_1$ , …, $p_k$ ,соответственно, а числа $m_1$, $m_2$, …., $m_k$ – определяют количество

кортежей отношений $p_1$ , …, $p_k$ , определенных на основных множествах $A_1 , A_2 ,…, A_s$, т. е. предполагаем, что отношения

$$p_1 , …, p_k \subset A_1 x A_2 x…x A_s.$$

Доступ к ***i***-му картежу отношений $p_1$ , …, $p_k$ , определенных на основных множествах $A_1 , A_2 ,…, A_s$ осуществляется, соответственно, операторами ***(Relat*** $_1$ ***+ i), (Relat*** $_2$ ***+ i),…, (Relat*** $_k$ ***+ i).*** Доступ к ***j*** – му элементу ***i***-го кортежа отношений $p_1$ , …, $p_k$ , определенных на основных множествах $A_1 , A_2 ,…, A_s$, соответственно, осуществляется операторами ***(Relat*** $_1$ ***+ i) -> value***$_j$, ***(Relat*** $_2$ ***+ i) -> value***$_j$***,…, (Relat*** $_k$ ***+ i) -> value***$_j$.

Для обеспечения быстрого доступа к кортежам отношений $p_1$ , …, $p_k$, определенных на основных множествах $A_1 , A_2 ,…, A_s$, используется **древовидная** организация данных, формируемая по принципу расположения ***j*** – ых элементов кортежей на ***j*** – ом этаже древовидной структуры.

Пусть явное определение множества кортежей отношений $p_1$ , …, $p_k$ , определенных на основных множествах $A_1 , A_2 ,…, A_s$ имеет вид:

$$p_1 = \{ (a_{11}, a_{12},…, a_{1n}) | \ a_{11} € A_1 , a_{12} € A_2,…, a_{1n} € A_n\},$$

$$p_2 = \{ (a_{21}, a_{22},…, a_{2n}) | \ a_{21} € A_1 , a_{22} € A_2,…, a_{2n} € A_n\},$$

………………………………………………..

$$p_k = \{ (a_{k1}, a_{k2},…, a_{kn}) | \ a_{k1} € A_1 , a_{k2} € A_2,…, a_{kn} € A_n\},$$

Задание в оперативной памяти древовидных структур ***Tre***$_1$ , ***Tre***$_2$ ,…, ***Tre***$_k$ , задающих отношения $p_1$ , …, $p_k$, определенных на основных множествах $A_1 , A_2 ,…, A_s$, осуществляется выделением, соответственно, областей (**куч**)

***Tr*** $_1$ ***= (struct Tree *) calloc (n * m***$_1$***, sizeof(struct Tree));***

***Tr*** $_2$ ***= (struct Tree *) calloc (n * m*** $_2$***, sizeof(struct Tree));***

………………………………………………..

***Tr*** $_k$ ***= (struct Tree *) calloc (n * m*** $_k$***, sizeof(struct Tree)),***

где структура **struct *Tree*** определена выше для примера ***2.1.1***.

Рассмотрим для произвольного ***i***, где ***1 <= i <= k,*** построение по индукции от числа кортежей $m_i$ в отношении $p_i$ древовидной структуры $Tre_i$. , задающей отношение $p_i$, определенное на основных множествах $A_1, A_2, \ldots, A_s$.

**Основание индукции.** Пусть первый кортеж отношения $p_i$ имеет вид $(a_1, a_2, \ldots, a_s)$, где $a_1 \in A_1, a_2 \in A_2, \ldots, a_s \in A_s$ и элементы $a_1, a_2, \ldots, a_s$ имеют индексы, соответственно, $t_1, t_2, \ldots, t_s$ в представлении $(Set_1 + t_1), (Set_2 + t_2), \ldots, (Set_s + t_s)$. Тогда начальные присвоения для элементов кучи по адресу (указателю) $Tr_i$ будут:

— формирование 0-го этажа (корень дерева)

$(Tr_i + 0)\text{->}value = -1; (Tr_i + 0)\text{->}down = 1; (Tr_i + 0)\text{->}right = -1;$

— формирование 1-го этажа

$(Tr_i + 1)\text{->}value = t_1; (Tr_i + 1)\text{->}down = 2; (Tr_i + 1)\text{->}right = -1;$

— формирование 2-го этажа

$(Tr_i + 2)\text{->}value = t_2; (Tr_i + 2)\text{->}down = 3; (Tr_i + 2)\text{->}right = -1;$

……

$(Tr_i + s)\text{->}value = t_s; (Tr_i + s)\text{->}down = -1; (Tr_i + s)\text{->}right = -1;,$

причем 1-ый свободный элемент $Fr1\_Tr_i$ кучи по адресу (указателю) $Tr_i$ будет $s + 1$.

**Индукционный шаг.** Пусть очередной кортеж отношения $p_i$, включаемый в древовидную структуру, имеет вид $(b_1, b_2, \ldots, b_s)$, где $b_1 \in A_1, b_2 \in A_2, \ldots, b_s \in A_s$ и элементы $b_1, b_2, \ldots, b_s$ имеют индексы, соответственно, $u_1, u_2, \ldots, u_s$ в представлении $(Set_1 + u_1), (Set_2 + u_2), \ldots, (Set_s + u_s)$.

Предположим далее, первые $l$ ($l < s$ обязательно!) элементов $b_1, b_2, \ldots, b_l$ совпадают с некоторой цепочкой $w_1, w_2, \ldots, w_l$, идущей от начальной (нулевой) вершины древовидной структуры. Более точно, это означает следующее:

— вершина $w_1$ непосредственный потомок начальной (нулевой) вершины;

— вершина $w_{o+1}$ непосредственный потомок вершины $w_o$;

$(Tr_i + w_1)\text{->}value = u_1;$

$(Tr_i + w_2)\text{->}value = u_2;$

……..….

$(Tr_i + w_l)\text{->}value = u_l;$

— если $ww$ самый правый непосредственный потомок вершины $w_l$, то

$(Tr_i + ww)\text{->}right = Fr1\_Tr_i;$

$(Tr_i + Fr1\_Tr_i)\text{->}right = -1;$

$(Tr_i + Fr1\_Tr_i)\text{->}value = u_{l+1};$

……

$(Tr_i + Fr1\_Tr_i + s - l)\text{->}right = -1;$

$(Tr_i + Fr1\_Tr_i + s - l)\text{->}value = u_s;,$

причем 1-ый свободный элемент $Fr1\_Tr_i$ кучи по адресу (указателю) $Tr_i$ будет обновлен присвоением $Fr1\_Tr_i = Fr1\_Tr_i + s - l;$.

Индукционный шаг построения древовидной структуры выполнен.

3. **Задание ограничений $R_1, R_2,\ldots, R_m$, которым должны удовлетворять многоосновные алгебраические системы**, и методы проверки выполнимости этих ограничений. Основные отличия программных решений этого пункта от предыдущего заключены в том, что ограничения $R_1, R_2,\ldots, R_m$ не имеют явного задания как конкретные отношения, определенные на основных множествах $A_1, A_2, \ldots, A_s$.. Если от процедурного представления ограничений $R_1, R_2,\ldots, R_m$ удается перейти перечислению кортежей, которые задают хотя бы некоторые из ограничений $R_1, R_2,\ldots, R_m$, то дальнейшая работа с ними практически не отличаются от задания древовидными структурами отношений $p_1, \ldots, p_k$, с последующей организацией их вычислимости.

Основные проблемы обеспечения быстрой вычислимости ограничений $R_1, R_2,\ldots, R_m$ две:

1) количество кортежей, которые задают некоторые из ограничений $R_1, R_2, \ldots, R_m$, может быть **очень большим**, что не позволяет все их одновременно держать в оперативной памяти.

2) при преобразованиях многоосновной алгебраической системы необходимо **редактировать совокупности кортежей**, представляющих ограничений $R_1, R_2, \ldots, R_m$, что резко усложняет все технические задачи построения древовидных структур.

Достаточно часто **первая проблема** может быть решена введением так называемых **декубов**, т. е. всеобщих значений (*) элементов из какого-либо основного множества $A_1, A_2, \ldots, A_s$ для кортежей. Расплатой за ввод **декубов** является переход к более сложной древовидной структуре, где вершина (узел) задается в виде

```
struct Tree_decube {
int value; //значение, приписанное вершине (операция, число, имя и др.)
int down; //ссылка на первого потомка (-1: отсутствие ссылки);
int right; //ссылка на следующего брата (-1: отсутствие ссылки);
int decube; //ссылка на декуб (-1: отсутствие ссылки);
};
```

Кроме того, сложность вычислений для древовидной структуры с декубами не линейная, а квадратичная.

Полученная древовидная структура называется **деревом решения** и позволяет проверять все ограничения одновременно, т.е перебор по проверке ограничений исчезает.

**3. Задание операторов преобразований многоосновных алгебраических систем и организации перебора** последовательностей преобразований наиболее выпукло можно представить на примере задачи планирование расписания учебных занятий (параграф **1.5.1.**), где каждому занятию из $ZZ \subseteq P \times C \times D$ необходимо сопоставить ресурс, т. е. элемент из $R: A \times M$ (аудиторию и время проведе-

ния занятий). Таким образом, преобразование многоосновной алгебраической системы состоит в построении (формировании) одного кортежа отношения (либо его «уничтожения»). **Организация перебора** преобразования состоит в процедуре выбора **элементов**, которым сопоставляется **ресурс.**

**4. Стратегии оптимизации перебора последовательностей преобразований** состоят в возможно более быстром решении в каждой точке пространства поиска трех фундаментальных задач:

а) просмотру вперед (checking forward) для уменьшения количества применяемых преобразований;

б) определению точки возврата для тупика (intelligent backtracking или глубокий возврат по принятой у ряда авторов в России терминологии);

в) проверки выполнимости ограничений на многоосновной алгебраической системе, полученной после выполнения выбранного преобразования (применение демонов для определения невязок – терминология специалистов по системам искусственного интеллекта).

Оптимизация переборов для этих трех задач для каждой предметной области производится весьма специфическим образом, но для рассматриваемых в диссертационной работе задач организации производственных процессов есть общие черты, которые будут описаны ниже.

Оптимизация, реализуемая просмотром вперед (checking forward) для уменьшения количества применяемых преобразований, достигается уменьшением ресурса при применении каждого преобразования (большинство задач сетевого планирования происходит при убывающем ресурсе). Для других задач используется стратегия «**первой неудачи**» (first fail) [73], которая, как правило, существенно уменьшает пространство перебора следующего шага решения.

Оптимизация, реализуемая определением точки возврата для тупика (intelligent backtracking или глубокий возврат по принятой у ряда авторов в России терминологии), для задач сетевого планирования с убывающим ресурсом достигается рассмотрением, либо правой точки, либо левой точки, где произошло преобразование с объектами тупика. Если возврат осуществляется с левой точки, то это со-

ответствует стандартному глубокому возврату, а если с правой точки, то глубокому возврату со стратегией «чем хуже, тем лучше», где решение может быть найдено быстрее, но данная стратегия не полна (может пропустить возможные решения).

Оптимизация, реализуемая проверкой выполнимости ограничений, достигается для всех задач организации производственной деятельности использованием деревьев решения (см. выше пункт 3. **Задание ограничений $R_1, R_2,\ldots, R_m$**).

# 2 Математические модели для решения задач в дорожной отрасли и реализация программного комплекса мониторинга и поддержки управления сетью автомобильных дорог

## 2.1 Математические модели для решения задач поддержки управления сетью автомобильных дорог

### 2.1.1 Сетевое планирование работ по нормативному содержанию сети автодорог

Задача сетевого планирования работ по нормативному содержанию сети автомобильных дорог заключается в расчете стоимости и планировании сроков исполнения работ по содержанию сети автомобильных дорог с минимальными затратами (временными, ресурсными и стоимостными).

Опишем математическую модель территориально распределенного комплекса – сети автомобильных дорог ([42]). В качестве такой модели будем использовать ориентированный нагруженный граф $G$ с координатной системой на ребрах. Данный граф является двуосновной алгебраической системой

$$\boldsymbol{G = \{V, E, \sigma\},}$$

где $V$ – множество вершин, $E$ – ориентированные ребра (участки автомобильных дорог, включая необслуживаемые), $\sigma$ – совокупность отношений и операций, определенных на вершинах и ребрах графа (сигнатура алгебраической системы).

Вершины графа в основном являются примыканиями и пересечениями, либо концом/началом автомобильной дороги, т. е. концом/началом учетной единицы реестра обслуживаемых автомобильных дорог. Также вершинами могут быть муниципальные образования, которые сами обслуживают свои дороги, в этом случае вершине должно быть сопоставлено расстояние, которое необходимо проезжать по дорогам этого муниципального образования.

Выделим основные отношения алгебраической системы $G$, входящими в сигнатуру $\sigma$.

*Met*(*e*, *Name*, *start*, *end*), здесь *e* – ребро графа *G*; *Name* – наименование автомобильной дороги (титул); *start*, *end* – пикетаж – числа – соответственно, нача-

ла и конца ребра. Это отношение в дальнейшем будем называть определяющим титулы и пикетаж участков автомобильных дорог.

$$\boldsymbol{Rel(e_1, e_2, v, d, p),}$$

где $e_1$, $e_2$ – ребра графа, примыкающие к вершине $v$; $d$ – расстояние между участками $e_1$ и $e_2$; $p$ – совокупность других дополнительных параметров для описания свойств вершины графа $v$. Это отношение в дальнейшем будем называть задающим связность участков автомобильных дорог.

$$\boldsymbol{Inc(s, e, c, f),}$$

где $s$ из $S$ – подмножество частей ребер графа $G$, объединенные по административному признаку (расположены в одном районе), или хозяйственному (обслуживаются одним эксплуатационным предприятием) или какому-то другому, $e$ – ребро, причем, если $Met(e, Name, start, end)$, то $c(f) \geq start$ (соответственно, $c(f) \leq end$). В случае необходимости явно представить элементы подмножества $s$ из $S$ будем использовать несколько упрощенное обозначение

$$\boldsymbol{s = \{e_1, e_2, \ldots, e_k\}.}$$

Подмножество $s$ из $S$, будем называть маршрутом, если для любого $i < k$ ребра $e_i$ и $e_{i+1}$ связаны отношением $Rel$.

Автомобильная дорога [42] состоит из четырех основных конструктивных элементов:

- земляное полотно и полоса отвода,
- дорожная одежда,
- искусственные сооружения,
- обстановка и благоустройство.

В таблице 2.1 указаны основные дорожные объекты, над которыми возможно проведение ремонтных работ.

Таблица 2.1

Основные объекты и характеристики автомобильной дороги

| Тип объекта | Наименование объекта | Характеристика для назначения работ | Характеристика объема |
|---|---|---|---|
| Весенне-летне-осеннее содержание | | | |
| ДОРОЖНАЯ ОДЕЖДА | Проезжая часть | Тип покрытия | Длина, Ширина |
| ЗЕМЛЯНОЕ ПОЛОТНО И ПОЛОСА ОТВОДА | Разделительная полоса | Конструкция | Длина, Ширина |
| | Обочина | Тип укрепления | Длина, Ширина |
| | Откос | Тип конструкции | Высота, Угол, Длина |
| | Водоотводные сооружения | Тип конструкции | Длина |
| ИСКУССТВЕННЫЕ СООРУЖЕНИЯ | Тротуар | Тип покрытия | Площадь |
| | Ограждения | Тип конструкции | Длина |
| | Проезжая часть | Тип покрытия | Длина, Ширина |
| | Водопропускная труба | Тип конструкции | Длина, площадь поперечного сечения |
| ОБСТАНОВКА И БЛАГОУСТРОЙСТВО | Дорожные знаки | Номер по ГОСТ | Площадь щитка |
| | Сигнальные столбики | Тип конструкции | Количество |
| | Ограждения | Тип конструкции | Высота |
| | Павильоны | Тип конструкции | Длина |
| | Тротуар | Тип покрытия | Ширина |
| | Бордюр | Тип конструкции | Длина |
| | Разметка | Номер по ГОСТ | Длина |
| | Площадки отдыха и стоянки | Тип конструкции | Площадь |
| | Посадочные площадки | Тип конструкции | Площадь |

Продолжение таблицы 2.1

| Тип объекта | Наименование объекта | Характеристика для назначения работ | Характеристика объема |
|---|---|---|---|
| Зимнее содержание | | | |
| ДОРОЖНАЯ ОДЕЖДА | Проезжая часть | Тип покрытия | Длина, Ширина |
| | Проезжая часть | Опасный участок | Длина, Ширина |
| | Проезжая часть | Снегозаносимый участок | Длина, Ширина |
| ЗЕМЛЯНОЕ ПОЛОТНО И ПОЛОСА ОТВОДА | Разделительная полоса | Тип конструкции | Длина, Ширина |
| | Обочина | Тип конструкции | Длина, Ширина |
| ИСКУССТВЕННЫЕ СООРУЖЕНИЯ | Тротуар | Тип покрытия | Площадь |
| | Проезжая часть | Тип покрытия | Длина, Ширина |
| | Водопропускная труба | Тип конструкции | Длина, площадь поперечного сечения |
| ОБСТАНОВКА И БЛАГОУСТРОЙСТВО | Павильоны | Тип конструкции | Площадь |
| | Дорожные знаки | Номер по ГОСТ | Площадь щитка |
| | Сигнальные столбики | Тип конструкции | Количество |
| | Ограждения | Тип конструкции | Длина |

Определим еще одно основное множество объектов исполнения работ по содержанию *Obj* со отношениями Dis и ProperObj.

***Dis(o, e, begin, end, p),***

где *o* – объект из *Obj*, *e* – ребро графа, *begin, end* – пикетаж, *p* – дополнительные параметры уточнения дислокации объекта (дислокация).

***ProperObj(o, type, name, p),***

где *o* – объект из *Ob*, *type*, *name* – тип и имя объекта, *p* – дополнительные параметры, обеспечивающие связь с работами, которые можно назначать для данного объекта.

Основной задачей содержания автомобильных дорог является планирование работ с наименьшими затратами.

Необходимо определить еще одно основное множество работ *Works* по содержанию объектов со следующим отношением:

***ProperWork(w, time, type, name, p),***

где *w* из *Works*, *time* – период проведения работ, *type*, *name* – тип и имя объекта, *p* – дополнительные параметры, обеспечивающие связь с объектами для которых можно назначать данные работы.

Для некоторых работ определены укрупненные единичные расценки (УЕР), задающие временные, ресурсные и стоимостные показатели для данного объема работ:

***OneCost(w, cap, clock, worker, tech, mater, money),***

где *w* из *Works*, *cap* – объем работ, *clock* – время, необходимое для исполнения данного объема работ, *worker* – необходимые рабочие, *tech* – необходимые средства механизации, *mater* – объем и стоимость материалов, *money* – стоимость работ.

Планом работ назовем результат применения регламента работ к совокупности участков дорог. План работ представляется отношением

***PlanWork(r, begin, end, o, num, w),***

где объект *o* участка дороги *r* (*begin*, *end* – дислокация), связанный с работой *w* отношениями *ProperObj* и *ProperWork*, *num* – объем работ на объекте *o*, причем, если работа *w* имеет единичную расценку, то стоимость выполнения работы *w* на объекте *o* составляет *money* * *num*, а время выполнения работы равно *clock* * *num*.

Введем множество ресурсов

***Resource = Time x Worker x Tech x Mater,***

где *Time* – календарные сроки выполнения работ, *Worker* – совокупность рабочих, *Tech* – совокупность средств механизации, *Mater* – совокупность материалов.

Реализацией (сетевым графиком выполнения) плана работ будем называть выделение ресурсов, т. е. определение времени исполнения работ, средств механизации, рабочих и материалов. Запишем отношение сетевого графика выполнения работ

***Schedule(e, beign, end, o, num, w, clock, worker, tech, mater), (2.1)***

где параметры e*, begin, end, o, num, w* – те же, что и в отношении *PlanWork*, *clock* – начало и конец календарных сроков проведения работ, *worker*, *tech*, *mater* – элементы множеств *Worker*, *Tech*, *Mater*.

Это отношение должно удовлетворять ряду ограничений, связанных с планированием с убывающим ресурсом (любые две записи отношения *Schedule*, связывающие одно и то же средство механизации или рабочего, не могут иметь пересекающееся календарное время по срокам начала и конца $clock_1$ и $clock_2$, соответственно) и учетом затрат на доставку средств механизации и рабочих, а для некоторых работ и материалов (любые две записи отношения *Schedule*, связывающие одно и то же средство механизации или рабочего, не могут иметь дислокации, недостижимые по времени перехода равному разности конца $clock_1$ и начала $clock_2$).

Определим тариф на доставку средств механизации и рабочих, как функцию

***money = Tariff(clock, object),***

где *clock* – время доставки, *object* – наименование объекта доставки (средство механизации или рабочий), *money* – стоимость доставки за данное время.

Теперь можно определить затраты на проведение работ $CostSched_{work}$ (без стоимости затрат на доставку средств механизации и рабочих) как

$$CostSched_{work} = money_1 * num_1 + \ldots + money_k * num_k,$$

где $money_i$ – единичная расценка, соответствующая работе $w_i$.

Для определения затрат на доставку средств механизации и рабочих предположим, что график отношения *Schedule* состоит из смежных работ *w*,

упорядоченных по времени и проводимых в течение одного рабочего дня *day*, ресурсы данной работы будем обозначат *r*. Тогда затраты на доставку средств механизации и рабочих будут равны

$$CostSched_{object,day} = CostWay(r_1, b_1, object) + Tariff(clock_2 - clock_1, object) + \ldots + Tariff(clock_k - clock_{k-1}, object) + CostWay(r_k, e_k, object), \quad (2.2)$$

где первое и последнее слагаемое соответствуют затратам на доставку средств механизации и рабочих на первый объект работы и, соответственно, с последнего объекта работы на место постоянной дислокации. Затраты $CostSched_{transp}$ на доставку средств механизации и рабочих будут равны сумме $CostSched_{object,day}$ по всем ресурсам (*worker, tech*).

Очевидным образом посчитаем общие затраты *CostSched* на выполнение сетевого графика работ:

$$CostSched = CostSched_{work} + CostSched_{transp}. \quad (2.3)$$

Отметим, что первое слагаемое в этой формуле полностью определяется регламентом, что не позволяет применять математические методы минимизации расходов на проведение работ. Второе же слагаемое $CostSched_{transp}$ напротив определяется выбранными маршрутами доставки средств механизации и рабочих и позволяет использовать нетрадиционные методы оптимизации перебора: просмотр вперед (отсечение тупиковых вариантов) и переход на определенное количество итераций назад (глубокий возврат) для минимизации расходов.

Рассмотрим алгоритмическую сложность построения рациональных сетевых графиков (2.1), удовлетворяющих бинарным ограничениям, связанных со временем проведения работ.

Бинарные ограничения порождаются, как минимум, по двум причинам:

1. Некоторые виды работ должны выполняться последовательно (сначала одни, а потом другие, достаточно вспомнить составляющие работы ямочного ремонта или ликвидацию зимней скользкости после снегопада).

2. Как уже отмечалось, уменьшение общих затрат *CostSched* (2.3) возможно только по транспортной составляющей $CostSched_{transp}$. Достаточно рационально

ввести порядок по времени проведения однотипных работ (смотри формулу (2.2)) в зависимости от расстояния дислокации объекта от предприятия, проводящего работы, где проводятся данные работы (т. е. работы, проводимые ближе к предприятию, должны проводиться раньше).

Задача построения рациональных сетевых графиков (2.1), удовлетворяющих бинарным ограничениям, связанных со временем проведения работ, по алгоритмической сложности NP – полна. При достаточной степени абстракции от специфики создания сетевых графиков содержания и текущего ремонта автомобильных дорог и их обустройства, данную задачу можно рассматривать как составление расписаний, удовлетворяющих бинарным отношениям частичного порядка на множестве работ

$$Works = \{w_1, w_2, \ldots, w_n\} \qquad (2.4)$$

Определим явно бинарное отношение $\Xi$ частичного порядка на множестве работ ***Works*** **(4)**

$$\Xi = \{(w_i, w_j) \mid w_i, w_j \text{ из } Works\} \qquad (2.5)$$

Для точного доказательства верхней границы алгоритмической сложности рассматриваемой задачи (включения в класс **NP** – недетерминированные полиномиальные) необходимо построение алгоритма решения полиномиальной сложности на недетерминированной машине Тьюринга (НДТМ), которую, следуя [34], определим, как некое устройство (черный ящик) способное выполнять две группы операций:

1) +, –, *, /, ИЛИ, И, ЧИТАТЬ, ПИСАТЬ, ЕСЛИ … ТО….., ПОВТОРЯТЬ;

2) ВЫБОР [E], что означает возможность перебора всех элементов множества E (считается одной операцией и для включения в класс **NP** необходимо, чтобы множество E имело полиномиальное число элементов относительно параметров размерности задачи).

Отметим, что детерминированная машина Тьюринга (ДТМ) имеет только первую группу операций.

Включение задачи в класс **NP** следует из полиномиальной мощности множества элементов $\Xi$ частичного порядка относительно $n$ (действительно, таких пар не более чем $C_n^2$ **(5)**), что позволяет использовать оператор ВЫБОР [$\Xi$] для составления программы для НДТМ проверки выполнимости ограничений на плане работ.

Более сложно доказывается NP – полнота, где требуется перестройка частичных порядков в графы и поиск примеров для интерпретации на этих графах любых 3-НКФ формул (3-НКФ – формулы исчисления высказываний нормальной конъюнктивной формы с дизъюнкциями, содержащими не более трех булевых переменных или их отрицаний, проверка выполнимости этих формул ***NP*** – полная задача [34]).

Пусть дана произвольная 3-НКФ формула $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_k$, где дизъюнкция $C_i = x_{i1} \,V\, x_{i2} \,V\, x_{i3}$, причем $x_{ij}$, либо переменная, либо ее отрицание (литерал).

Необходимый для доказательства граф будет иметь в качестве вершин совокупность элементов $(x, i)$, где $x$ – литерал $i$-го члена конъюнкции 3-НКФ формулы $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_k$. Определим теперь ребра графа следующим образом: вершины $(x, i)$ и $(y, j)$ связаны ребром если:

a) $i \neq j$ (т. е. литералы находятся в разных членах конъюнкции);

b) литералы $x$ и $y$ не являются взаимным отрицанием, т. е. исключается $x = \neg\, y$ или $y = \neg\, x$.

Индукцией по $k$ можно проверить, что 3-НКФ формула $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_k$, выполнима тогда и только тогда, когда построенный выше граф может быть достроен до наличия гамильтонова цикла.

### 2.1.2 Нахождение областей обслуживания ресурсных баз сети автодорог

Рассмотрим далее задачу рациональной организации областей ответственности структурных служб. Дислокация баз хранения противогололедного материала (ПГМ) относительно участков дорог, представляется в виде множества расстояний от базы до некоторого адреса на дороге. Одним из важнейших моментов

в задаче является то, что необходимо посыпать не всю дорогу, а только опасные участки. Нужно сказать, что поиск опасных участков дороги, как задача, стоит особняком. По геометрическим характеристикам (резкий поворот, большой уклон, ограничение видимости), а также исходя из нормативных требований (посыпание пешеходных переходов, карманов автобусных остановок) можно отыскать нужные участки.

Задача по отысканию областей обслуживания ресурсных баз ПГМ входит в класс задач по декомпозиции графа – разбиения множества элементов графа на классы. Задачи подобного рода возникают при адаптивном управлении сложными системами с дискретной эволюционирующей структурой, которые часто сводится к агрегации системы на заданное число подсистем таким образом, чтобы связи между подсистемами были минимальными.

Так, например, при производстве современного электронного оборудования особое место занимает разработка печатных плат и микросхем. Одной из задач, которая требует решения при их конструировании, является задача разбиения принципиальных схем устройств на составные части (подсхемы, функциональные блоки). Типовые элементы принципиальной схемы должны таким образом быть распределены по отдельным подсхемам, чтобы число внешних соединений было как можно меньше. Это связано с целью повышения надежности схемы, уменьшения влияния наводок, повышения технологичности и простоты конструктивного оформления.

Рассмотрим общую постановку задачи. Зададим неориентированный граф

$$G(V, E) = \langle V, E \rangle,$$

где $\mathbf{V}$ – множество вершин, $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ – множество ребер, $|\mathbf{V}| = \mathbf{n}$ – степень графа.

Пространством решений служит множество всевозможных разбиений графа на непересекающиеся подграфы. Требуется определить разбиение множества вершин $\mathbf{V}$ графа $\mathbf{G(V, E)}$ на $\mathbf{k}$ подмножеств – $(\mathbf{V_1,\ldots,V_k})$ таким образом, чтобы для кусков графа $\mathbf{G_1(X_1,V_1)}$, …, $\mathbf{G_k(X_k,V_k)}$ выполнялись следующие требования:

$$V_i \cap V_j = \varnothing,\ i \neq j,\ i, j = 1 \ldots k;$$

$$\bigcup_{i=1}^{k} V_i = V.$$

Рассмотрим построение функционала и налагаемых ограничений для данной задачи.

Очевидно, вершинами графа является множество опасных участков

$o_1, o_2, \ldots o_n$, для посыпки каждого из которых требуется соответственно

$v_1, v_2, \ldots v_n$ м$^3$ материала,

$v_i = p \cdot l_i \cdot w_i$, где **p** – норма посыпки, $l_i$ и $w_i$ – длина и ширина посыпаемого опасного участка.

Зная расстояние от базы до дороги и дислокацию каждого из участков, получаем расстояния до середины участков $d_1, d_2, \ldots d_n$.

Введем матрицу соответствия опасных участков и баз

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & & \ldots & \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & & \ldots & a_{nm} \end{pmatrix}.$$

Элемент $\mathbf{a_{ij}} = \{1, 0\}$ определяет обслуживается (1) или не обслуживается (0) **i**-ый участок **j**-ой базой. Нахождение именно матрицы **A** и будет решением задачи. Легко видеть, что общее число допустимых решений задачи будет $\mathbf{m^n}$. Действительно, каждый из n опасных участков может принадлежать какой-либо из m баз.

Учитывая, что требуется минимизировать затраты на доставку материала до участка, можно ввести функционал, учитывающий средние дальности возки и количества возок для каждой базы

$$F = \sum_j \left[ \frac{\sum_i a_{ij} \cdot v_i}{v_б} \right]^+ \cdot \frac{\sum_i a_{ij} \cdot v_i \cdot d_{ij}}{\sum_i a_{ij} \cdot v_i} \to \min, \qquad \textbf{(2.6)}$$

где $\mathbf{v_б}$ – объем бункера машины, а $[\ ]^+$ – операция округления числа до большего целого.

Формула средней дальности возки используется для вычисления стоимости работ по посыпке дороги ПГМ. Именно поэтому можно говорить, что функционал (6) обеспечивает экономическую интерпретацию задачи.

Естественно, наложить некоторые ограничения:

$\sum_j a_{ij} = 1, \forall i = 1,..., n$ – один участок принадлежит только одной базе,

$\sum_i a_{ij} v_i \le Vo_j, \forall j = 1,..., m$ – ограничение на кол-во материала на базе (**Voj** – общее кол-во материала на базе),

также можно ввести ограничение на количество механизмов, находящихся в распоряжении базы и на время проведения работ

$$\frac{2 \cdot F \cdot \upsilon_{cp}}{m_j} \le T, \forall j = 1,..., m,$$

где $\upsilon_{ср}$ – средняя скорость передвижения транспорта, $\mathbf{m_j}$ – кол-во средств механизации на базе, **T** – время.

После получения разбиения опасных участков по базам ПГМ, можно попытаться разбить пограничные участки. В этом случае возможен вариант, когда одна часть участка относится к одной базе, а другая – к другой.

Рассмотрим алгоритмическую сложность задачи по отысканию областей обслуживания ресурсных баз ПГМ для минимизации затраты на доставку материала по формулам (1) на основе построенной информационной модели.

Понятно, что какие-либо рассмотрения о минимизации затрат на доставку материалов можно рассматривать на основе рациональных сетевых графиков проведения работ по зимнему содержанию, удовлетворяющих ограничениям (бинарным отношениям).

Как было показано выше, задача построения рациональных сетевых графиков проведения работ является NP – полной. Следовательно, задача по отысканию

областей обслуживания ресурсных баз ПГМ для минимизации затрат на доставку материалов по алгоритмической сложности NP – полна.

### 2.1.3 Автоматизации создания проектов организации дорожного движения на основе применения правил, соответствующих положениям нормативных документов

В настоящем разделе диссертационной работы рассмотрены следующие вопросы:

формализация представления данных по автомобильной дороге (и ее обустройству) и норм проектирования для обеспечения возможности автоматического применения;

построение системы автоматизированного проектирования технических средств организации дорожного движения.

Целью разработки проектов организации дорожного движения (ПОДД) является рациональная организация дорожного движения на автомобильных дорогах или отдельных ее участках для повышения пропускной способности и повышения безопасности движения транспортных средств и пешеходов.

Технические требования на ПОДД весьма сложны и определяются на настоящий момент следующими нормативными документами:

Федеральным законом от 10 декабря 1995 г. № 196-ФЗ «О безопасности дорожного движения»;

«Порядком разработки и утверждения проектов организации дорожного движения на автомобильных дорогах» (письмо МВД РФ от 02.08.2006 г. № 13/6-3853, ФДА от 07.08.2006 г. № 01-29/5313);

а также другими нормативными документами, определяющими правила применения и технические условия [4 – 8].

Процесс разработки ПОДД можно разделить на три этапа:

- Проведение полевых работ, включающих этап сбора данных передвижной видео лабораторией.
- Камеральная обработка результатов полевых работ.
- Построение ПОДД с учетом данных обработки.

Автоматизация предполагает разработку программных решений, позволяющих снизить объемы работ по выполнению пунктов 2 и 3.

Обработка данных видеопаспортизации представляет собой крайне ресурсозатратный процесс, результатом которого является план автодороги, полученный на основе видеоряда. От оператора в данном случае требуется скрупулезное изучение видеоряда на предмет наличия тех или иных объектов (например: ЛЭП, барьерные ограждения, направляющие устройства и т. д.) и позиционирование их на плане. Автоматизация данного процесса предполагает разработку программ, позволяющих в автоматическом режиме находить необходимые объекты на кадрах видеоряда, тем самым снижая нагрузку на оператора.

Процесс построения ПОДД требует от специалиста построения нового плана на основе текущего путем добавления новых и удаление существующих объектов а/д, таким образом, чтобы расположение объектов на новом отредактированном плане было выполнена, в соответствии с требованиями нормативных документов, и в соответствии с текущей дорожной обстановкой.

Имея базу данных (БД) объектов, представляющих автодорогу, можно представить процесс формирования ПОДД схемой, отображенной на рисунке 2.1.

Где состояние – это некоторый участок автодороги, а свойство – это функция, протяженная на этом участке и соответствующая некоторому свойству дороги (причем в качестве свойств может выступать наличие на участке трубы, дорожного знака, съезда и т. д.). Стоит отметить, что в БД функция представлена, как правило, набором аппроксимирующих точек.

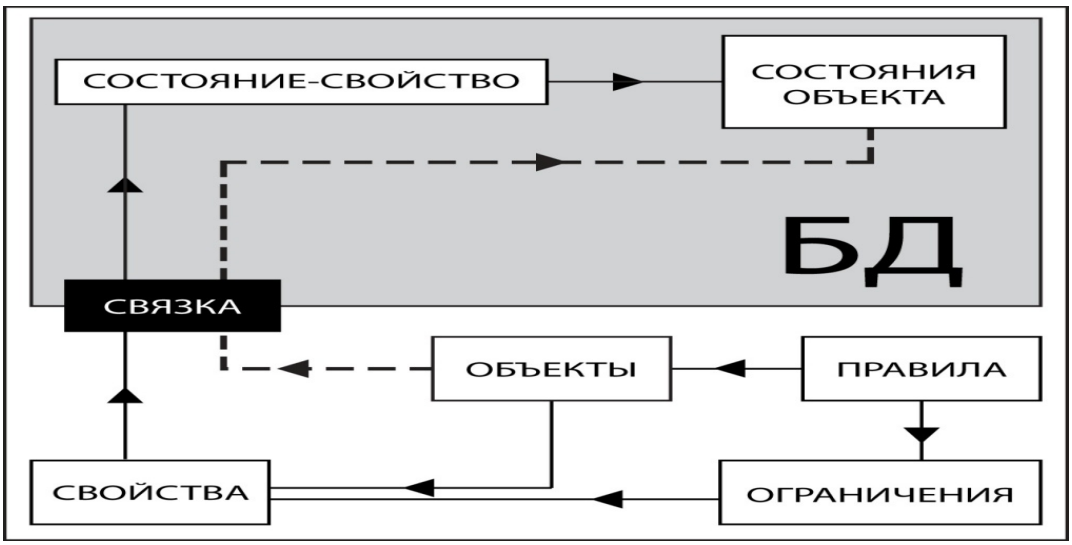



Рисунок 2.1. – Представление процесса формирования ПОДД для дорожных объектов

Таблица 2.2

Таблица связки

| id связки | имя таблицы в связи | имя поля содержащего значение в таблице связи |
|---|---|---|
| 12 | состояние-свойство | значение |

Таблица 2.3

Таблица правил

| id правила | id объекта | наименование | id списка ограничений |
|---|---|---|---|
| 1 | 2 | ГОСТ 1234 | 34 |

Таблица 2.4

Таблица ограничений

| id ограничения | id свойства | значение |
|---|---|---|
| 1 | 1 | асфальт |

В соответствии с таким подходом, было решено разработать надстройку над БД, представляющую собой набор фиксированных таблиц, и осуществляющую связь с основной БД, через таблицу «связка». Таблица «связка» имеет ключевое значение в схеме. За счет нее организуется «безболезненная» связь между основной БД и надстройкой. Представление таблицы связки – таблица 2.2, таблицы правил – таблица 2.3, таблица ограничений – таблица 2.4.

Так как схема разрабатывалась для общего случая автоматизация размещения подобъектов (т. е. в нашем случае дорожных ограждений, сигнальных устройств и т. д.) на протяженном объекте, то стоить разъяснить значения некоторые таблицы. Таблица «объекты» хранит все возможные протяженные объекты, на которых будут размещаться подобъекты. В нашем случае, в таблице будет только одна строка, соответствующая автомобильное дороге. Таблица «состояния объекта» хранит всевозможные участки автодороги (например, Иркутск-Листвянка, Братск-Усть-Илимск и т. д.). Таблица «состояние-свойство» хранит множество функций-свойств (высота насыпи, тип покрытия и т. д.) для каждого конкретного глобального участка (состояния) автодороги. Об особенностях представления правил будет сказано ниже.

Для представления а/д, заданной на промежутке $[a;b]$, используется система функций:

$$Road = \begin{Bmatrix} g_1(x), x \in [a;b] \\ g_2(x), x \in [a;b] \\ \cdots \\ g_n(x), x \in [a;b] \end{Bmatrix}$$

где $g_i$, $i = \overline{1,n}$ соответствует i-му свойству. Если свойство есть некоторая постоянно изменяющиеся величина (например «высота насыпи»), то функция имеет вид непрерывной кривой и при технической реализации представляет собой набор аппроксимирующих точек. Если же свойство представляет собой величину, которая может принимать значения только из строго заданного набора (например «тип покрытия»), то функция имеет ступенчатый вид (рисунок 2.2.).

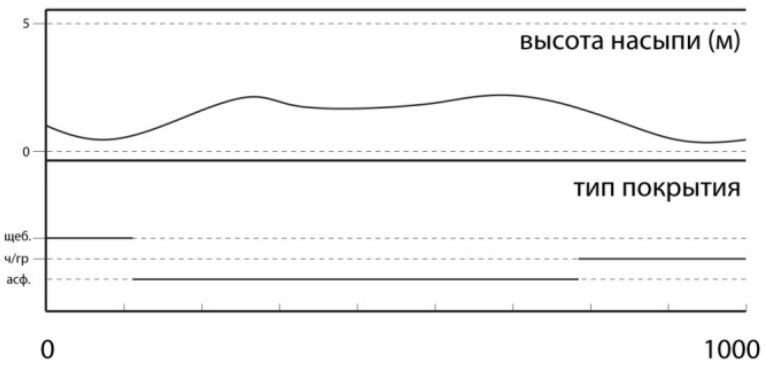



Рисунок 2.2. – Представление функций типа покрытия и высоты насыпи

В отличие от данных а/д, информацию о правилах в БД хранить не принято. Связанно это с тем что, при составлении ПОДД, вся ответственность о корректности расположения тех или иных объектов на дороге ложится на плечи проектировщика, который прекрасно владеет правилами, и в случае необходимости обращается к ГОСТам. С другой стороны, чтобы научить машину расставлять объекты на плане в соответствии с ГОСТами в автоматическом режиме, мы должны эти ГОСТы положить в БД.

Тут мы сталкиваемся с первой проблемой: с формальным представлением правила (т. е. как нам представить правило, чтобы его можно было хранить в БД). Так как каждое правило это есть некоторый набор конкретных значений свойств автодороги, то мы и будем представлять правило просто как набор значений:

$$P_j = (p_1, \ldots, p_n)$$

где каждая $p_i, \quad i = \overline{1,n}$ есть константа и соответствует i-му свойству. В более сложных случаях, например, когда значение свойства может принимать значения из некоторого заданного промежутка, можно заменить константы векторами, т. е.:

$$P_j = (\overline{p_1}, \ldots, \overline{p_n})$$

где $\overline{p_i} = (a_i, b_i), \quad i = \overline{1,n}$ и соответствует i-му свойству. Оба варианта легко реализуются в рамках СУБД.

Вторая проблема заключается в том, как ГОСТ привести к формальному виду. Рассмотрим решение данной проблемы на примере ГОСТ 52289 [4] пункта

8, правила применения дорожных ограждений и направляющих устройств. Так как постановка задачи требует от нас найти корректное расположение для некоторого объекта на автодороге, то можно пренебречь некоторыми правилами определяющими качественное представление объекта (напр.: кол-во направляющих устройств, удерживающая способность и т. д.). В качестве наглядного представления формализованного правила удобно использовать блок-схемы. На рисунке 2.3. приведена часть схемы для расстановки сигнальных устройств на АД. Каждый возможный «путь» в этой схеме будет соответствовать одной строке в таблице правил БД.

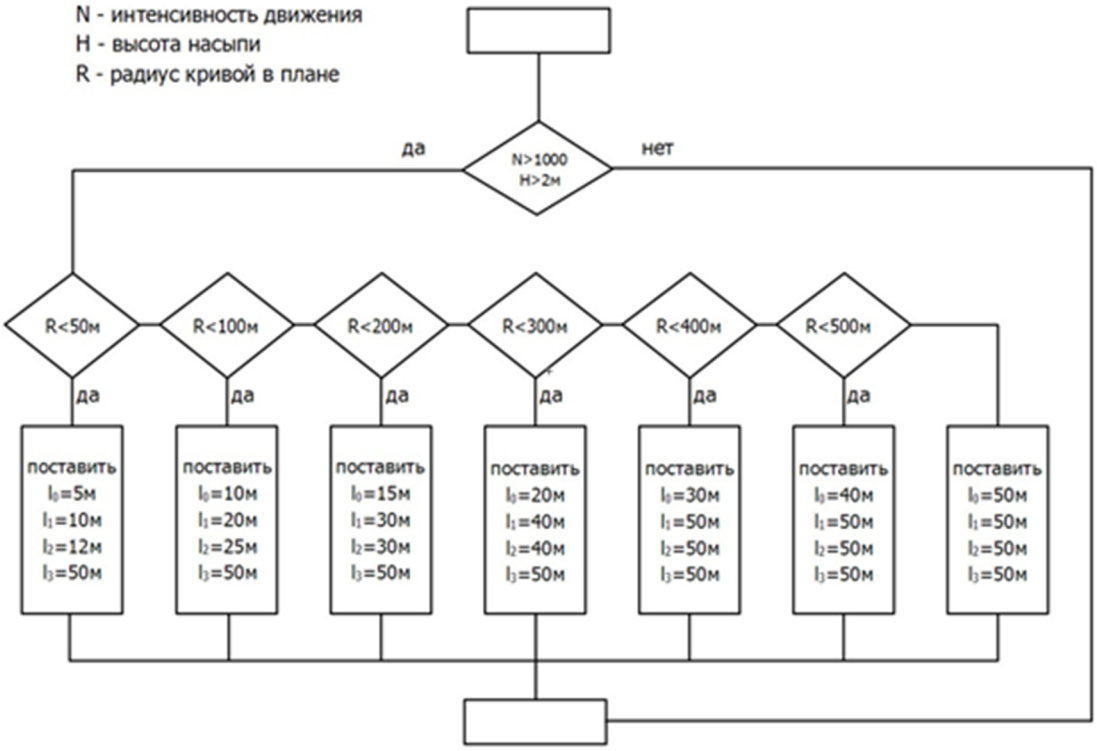



Рисунок 2.3. – Часть блок-схемы расстановки сигнальных устройств

Разработанная система представления, позволяет нам реализовать схему автоматизированного расставления объектов на плане АД. Для начала рассмотрим простой пример, который позволит нам лучше разъяснить принцип работы. Пусть автодорога определена в базе двумя функциями-свойствами A и B и задано два правила $P_1$, $P_2$:

$$P_1 = [(a_1, a_2), b_1]$$
$$P_2 = [(a_2, a_3), b_2]$$

На рисунке 2.4А представлены графики функций-свойств А (верхний) и В (нижний) на некотором участке дороги. Наша задача найти промежутки, удовлетворяющие заданным выше правилам. Мы разбиваем участок на под-участки точками пересечения функций A(x) и B(x) с прямыми $y = a_1$, $y = a_2$, $y = a_3$ и $y = b_1$, $y = b_2$ соответственно. В результате обнаруживается (рисунок 2.4Б), что промежуток $[r_1,r_2]$ удовлетворяет правилу $P_2$, а промежуток $[r_3,r_4]$ удовлетворяет правилу $P_1$. После того как промежутки найдены, остается подобрать объекты для которых выполнения любого из правил $P_1$ и $P_2$ достаточно для того, чтобы их расположение соответствовало правилам ГОСТ, и поставить их на плане АД.

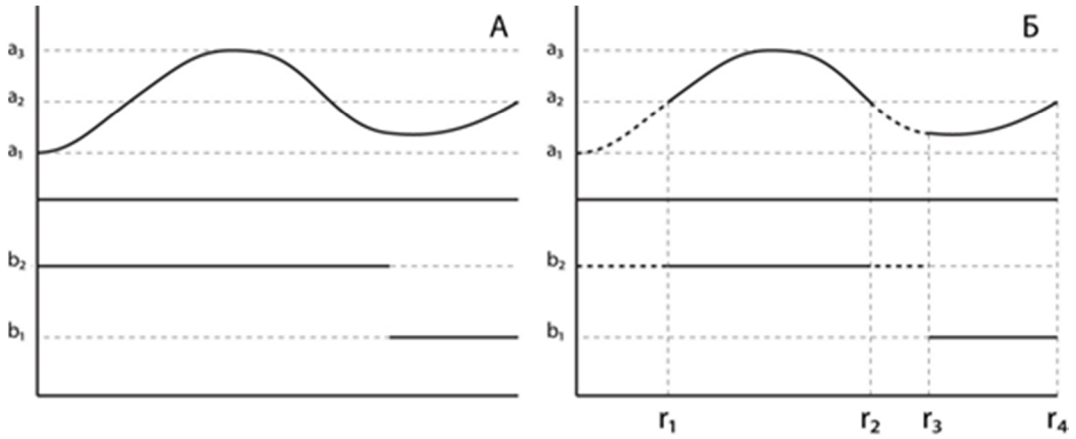



Рисунок 2.4. – Графики функций-свойств

Математически работа алгоритма описывается следующим образом. Дорога представлена системой функций *Obj* (далее объект).

$$Obj = \begin{Bmatrix} g_1(x), x \in [a;b] \\ g_2(x), x \in [a;b] \\ \cdots \\ g_n(x), x \in [a;b] \end{Bmatrix},$$

заданный в двумерном пространстве на промежутке $[a,b]$. Функции $g_i(x)$ будем называть свойствами объекта. Функции $g_i(x)$ являются непрерывными по X на промежутке $[a,b]$. Требуется найти промежутки из $[a,b]$, на которых объект удовлетворяет ограничениям.. Мы рассматриваем универсумы $A_i$ как области значений некоторых функций

$$A_i = E(f_i(x)) \quad i = \overline{1,n}$$

Где $f_i(x)$ – функции заданные на некотором промежутке Ограничения будем представлять следующим образом:

$$R_j = \begin{Bmatrix} f_1(x_j) \\ f_2(x_j) \\ \dots \\ f_n(x_j) \end{Bmatrix} \quad j = \overline{1,m}$$

**Этап 1.** Сведение объекта $Obj$ к новому объекту $Obj_2$ заданному ступенчатыми функциями. Необходимо обеспечить возможность эффективной работы с объектом. Для этого имеет смысл упростить представление объекта. Т. е. построить переход от $Obj$ к новому, упрощенному, объекту $Obj_2$.

$$Obj_2 = \begin{Bmatrix} h_1(x), x \in [a;b] \\ h_2(x), x \in [a;b] \\ \dots \\ h_n(x), x \in [a;b] \end{Bmatrix},$$

где $h_i = f_i(x_{j_1})), \quad x \in \{x \mid f_i(x_{j_1}) \le g_i(x) < f(x_{j_2})\}$

**Оценка**: сложность осуществления перехода порядка $O(m)$.

**Этап 2.** Разбиение объекта $Obj_2$ на множество подобъектов $\{obj_k\}$. Данный шаг позволяет нам перейти от объекта заданного системой функций к множеству подобъектов, каждый из которых определен системой констант. Такое разбиение позволяет нам реализовать методы проверки ограничений, который подробно были рассмотрены в работах [9].

Пусть $s_i$ – кол-во точек разрыва у функции $h_i(x)$ на $[a;b]$. $S = \sum_i s_i$ – это общее кол-во точек разрыва на $[a;b]$ у всех функций $h_i(x)$ вместе взятых. Пусть $\{x_1, \dots x_S\}$ – упорядоченное по возрастанию множество точек разрыва. Тогда наше множество подобъектов будет задано следующим образом:

$$\left\{ obj_i = \begin{Bmatrix} h_1^+(x_i), & x \in [x_i, x_{i+1}) \\ h_2^+(x_i), & x \in [x_i, x_{i+1}) \\ \dots & \\ h_n^+(x_i), & x \in [x_i, x_{i1+}) \end{Bmatrix} \right\}$$

при $i < S-1$

$$\left\{ obj_i = \begin{Bmatrix} h_1^+(x_i), & x \in [x_i, b] \\ h_2^+(x_i), & x \in [x_i, b] \\ \dots & \\ h_n^+(x_i), & x \in [x_i, b] \end{Bmatrix} \right\}$$

при $i = S-1$

Отметим что $x_1 = a, x_S = b$.

**Оценка**: сложность построения нового множества объектов порядка $O(S)$.

Оценки позволяют сделать вывод о возможности эффективной реализации предложенного метода.

## 2.2 Реализация программного комплекса поддержки управления дорожной сетью

### 2.2.1 Организация модульной структуры программного комплекса мониторинга и поддержки управления сети автодорог

Основная цель развития информационных технологий (ИТ) в дорожной отрасли – формирование и обеспечение функционирования системы информационного обеспечения органов управления, организаций и предприятий дорожного хозяйства, пользователей автомобильных дорог.

Одним из первых программных продуктов, функционирующих в отрасли, является разработанный в РосдорНИИ автоматизированный банк дорожных данных (АБДД), в котором хранится информация о дорогах федерального подчинения. На региональном уровне для хранения оперативной информации создавались собственные банки данных с использованием различного программного обеспечения [12]. В 2006 г. в Федеральном дорожном агентстве завершены работы по созданию корпоративной информационной системы управлении (КИСУ) Феде-

рального дорожного агентства (Росавтодора). С 2000 г. ведутся работы по созданию и развитию современных систем связи и автоматизированных систем оперативного управления (АСОУ) в органах управления федеральными автомобильными дорогами с целью решения следующих задач:

— оперативный непрерывный мониторинг транспортно-эксплуатационного состояния автомобильных дорог и соблюдение технологических процессов их содержания;

— своевременное предоставление информации об объемах и качестве выполняемых дорожных работ;

— мониторинг ситуации на важнейших участках автомобильных дорог путем внедрения систем видеоконтроля и автоматизированного учета интенсивности движения;

— обеспечение информирования участников дорожного движения об обстановке на дорогах.

Анализируя существующее программное обеспечение, можно выделить следующие недостатки:

Прикладное программное обеспечение слабо интегрировано;

Практически отсутствуют программы, обеспечивающие интеграцию производственных и сопутствующих задач, например таких, как автоматизация бухгалтерского учета, ведения документооборота, решения задач планирования и т. п.

В настоящее время определены основные направления развития для информационных систем автомобильно-дорожной отрасли. Разрабатываемый программный продукт должен предоставлять всем сотрудникам доступ к территориально распределенным базам данных, содержащих информацию о строительстве и эксплуатации, мониторинга состояния автомобильных дорог, финансовых ресурсах, сметных нормативов, наличии рабочей силы, средств механизации и материалов.

Необходимость создания гибкого настраиваемого программного комплекса продиктована современным интенсивным развитием информационных технологий. Поэтому следует проектировать программный комплекс так, чтобы при

необходимости можно было с минимальными трудозатратами отделить какую-то часть системы. Для этого необходимо, чтоб каждый из модулей программы работал со своей независимой частью базы данных, при этом во избежание ошибок, преобразование данных должно происходить легко и по возможности незаметно для пользователя.

Помимо всего прочего, могут использоваться различные платформы СУБД, поэтому следует разрабатывать систему так, чтобы клиентская часть наименее всего зависела от серверной. При этом не следует слишком перегружать серверную часть, так как при переходе на другую СУБД некоторые решения, используемые в хранимых процедурах для MS SQL Server, абсолютно не подходят для таких систем как Oracle или PostgreSQL. Конечно, трудно отказаться от мощных возможностей, предлагаемых современными СУБД, но ограничение использования какой-то конкретной системы управления базами данных ограничивает круг потенциальных пользователей. Кроме того, конечно, объем данных хоть и кажется достаточно внушительным, все же считается небольшим даже для open source СУБД.

Что же касается клиентской части, ее также не следует чересчур перегружать. Ведь любое изменение в логике программы должно будет в этом случае повлечь за собой изменения на всех рабочих местах. На этапе внедрения это может сказаться катастрофически. Однако следует сказать, что расчетные модули программы имеют дело только непосредственно с локальными данными. Так, чтобы пользователь при необходимости мог бы с легкостью отказаться от части расчетов, или вести расчет независимо от базы данных.

В составе прикладных систем выделяются следующие блоки:

— информационно-аналитический;

— финансы и имущество;

— производство;

— нормативно-справочная информация.

Эффективное применение ресурсного метода расчета стоимости работ возможно только при наличии компьютерных баз данных по всем элементам автомо-

бильных дорог и обустройства. Поэтому необходимо обеспечить максимальную полноту данных для проведения расчетов.

Рассмотрение ведения работы по организации управления дорожной сети показало, что здесь нет уже ярко выраженной этапности формирования информационной наполняемости базы данных. Это обусловлено, прежде всего, огромными потоками информации, проходящими через широко разветвленную управленческую структуру. Проект сам по себе, представляет собой хранилище данных, на основании которых возможно получать некоторые решения.

Рисунок 2.5. – Информационная основа системы мониторинга сети автомобильных дорог

Система имеет возможность получать информацию по состоянию покрытия, геометрическим характеристикам, наличию труб и мостов, ограждений и другим данным по дороге из электронных инвентаризационных паспортов, видеоизмертиельной системы видеопаспортизации дорог, информационной системы состояния искусственных сооружений (ИССО), а также из удаленных рабочих мест самой системы мониторинга, установленных, как правило, у следящих за

участками дорог инспекторов. На рисунке 2.5. представлены информационные потоки, наполняющие БД системы мониторинга.

Вся информация интегрируется в единую базу данных и в дальнейшем доступна для получения всевозможных отчетов по каждому из информационных потоков в отдельности. Нужно сказать, что каждый из перечисленных источников практически всегда содержит в себе неполную картину состояния дороги. К примеру, в инвентаризационном паспорте нет дефектной ведомости, а в видеопаспортизации ничего не говорится об эксплуатационной категории. Такое разнообразие взаимодополняющей информации, необходимо каким-то образом реплицировать, и, в данном случае, единственным самым простым и верным решением, кажется, указание каждой из характеристик из какого источника данных она была получена.

При этом нельзя забывать, что система должна на основании этих характеристик делать какие-то расчеты, при этом всегда вставал бы вопрос по какому источнику данных производить просчет. Это было бы крайне затруднительно и, возможно, привело бы к противоречивым результатам при использовании разных источников данных. Поэтому было принято решение, что пользователь системы должен сформировать «рабочий» источник данных, взяв по своему желанию различные характеристики из разных источников данных.

На основе «рабочих» данных формируется уже дальнейшая трансформация данных, способствующая принятию управленческих решений, а также проводящую обычную калькуляцию.

Понятно, что невозможно готовить отчетную информацию (в особенности финансовую), основываясь только лишь на автоматическом расчете. Поэтому должна иметься возможность корректировки любого из полученных результатов.

Большое количество характеристик, составляющих информацию по дороге и влияющих на производимые расчеты, заставляет разработчика создавать инструментарий для обеспечения гибкости расчетных модулей.

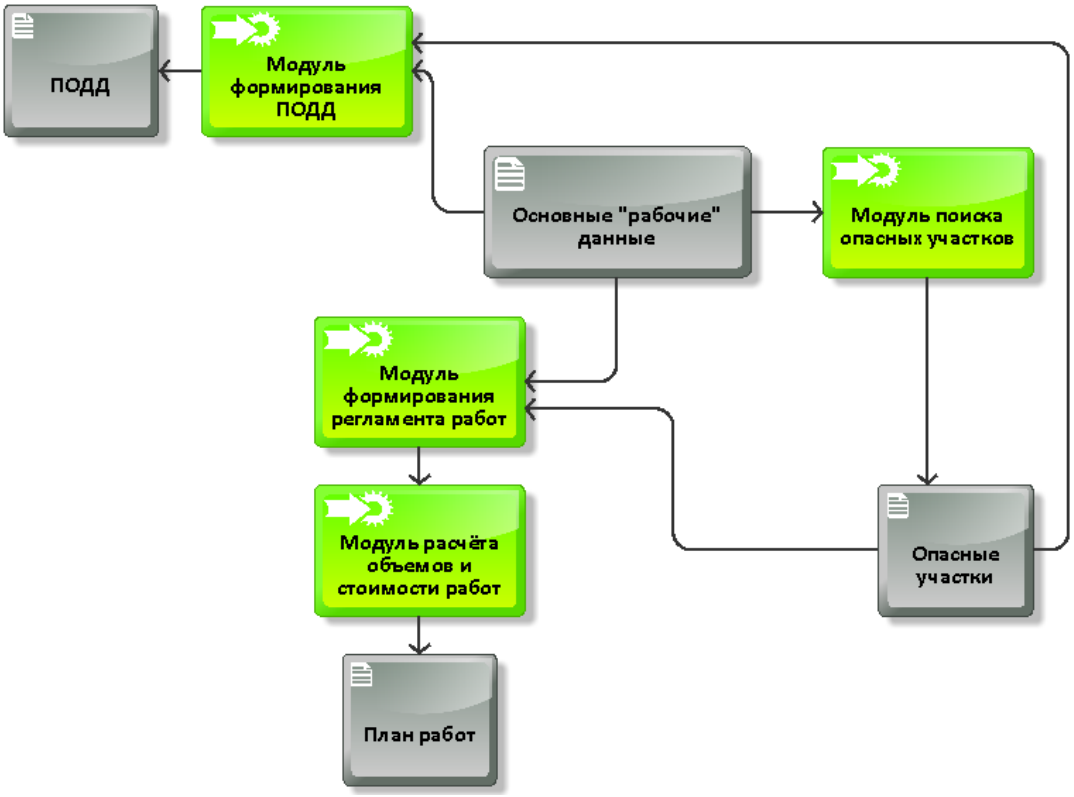



Рисунок 2.6. – Трансформация данных в системе мониторинга

Так, для модуля расчета объемов было создано средство, позволяющее задавать способы расчета объемов работ.

Система мониторинга и поддержки управления дорожной сети является достаточно крупным проектом, обладающим широкими функциональными возможностями. Основным принципом проектирования данной системы было понятие модульности. Каждый из модулей должен был наделять определенным функционалом рабочее место того или иного специалиста. Поэтому была принята на вооружения технология плагинов (plugin).

Система состоит из основного модуля, который отвечает за загрузку всех плагинов и содержит в себе некоторые общенеобходимые функции. Этот модуль также обеспечивает связь между различными плагинами, управление главным меню программы, в зависимости от наличия плагинов и содержит такие программы как: обработка исключений, программу, отвечающую за работу с базой данных, программу для работы с журналом действий (LOG).

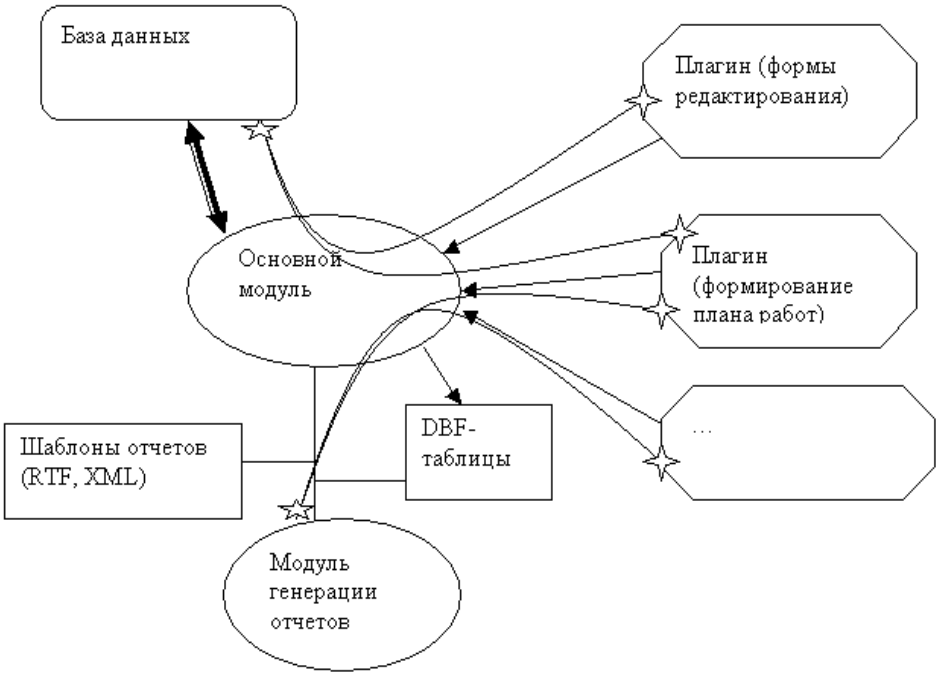



Рисунок 2.7. – Принцип работы системы мониторинга и поддержки управления сетью автомобильных дорог

Особняком от остальных плагинов стоит модуль генерации отчетов, позволяющий по заданному шаблону в формате RTF или XML формировать отчеты в соответственном формате. На вход ему подается имя шаблона, в котором описаны какие таблицы (в формате FoxPro) и каким образом нужно использовать, чтобы сформировать тот или иной отчет. Этот модуль не является по сути плагином, так как не требует никакой служебной информации от основного модуля. Он может быть использован в абсолютно других приложениях. Так, например, он активно используется в информационной технологии организации учебного процесса.

В остальном, система целиком и полностью состоит из плагинов, добавляя или убирая которые из программы можно менять функциональные возможности программы. Например, созданы такие плагины как: плагин, обеспечивающий автоматическое создание формы ввода данных по конфигурационному файлу; плагин, отвечающий за копирование данных из одной базы в другую; плагин, рассчитывающий объемы работ и многие другие. На сегодняшний момент система содержит около двух десятков плагинов, но в будущем вполне может продолжать

расширяться за счет них. Реализация каждого из них, возможно, требует детального рассмотрения, однако это выходит за рамки данной работы.

В следующем параграфе перейдем к структуре базы данных и решениях, предложенных для их хранения и получения.

### 2.2.2 Организация хранилища данных и основные технические решения, используемые в проекте базы данных

Система мониторинга и поддержки управления дорожной сетью представляет собой достаточно большой программный комплекс, который, прежде всего, должен обеспечивать максимально полной информацией о состоянии дороги на основании полученной из какого-либо источника данных информации. Так как различные источники данных дают различные характеристики дорожных объектов, встает проблема представления данных в базе в наиболее общей, легко масштабируемой форме.

Традиционно информация об объектах, представленная в виде таблицы выглядит следующим образом (таблица 2.5.): запись в целом содержит информацию об объекте, а поля записи – атрибутивные характеристики объекта.

Таблица 2.5

Хранение атрибутов объектов

| ID | Attr1 | Attr2 | Attr3 |
|---|---|---|---|
| 0101 | 12 | 01.01.2006 | Комментарий к строке 1 |
| 0102 | 15 | 12.12.2006 | Комментарий к строке 2 |

Таблица 2.6

Хранение атрибутов по значениям

| HOI | HDC | VAL | | |
|---|---|---|---|---|
| 0101 | 0101001 | 12 | | |
| 0101 | 0101002 | | 01.01.2006 | |
| 0101 | 0101003 | | | Комментарий к строке 1 |
| 0102 | 0101001 | 15 | | |

Однако подобное представление не может быть использовано по причине возможного увеличения количества атрибутов у объекта. Вполне естественно, что

различные объекты могут иметь различное количество атрибутов. Выходом из сложившейся ситуации является подход, когда каждой атрибутивной характеристике соответствует только одна запись (таблица 2.6).

**HOI** – Hierarchy Object Identification – Иерархический идентификатор объекта.

**HDC** – Hierarchy Data Classification – Иерархический классификатор данных.

**VAL** – Value – Значение.

Единственной технической сложностью реализации такого представления данных, является хранение значения атрибута, поскольку разные атрибуты могут быть представлены различными типами данных. Можно предложить несколько возможных вариантов решения проблемы. Например, использовать в качестве типа данных поля [VAL] тип BINARY, или создать в таблице поля, соответствующие всем возможным используемым типам данных (фактически расщепление поля [VAL] на [VAL_INTEGER], [VAL_DOUBLE], [VAL_STRING], [VAL_DATA], [VAL_BINARY] и т. д.). Корректность информации, помещаемой в базу данных, может в этом случае обеспечиваться программным обеспечением. Существует возможность простого преобразования таблицы подобной структуры в «традиционный» вариант. Для этого достаточно в названии или комментарии к полям «классической» таблицы указывать иерархический классификатор данных (HDC).

Сама возможность таких преобразований данных чрезвычайно важна, поскольку позволяет использовать стандартные, традиционные методы построения форм и отчетов, базирующихся именно на «классическом» представлении данных.

Однако такой подход не совсем отвечает требованиям системы. Дело в том, что полученные данные из, скажем, видеопаспортизации представляют собой просто громаднейший массив данных по отдельно взятой дороге, а чаще требуется хранение только лишь интегральных характеристик. В этом случае приходится приспосабливать данные под жесткую структуру, конвертируя их специальными программами, и храня множественные данные (такие как

координаты точек краев дороги) в поле типа BINARY для более быстрого доступа к данным. В целом, надо сказать, подход вполне удовлетворяет потребностям системы, единственное расширение общей структуры хранения информации это добавление поля, идентифицирующего источник данных. Как уже отмечалось выше, таких источников данных достаточно много.

На этапе, когда система была уже введена в эксплуатацию, выяснилось, что требуется не только простое хранение данных, но и отслеживание их изменений. Например, для предоставления годового отчета по исполненным работам нужна информация по изменению дорожного покрытия. К сожалению, разработчики не были готовы к подобному требованию. Конечно, на сегодняшний день разработана масса вариантов хранения данных. Приведем в качестве примера структуру данных на основе хранения ревизий объектов (таблицы 2.7, 2.8).

Таблица 2.7

Хранение ревизий объектов

| OBJID | REVISION | FROMDATE | TODATE |
|---|---|---|---|
| 0101 | 1 | 01.01.2006 | Null |
| 0102 | 1 | 01.01.2006 | 01.11.2006 |
| 0102 | 2 | 01.01.2006 | Null |

Таблица 2.8

Хранение ревизий свойств объектов

| OBJID | PROPID | FROMREVISION | TOREVISION | VAL |
|---|---|---|---|---|
| 0101 | 0101001 | 1 | Null | 10 |
| 0102 | 0101001 | 1 | 2 | 11 |
| 0102 | 0101001 | 2 | Null | 12 |

Данная структура позволяет восстановить состояние базы данных на любой момент времени, однако такое решение подразумевает глубокий рефакторинг базы данных и, как оказалось, переписывания программного кода клиента, что не всегда является приемлемым. Кроме того, одним из недостатков данного подхода является то, что для выборки текущей ревизии будет затрачиваться ресурс

времени, а при больших объемах данных этот ресурс будет значительно увеличиваться. Поэтому был предложен другой подход.

Смысл подхода заключается в клонировании таблиц, по которым нужно проводить отслеживание изменений. Чаще всего, это были таблицы, содержащие интегральные характеристики по дороге (ширина проезжей части, покрытие, обочины). В таблицу клон добавлялись поля, хранящие информацию о том, когда, по какой причине и какие изменения (запись добавлена, запись изменена, запись удалена) были внесены в основную таблицу.

Рассмотрим приведенную схему на примере таблицы, хранящей общую информацию по участкам дорог. В таблице 2.9 NumRoad – код дороги, NumDataSource – код источника данных, NumPartType – ссылка на классификатор типов участков дорог (ширина проезжей части, тип покрытия, техническая категория и многие другие), StartPos и EndPos – координаты начала и конца участка, Reference – ссылка на объект участка дороги и Value – его значение.

Соответственно, в таблицу клон необходимо добавить поля NumReason – причина изменения, NumOperation – операция (добавление, удаление, изменение) и ModifyDate – дата изменения.

Таблица 2.9

Хранение информации изменений по участкам дорог

| NumPartType | NumRoad | NumDataSource | StartPos | EndPos | Reference | Value | NumReason | NumOperation | ModifyDate |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 910 | 0 | 0 | 90 | 456 | | 1 | 3 | 01.11.2006 |

Затем следовало написать триггеры, которые бы добавляли соответствующую запись в таблицу клон.

И немного изменить модуль работы с базой данных, таким образом, чтобы можно было передавать причину проделываемых изменений. Так как этот модуль отвечал в целом за работу всей системы с базой данных, то практически, можно сказать, отслеживание изменений, производимых с базой данных, при таком подходе применим без изменения большого количества программного кода.

Конечно, потребовалось написание модуля, который выводил бы отчеты по проделанным изменениям и позволял бы отменять их или изменять причину. В силу того, что структура данных достаточно сложна, серверная часть этого модуля достаточно громоздка и требует глубоких знаний языка запросов MS SQLServer T-SQL. Однако, предложенный подход позволяет решить вопрос хранения изменения данных без потери в производительности запросов к текущим данным и является наиболее удобным, если глубокий рефакторинг базы данных затруднен или невозможен.

# 3 Применение логико-эвристического подхода для создания программного комплекса, обеспечивающего решение задач организации учебного процесса

## 3.1 Математические модели задач организации учебного процесса

При организации учебного процесса приходится решать ряд задач сетевого планирования, причем без решения каждой из них нельзя говорить о замкнутости технологического цикла. Действительно, формирование учебного плана на основе государственного образовательного стандарта, формирование нагрузки преподавателей, построение рабочего графика учебного процесса на текущий учебный год, распределение аудиторного фонда по занятиям и проектирование расписания занятий представляют себе полную схему технологии организации учебного процесса.

Пусть $T = \{t_1, t_2, \ldots, t_{kt}\}$ – множество семестров, $S = \{s_1, s_2, \ldots, s_{ks}\}$ – множество специальностей, $D = \{d_1, d_2, \ldots, d_{kd}\}$ – множество учебных предметов (дисциплин), N – множество натуральных чисел.

Рассмотрим математическую постановку задачи формирования учебного плана по государственному образовательному стандарту специальностей. Отображение $STAND: D \times S \to N$ – государственный образовательный стандарт специальностей учебного заведения,

$$\boldsymbol{U: D \times S \times T \to N} \qquad \textbf{(3.1)}$$

будем называть учебным планом.

Норматив количества часов в неделю для учащихся

$$C1 \le \sum_{i=1}^{kd} U(d_i, s_j, t_k) \le C2 \qquad \forall j = 1 \ldots ks, k = 1 \ldots kt \qquad \textbf{(3.2)}$$

Константы C1 и C2 определяются нормативными документами учебного заведения и обозначают максимальное и минимальное количество учебных часов в однонедельном цикле (обычно C1 = 24, C2 = 36).

Норматив возможного отклонения от государственного стандарта специальностей.

$$DMIN \cdot \sum_{i=1}^{kd} STAND\,(d_i, s_j) \le \sum_{i=1}^{kd}\sum_{k=1}^{kt} U(d_i, s_j, t_k) \le DMAX \cdot \sum_{i=1}^{kd} STAND\,(d_i, s_j) \qquad \forall j = 1...ds \quad \textbf{, (3.3)}$$

DMIN и DMAX – константы.

Пусть теперь построено отображение U: D × S × T → N. Здесь возникает задача построения отображения NAGR: U × P → N, называемого нагрузкой преподавателей. Для построения отображения нам потребуется следующие множества

OMIN= {$Omin_1$, … , $Omin_l$} – минимально допустимая нагрузка для преподавателей; OMAX= {$Omax_1$, … , $Omax_l$} – максимально допустимая нагрузка для преподавателей.

Построение множеств OMIN и OMAX, довольно сложная задача, зависящая от многих факторов, которые не поддаются формализации, поэтому, например, в высших учебных заведениях данная задача решается на уровне заведующих кафедрами.

Введем следующее ограничение на построение отображения NAGR:

$$O\min_k \le \sum_i \sum_j NAGR\,(U(d_i, s_j, t_h), p_k) \le O\max_k, \qquad \forall k, h, \qquad d_i \in D, s_i \in S, t_h \in T.$$

После построения отображений U и NAGR, то можно рассмотреть задачу построения графика учебного процесса, где ограничения определяются возможностями аудиторного фонда и максимально допустимой нагрузкой для преподавателей. Данная задача также имеет комбинаторный характер и может быть решена теми же методами, что и проектирование расписания.

Основными объектами задачи планирования расписания (по дневной форме обучения) являются множества:

Преподавателей – P = {$p_1$, $p_2$, …, $p_{kp}$},

Контингентов учащихся C = {$c_1$, $c_2$, …, $c_{kc}$},

Предметов D = {$d_1$, $d_2$, … $d_{kd}$},

Аудиторий A = {$a_1$, $a_2$, …, $a_{ka}$},

Пар (двухнедельный цикл) M = {$m_1$, $m_2$, …, $m_{km}$}.

Совокупность планируемых занятий ZZ ⊆ P × C × D. Расписание занятий Sh – инъективное отображение ZZ → R, где R: A × M – ресурс. Расписание занятий

Sh должно удовлетворять ограничениям, как по типам объектов (смены для контингентов учащихся, сбалансированность учебной недели и др.), так и по ограничениям для конкретных объектов (для преподавателей: освобожденные дни или пары, максимальное количество занятий в день, причем количество лекций из них и др.).

Для заочной формы обучения требуются существенные уточнения по следующим двум позициям.

1. Множество семестров $T = \{t_1, t_2, \ldots, t_{kt}\}$ должно быть дополнено множеством сессий $TT = \{tt_1, tt_2, \ldots, tt_{ktt}\}$, проводимых для соответствующих потоков учащихся $SS = \{ss_1, ss_2, \ldots, ss_{kss}\}$ (отметим, что сессии могут быть установочными и экзаменационными, что для наших целей, вообще говоря, не имеет значения), требующих определения своих сроков проведения в период соответствующих семестров. Отметим, что в дальнейшем построение рабочего графика учебного процесса будет сводиться к определению этих сроков проведения, удовлетворяющих определенным ограничениям, т. е. фактическому построению отображения

$$WG: TT \to N \times N, \qquad (3.4)$$

где WG(tt) = (n1, n2) означает: n1 (n2) первый (соответственно, последний день) сессии $tt \in TT$.

Соответствие между сессиями $TT = \{tt_1, tt_2, \ldots, tt_{ktt}\}$ и потоками учащихся $SS = \{ss_1, ss_2, \ldots, ss_{kss}\}$ определяет отображение

$$WT: TT \to SS. \qquad (3.5)$$

2. Множество специальностей $S = \{s_1, s_2, \ldots, s_{ks}\}$ должно быть дополнено множеством потоков учащихся $SS = \{ss_1, ss_2, \ldots, ss_{kss}\}$, причем каждый поток учащихся имеет определенную специальность и количество групп, т. е. заданы отображения

$$WS1 : SS \to S; \; WS2 : SS \to N. \qquad (3.6)$$

Определим одно из основных ограничений, которому должен удовлетворять рабочий график учебного процесса (3.4). Положим для каждого календарного дня

i семестра множество потоков учащихся DaySS(i) = {ss | где ss ∈ SS такие, что календарный день i включен в период WG (tt), причем WT (tt) = ss}.

Количество групп (определяемое отображением WS2 : SS -> N) в множестве потоков учащихся DaySS(i) для каждого календарного дня i должно быть в некотором заранее определенном интервале (должна быть обеспечена равномерность количества учащихся по заочной форме обучения на весь учебный год).

Другие важные ограничения для рабочего графика учебного процесса (3.4) вводятся на основании рабочего учебного плана (3.1). Основные требования ограничений связаны с равномерностью загрузки специализированного аудиторного фонда и преподавателей. Объемы такой загрузки (количество пар) определяются рабочим учебным планом (3.1).

Построение рабочего графика учебного процесса (3.3), является типичной задачей сетевого планирования мероприятий, удовлетворяющих ограничениям (бинарным отношениям), многие из которых являются отношениями частичного порядка (достаточно вспомнить установочные и экзаменационные сессии, тематические связи, упорядочивающие время проведения дисциплин и др.).

Для определения соответствия между занятиями и аудиториями определим множество специализаций занятий $Sp = \{sp_1, sp_2, \ldots, sp_{ksp}\}$, где $sp_1$ – соответствует занятиям, проводимым в аудиториях без специального оборудования.

Тогда отображения

$$\boldsymbol{ZZsp: ZZ \rightarrow Sp,\ Asp: A \rightarrow Sp,} \qquad \boldsymbol{(3.7)}$$

определяют соответствие по специализации между занятиями и аудиториями. В частности, если Sh (zz) = (a, m), то ZZsp (zz) = Asp (a).

Положим подмножества $ZZ_i$ (соответственно, $A_i$) состоящими из элементов множеств ZZ (соответственно, A) специализации $\mathbf{sp_i}$, т. е.

$$\boldsymbol{ZZ_i = \{zz \mid zz \in ZZ \text{ и } ZZsp\ (zz) = sp_i\}}$$

(соответственно, $A_i = \{a \mid a \in A$ и Asp (a) = $\mathbf{sp_i}\}$).

Задача распределения аудиторного фонда по занятиям перед проектированием расписания состоит в формировании по множествам занятий $ZZ_i$ подмножеств аудиторий $A_i$ достаточных для проектирования расписания.

Отдельным важным вопросом является обеспечение занятий аудиториями не только необходимой специализации (по оборудованию), но и достаточной вместимости. Данный вопрос решается определением численности учащихся для занятий и вычисления по этим данным необходимого числа аудиторий соответствующей вместимости.

## 3.2 Этапы организации учебного процесса

Организация учебного процесса – комплексная проблема, включающая в себя как чисто технические аспекты (ведение баз данных, документооборот, расчет нагрузки и т. д.), так и решение наукоемких задач (проектирование расписания, создание графика учебного процесса, распределение аудиторного фонда и др.). Ведение крупных информационных потоков, связанных с организацией учебного процесса, должно сопровождаться постоянным контролем и не должно отнимать слишком большой временной ресурс у пользователей системы.

Бесспорно, окончательным результатом технологии служит составленное расписание. Подготовка данных для составления расписания очень сложный и трудоемкий процесс. Действительно, непосредственный ввод в базу данных нескольких тысяч занятий сравним по сложности с составлением самого расписания, при этом велика вероятность допустить ошибку, отследить которую очень непростая задача. Кроме того, понятно, что малейшие изменения в структуре контингента или в учебном плане повлекут за собой довольно объемные изменения в данных к расписанию. Именно поэтому следует проходить большинство этапов организации учебного процесса для получения действительно корректных данных. При всем этом, результат каждого из промежуточных этапов важен и также является неотъемлемой частью работы учебного отдела.

Последовательное получение информации при переходе к решению от одной задачи к другой выгодно отличает организацию учебного процесса. Для построе-

ния полноценного функционала информационной системы необходимо знать все аспекты предметной области и иметь возможность выделить этапы трансформации данных.

Организация учебного процесса включает в себя следующие основные последовательно исполняемые этапы.

Формирование учебного плана на основе государственного образовательного стандарта. Государственный образовательный стандарт содержит перечень предметов и количество часов по ним. В учебном же плане эти часы по каждому предмету должны быть разнесены по семестрам с соблюдением возможной нормы отклонения от государственного образовательного стандарта и нормы учебных часов в неделю учащихся. Также там должен учитываться и частичный порядок следования дисциплин, так как иногда проведение одной из дисциплин должно предварять проведение другой. В учебном плане указывается список практик – множество названий практик с указанием семестра проведения и количества недель. Зная общее количество недель теоретического обучения, объем необходимого времени на сессию, а также список практик по каждому семестру можно построить типовой график учебного процесса для каждого плана. В типовом графике указываются сроки проведения учебной нагрузки так, будто учебный год начинается с понедельника 1 сентября и разбит на циклы с минимальной длиной в половину недели. Совокупность вышеперечисленных информационных потоков (перечень дисциплин с указанием разбиения часов по семестрам, план практик, типовой график) будем называть учебным планом. Кроме того, необходимо сформировать и откорректировать дополнительные параметры учебных планов, которые необходимы системе. Обычно информацию по учебным планам предоставляют кафедры.

Здесь стоит отметить, что необходимость ввода всей этой информации обуславливается не только и не столько требованиями системы, а, в первую очередь, нужна для ведения отчетности по учебным планам специальностей. Хотя, несомненно, работа системы организации учебного процесса основывается на информации, получаемой из учебных планов.

В плане контингента указываются учебные планы, по которым производится обучение каждой единицы контингента. Под контингентом в данном случае понимается потоки, группы и подгруппы на основе административного деления.

Обычно в учебном заведении не всегда ведется работа по всем учебным планам каждый год. Однако, необходимо сохранять информацию о каждом учебном плане, по которому когда-либо проходило обучение. Поэтому удаление учебного плана происходит очень редко и вводится понятие актуального учебного плана. Актуальными учебными планами будем называть те учебные планы, по которым учится какой-либо контингент в текущем учебном году. План контингента и учебные планы создают фундамент получения основных информационных потоков для подготовки данных к расписанию.

Рисунок 3.1. – Получение информации на текущий учебный год

Выбрав все дисциплины из актуальных учебных планов, и распределив часы каждой из дисциплин по семестрам и видам работ (лекции, семинары, лаборатор-

ные, и т. д.), получаем рабочий учебный план. Рабочий план практик получается аналогичным образом, только здесь уже выбираются практики из актуальных учебных планов.

Одним из важнейших и непростых моментов в технологии является объединение и переформирование контингента. Автоматически можно сформировать предлагаемый список занятий для объединения на основе рабочего учебного плана и плана контингента. Не следует забывать, что построенные объединения очень сильно отразятся в дальнейшем при формировании рабочего графика учебного процесса, расчете нагрузки и проектировании расписания.

Построение рабочего графика учебного процесса, где указываются виды работ (теоретическое обучение, сессия, практики, каникулы) вместе с реальными сроками их проведения, основывается на данных из актуальных учебных планов (типовые графики) и дате начала учебного года. Следует отметить, что при формировании рабочего графика учебного процесса, особенно для заочной формы обучения, если имеется достаточная свобода выбора сроков проведения теоретического обучения и при наличии информации, ограничивающей выделенный аудиторный фонд, максимально возможную нагрузку преподавателей и общее количество студентов, приходится решать непростую задачу. Эта задача имеет комбинаторный характер.

Рисунок 3.2. – Получение информации по рассчитанной нагрузке

Расчет нагрузки ведется на основе норм времени, где определяются общие правила подсчета часов (не путать с аудиторными), а также плана контингента (с учетом объединения и переформирования), рабочего учебного плана и рабочего плана практик. После расчета нагрузки получится таблица, где в каждой строке будут указаны дисциплина, вид работы, контингент и рассчитанные часы.

Теперь уже можно переходить к распределению нагрузки между преподавателями, то есть сопоставлению часов нагрузки конкретным преподавателям. Таким образом, данные к расписанию получаются на основе информации о распределении нагрузки между преподавателями и рабочего графика учебного процесса. Здесь уже получается таблица, где каждой из строк соответствует занятие: дисциплина, вид работ, контингент, сроки обучения, преподаватель (список преподавателей) и аудитория (список аудиторий).

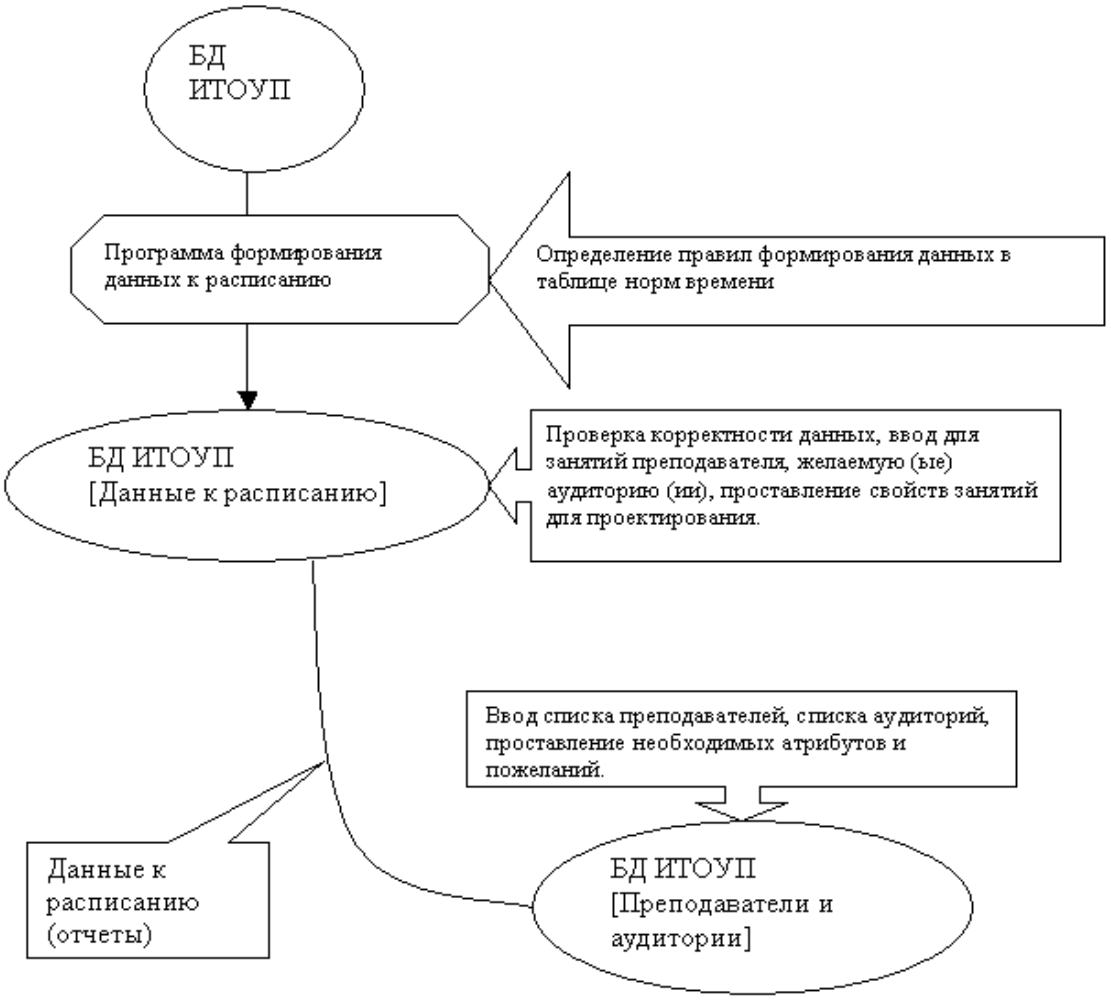



Рисунок 3.3. – Получение информации по данным к проектированию расписания занятий

Стоит отметить, что для ускорения процесса проектирования расписания не лишним было бы распределить аудиторный фонд по занятиям в полуавтоматическом режиме, при этом следует, по меньшей мере, обеспечить саму возможность вмещения занятий в распределенный аудиторный фонд. Для составления расписания также необходима информация о пожеланиях преподавателей, контингента и аудиторных пожеланиях. Речь идет об освобождении конкретных пар в нужные дни для каждого из объектов.

Каждое из учебных заведений имеет свою специфику. Невозможно учесть все возможные способы проведения занятий и автоматически сформировать действительно корректные данные. Но многие аспекты следует учитывать. Например, планирование занятий, на которые студенты ходят не на основе административного деления на группы и подгруппы. К таким занятиям относятся занятия по

иностранному языку, физкультуре, выбору и дисциплинам специализаций. Решением этого вопроса может послужить объединение таких занятий в списки с указанием одновременности их проведения.

Кроме того, для некоторых занятий требуются определить некоторые свойства, определяющие правила проектирования расписания для них (количество выделяемых аудиторий, количество пар проводимых подряд, определение дня недели для проведения занятия и др.). Определение свойств занятий и работа со списками занятий ложится на диспетчера, при работе с данными к расписанию.

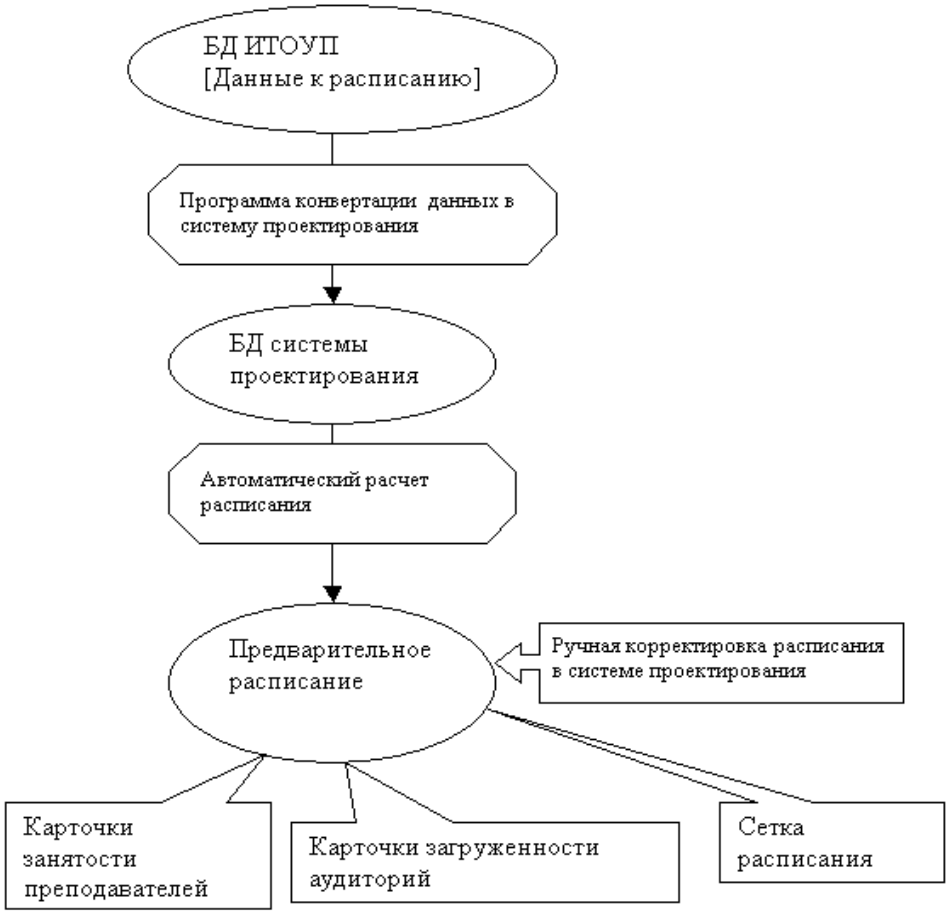



Рисунок 3.4. – Получение информации по расписанию занятий

## 3.3 Реализация программного комплекса поддержки организации учебного процесса

Как уже отмечалось ранее, разработка обширной системы автоматизации процесса требует от разработчика максимальной масштабируемости системы. Так, чтобы ее возможно было приспособить к потребностям любого предприятия,

где планируется ее внедрение. Во многих учебных заведениях сейчас уже разработаны собственные решения некоторых частей организации учебного процесса.

Система организации учебного процесса спроектирована достаточно демократично к любым вмешательствам в наполнение ее базы данных. Если в конкретном учебном заведении имелась бы программа, автоматизирующая один из этапов описанных выше, то требуется только написание сервиса, который бы заполнил специальную структуру данных в базе, а система по обеспечению остальных этапов продолжала бы работать.

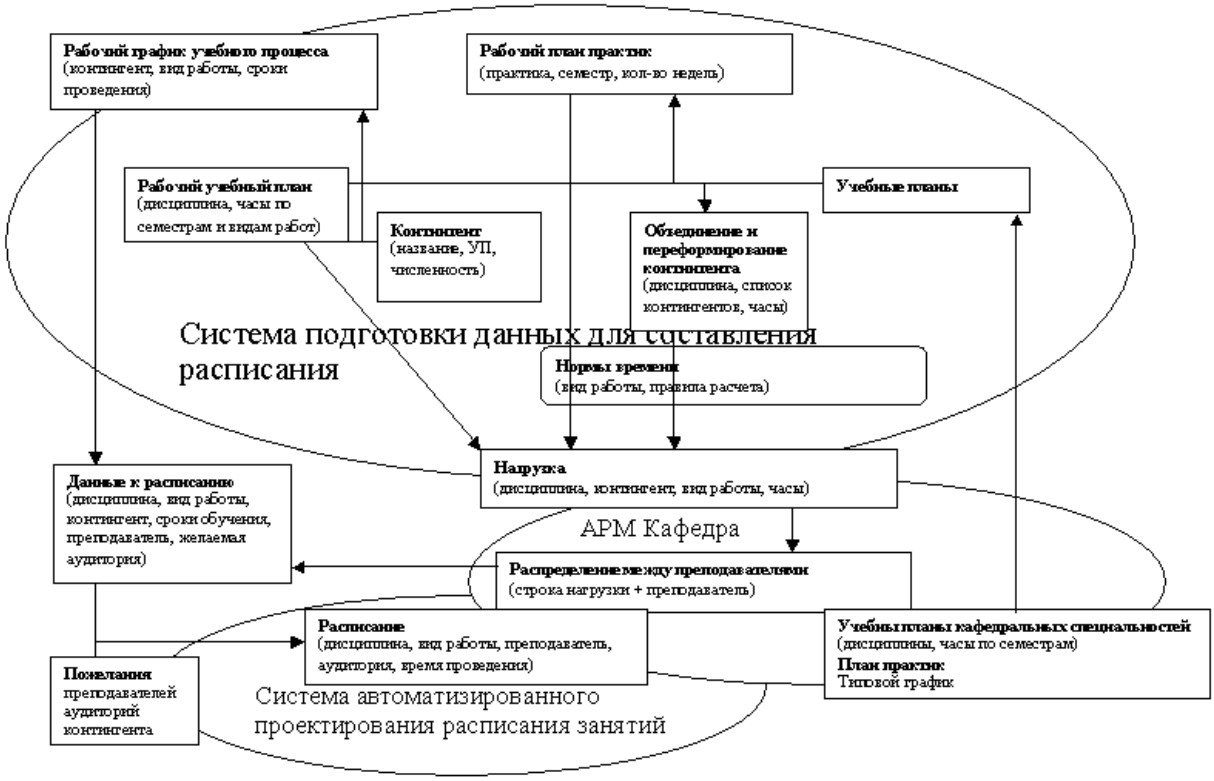



Рисунок 3.5. – Структура информационной технологии организации учебного процесса

На данный момент разработана «Информационная технология поддержки организации учебного процесса». Хотя данная система не полностью автоматизирует организацию учебного процесса, но, очевидно, является замкнутой и самодостаточной, при этом достаточно полно поддерживает технологическую цепочку работы учебного отдела. Сейчас система состоит из трех основных программных

подсистем: «Система подготовки данных к расписанию», «АРМ Кафедра», «Система автоматизированного проектирования расписания».

«Система подготовки данных для составления расписания» включает в себя работу с учебными планами, графиком учебного процесса, расчету нагрузки и работой с контингентом студентов. Обычно, эта программа устанавливается у работников учебного отдела, отвечающих за учебный процесс в целом. Именно она отвечает за мониторинг состояния всех аспектов процесса учебного года. Эта программа позволяет выводить все виды отчетов, начиная с учебного плана и заканчивая сеткой расписания. В нее включены модули по контролю заполнения учебных и рабочих учебных планов, формированию спотачивания контингентов и проектирования рабочего графика учебного процесса.

На каждую кафедру учебного заведения можно также установить программу по распределению нагрузки между преподавателями – «АРМ Кафедра». Здесь уже обеспечивается наполняемость данных к расписанию: расстановка соответствия между проводимыми занятиями и преподавателями введение, если необходимо аудиторий, в которых необходимо проводить занятия, а также различных пожеланий преподавателей. Кроме того, программа позволяет редактировать учебные планы, с последующим переносом их в основную базу. Полномочия переносить созданный кафедрой учебный план имеется только лишь у специалиста учебного отдела.

И, наконец, основная и главная составляющая информационной технологии организации учебного процесса – это программа, позволяющая рассчитывать расписание занятий. «Система автоматизированного расчета расписания» загружает необходимые данные из базы данных ИТОУП, затем, как это следует из названия, автоматически проектирует и позволяет вносить ручные корректировки диспетчеров по расписанию. После чего она отправляет результат своей работы в базу данных ИТОУП.

# 4 Применение логико-эвристического подхода при создании программного комплекса для определения параметров тепловых сетей

## 4.1 Математическая модель для задачи определения параметров тепловых сетей на стадии схемы теплоснабжения и проектов тепловых сетей

В настоящее время на стадии разработки схем теплоснабжения и проектов тепловых сетей актуальной является задача оптимизации развития и реконструкции сложившихся систем, когда вместе с выбором параметров вновь проектируемых необходимо определять способы реконструкции существующих тепловых сетей, включая выбор оптимальной трассировки с учетом обеспечения аварийных гидравлических режимов.

В различных программных комплексах, разработанных для расчетов гидравлических режимов, используются программы для конструктивных расчетов по выбору диаметров [57]. В этих программах используется метод, при котором выбирается удельная потеря давления вдоль главной магистрали, исходя из располагаемого напора на источнике и требуемого давления у самого удаленного потребителя. Под эту удельную потерю давления и подбираются диаметры тепловых сетей. В таких расчетах не учитываются технико-экономические показатели сети. В разветвленных тепловых сетях большой протяженности со сложным рельефом местности и наличием существующих участков применение такого метода не позволяет принять обоснованных проектных решений.

Пусть тепловая сеть задана графом исходных данных

$$\{V, E, \delta\} \qquad (4.1),$$

где $V$ – множество вершин (потребители, источники тепловой энергии, тепловые пункты, регуляторы давления), $E$ –ребра (участки сети, включая новые и существующие), $\delta$ – совокупность отношений и операций, определенных на вершинах и ребрах графа, задающих их связность. В каждой из вершин задается кортеж

$$\langle h, \Delta H, \Delta P \rangle \qquad (4.2),$$

где h – отметка земли, ΔH – потери тепла (для потребителей) или мощность (для источников), а ΔP – изменение напора (для насосных станций). Для существующих участков в ребрах заданы множества трубопроводов T, каждый элемент которого

$$t_i = <D, h, к_э> \quad (4.3),$$

где D – внешний диаметр трубопровода, h – толщина стенки, $к_э$ – коэффициент шероховатости.

Результатом определения параметров тепловой сети будет граф

$$\{V, V_R, E_R, \delta_R\} \quad (4.4),$$

где $V_R$ – добавленные насосные станции и узлы дросселирования, $\delta_R$ – изменения связности графа, в связи с добавлением множества узлов $V_R$, а $E_R$ – ребра участков сети, на которых определены кортежи из (4.3).

Следует отметить, что (4.4) получается в виде преобразования исходного (4.1) в результате итерационного преобразования системы. При этом на результат налагаются ограничения на расчетные значения тепловых потерь, расходы сетевой воды, потери давления в подающем и обратном трубопроводах на каждом из участков и др. [47], а оптимизация производится по минимуму расчетных затрат

$$З_{сум} = \sum З_{уч_i} + \sum З_{нс_i} \rightarrow min \quad (4.5),$$

где $Зуч_i$ – затраты на реконструкцию (для существующих) или строительство (для новых) трубопроводов на участке, а $Знс_i$ – затраты на строительство насосной станции.

Следует помнить, что исходные данные зачатую представляют собой кольцевую сеть, содержащую несколько источников тепловой энергии, т. е. граф (4.1) содержит циклы. Поэтому при проведении расчета потребуется определение оптимальной трассировки. Кроме того, в соответствии с требованиями к разработке схем теплоснабжения [1], должна быть обеспечена надежность теплоснабжения потребителей. Значит при построении графа результатов расчета (4) следует рассматривать аварийные ситуации на любом из участков с учетом сохранения нор-

мативного обеспечения потребителей. С учетом сказанного, можно предложить следующий алгоритм проведения расчета (рисунок 4.1):

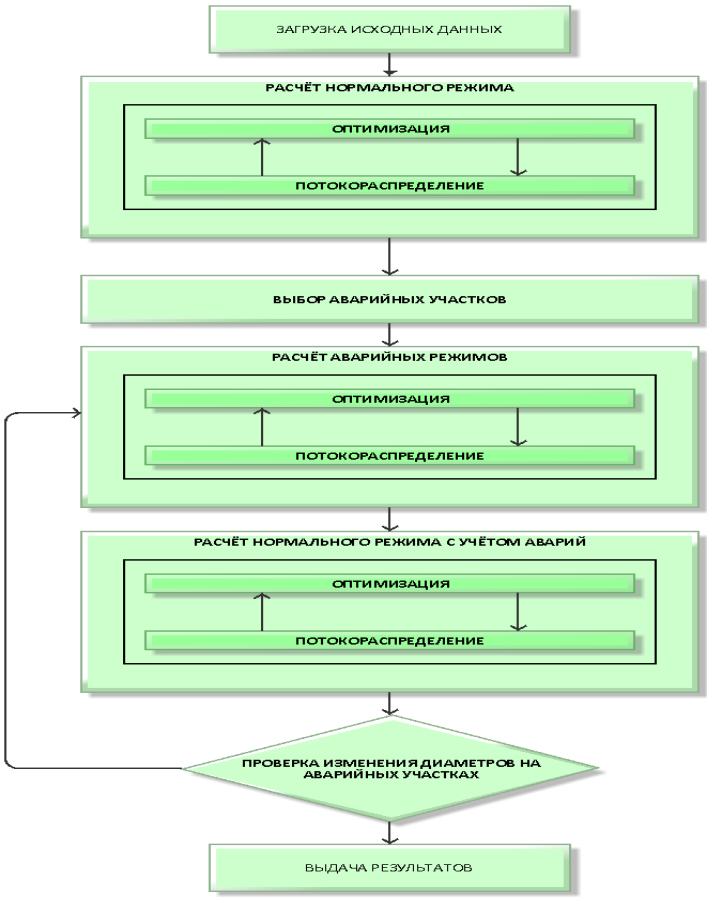



Рисунок 4.1. – Укрупненная блок схема расчета

**Ст.1.** Выполняется расчет нормального режима без учета аварийных ситуаций. На этой стадии расчетов для кольцевой сети выбирается оптимальная трассировка. Для новых участков определяется диаметр, а для существующих участков определяется необходимость реконструкции и способ реконструкции.

**Ст.2.** Определяются участки, на которых будут рассмотрены аварийные ситуации, исходя из расчета Ст.1. Аварийные ситуации рассматриваются не на всех участках. Количество аварийных ситуаций, выбирается исходя из необходимости обеспечения потребителей теплом в любой аварийной ситуации.

**Ст.3.** Выполняется расчет аварийных ситуаций определенных, в Ст.2. Для каждой аварийной ситуации учитывается коэффициент снижения подачи тепла, зависящий от величины диаметра на аварийном участке. На этой стадии расчетов определяется оптимальная трассировка для обеспечения потребителей теплом в

каждой аварийной ситуации. При расчете, минимальные значения диаметров по участкам принимаются по результатам расчетов нормального режима Ст.1. Проводится анализ результата расчета для каждой аварийной ситуации, и отслеживаются участки, на которых полученные диаметры превышают значения диаметров, выбранных при расчете нормального режима Ст.1.

**Ст.4.** Выполняется расчет нормального режима с учетом аварийных ситуаций. Расчет аналогичен Ст.1. Отличие заключается в том, что для каждого участка принимаются минимальные диаметры, полученные в результате расчета аварийных ситуаций.

**Ст.5.** После расчета нормального режима Ст.4 проверяются участки сети, на которых были рассмотрены аварийные ситуации. В случае увеличения диаметров аварийных участков, расчет аварийных ситуаций для данных участков повторяется Ст.3. После этого вновь производится расчет нормального режима с учетом аварийных ситуаций Ст.4.

Расчеты прекращаются, в случае если диаметры аварийных участков после расчета в Ст.4 не изменяются.

Решение задачи по выбору оптимальных параметров тепловых сетей обеспечивает совместная работа блоков «ПОТОКОРАСПРЕДЕЛЕНИЕ» и «ОПТИМИЗАЦИЯ». Блок «ПОТОКОРАСПРЕДЕЛЕНИЕ» позволяет определить трассировку в кольцевой сети, а «ОПТИМИЗАЦИЯ» решает задачу оптимизации параметров тупиковых сетей, минимизируя (4.5).

## 4.2 Реализация программного комплекса определения параметров тепловых сетей

Опираясь на описанную модель определения параметров теплосети в Иркутском филиале ЗАО «Сибирский ЭНТЦ» был разработан программный комплекс «КЕДР» [56]. Программный комплекс позволяет автоматизировать определение оптимальных параметров тупиковых и кольцевых тепловых сетей с учетом обеспечения работы в аварийных ситуациях, а также способствует снижению

трудозатрат по разработке проектной документации и в целом улучшает технологию проектирования.

«КЕДР» состоит из 2-х частей: графической оболочки и расчетного модуля.

Графическая оболочка является программным расширением AutoCAD, т. е. встраивается в среду AutoCAD. Только с помощью графической оболочки пользователь имеет возможность работать с данными в программном комплексе. Функциональные возможности графической оболочки следующие:

- Формирование схемы сети.
- Заполнение исходных данных.
- Контроль корректности исходных данных.
- Отправка данных для расчета на сервер.
- Получение результатов расчета с сервера.
- Формирование отчетов по результатам расчета (гидравлический расчет, кап. вложения и др.).

Построение и оформление пьезометров по результатам расчета.

В исходных данных необходимо задать параметры по участкам сети (длина, вид компенсаторов, способ прокладки и др.) и узлам (тепловая нагрузка, отметка земли, в случае необходимости ограничения по давлению и др.). Помимо заполнения параметров элементов схемы заполняются общие данные проекта, где указываются коэффициенты удорожания, прирост коэффициента шероховатости и другие технико-экономические показатели.

После того как исходные данные сформированы, схема может быть отправлена на расчет в «облако» на сервер. Графическая оболочка отправляет исходные данные на расчет и получает результаты расчета посредствам общения с «облаком». Использование облачной технологии позволяет проектировщику начать работу быстро и легко сразу же после установки графической оболочки: сам расчет можно выполнить, подключившись к сети интернет.

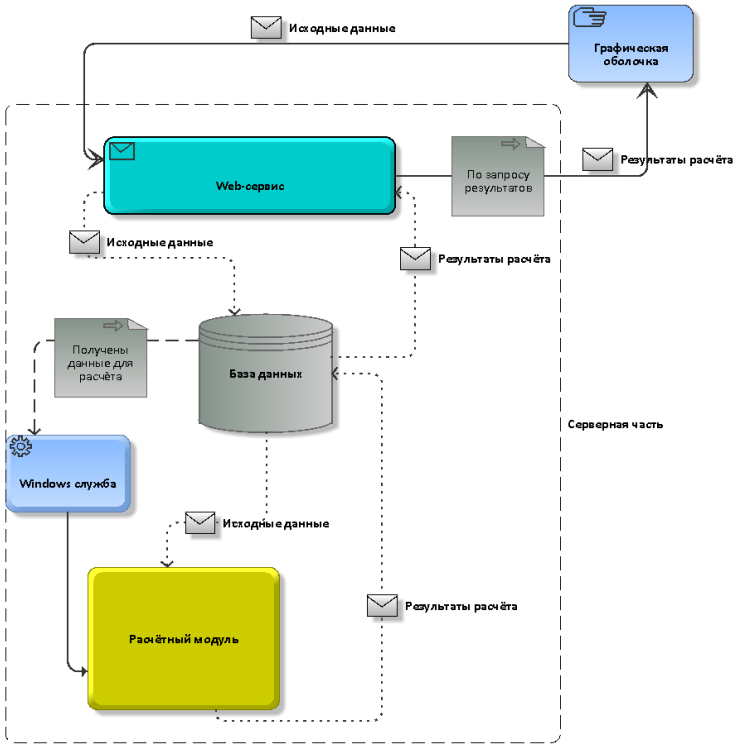



Рисунок 4.2. – Процесс передачи данных в «КЕДРе»

Web-сервис предназначен для организации коммуникации серверной части с графической оболочкой. Получив исходные данные от графической оболочки, web-сервис сохраняет их в базу данных, которая служит хранилищем всех исходных данных и всех результатов расчетов. Windows служба, запущенная на сервере, которая постоянно мониторит базу данных, при появлении нового запроса на расчет запускает расчетный модуль. Расчетный модуль производит расчет по исходным данным и по его завершению сохраняет результаты в базу данных, после чего завершает свою работу.

Отправив данные на расчет, графическая оболочка запрашивает состояние расчета у web-сервиса, и как только результаты расчета появляются в базе данных, web-сервис отправляет их в графическую оболочку. Процесс передачи данных в программном комплексе «КЕДР» представлен на рисунке 4.2.

Результат расчета в графической оболочке представляется схемой сети (рисунке 3) на которой наглядно представлена полученная информация по:

— диаметрам трубопроводов на проектируемых участках;

— существующим участкам, подлежащим реконструкции, с определением способа реконструкции;

— местам расположения и параметрам насосных станций и узлов дросселирования;

— напорам сетевых насосов на источниках.

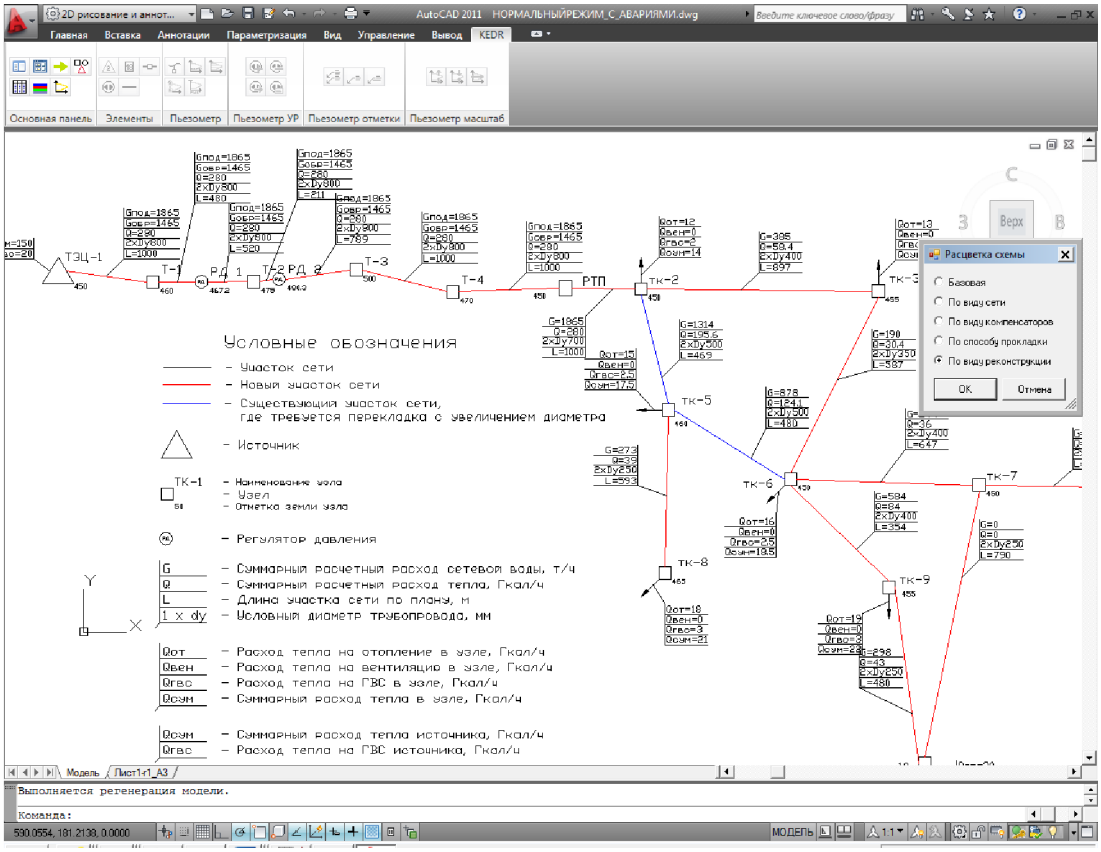



Рисунок 4.3. – Схема результата расчета

Построение пьезометрических графиков (рисунок 4.4.) возможно только из файлов результата расчета. На графике строятся линии расчетного режима, ограничения по подающему и обратному трубопроводу, линии режимов при максимальном водозаборе из подающего и обратного трубопровода. По оси абсцисс строится план трассы. Помимо оформления графика (масштабирование, простановка выносных линий, обрезка ответвлений), имеется возможность строить линию статического давления, корректировать результат расчета, редактируя узлы регулирования (добавление, удаление, перемещение, изменение напора) или изменяя значение давления на источниках.

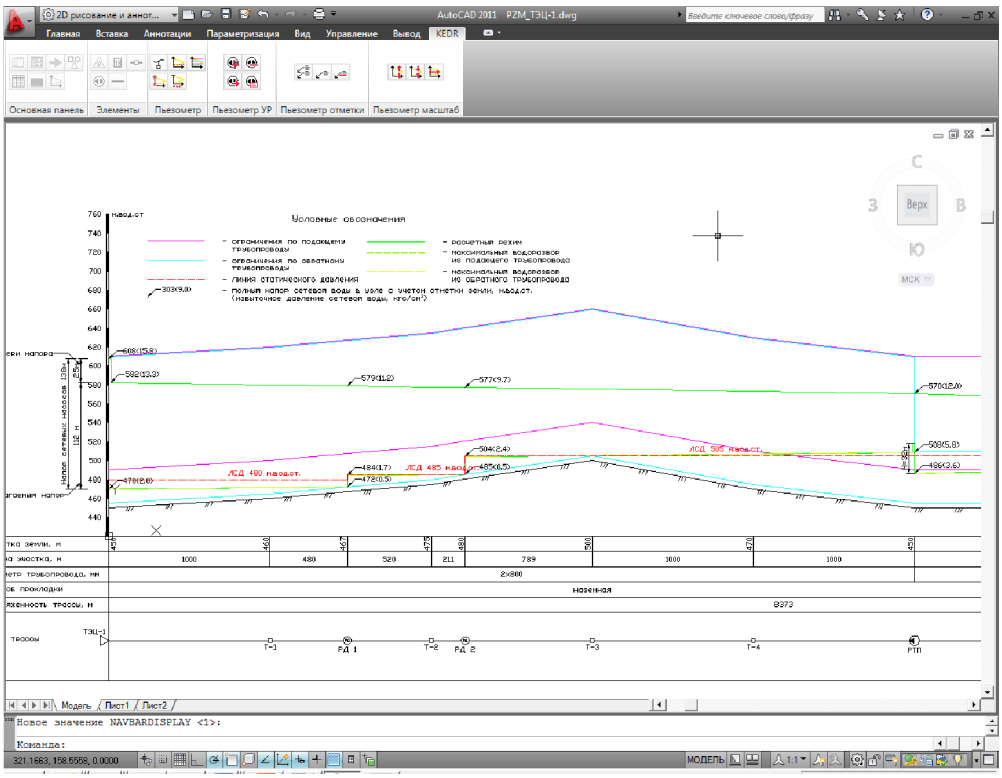


Рисунок 4.4. – Пьезометрический график

Программный комплекс «КЕДР» решает задачу по выбору оптимальных параметров тепловых сетей с учетом выбора оптимальной трассировки и обеспечения аварийных гидравлических режимов. Программный комплекс представляет собой программное обеспечение, функциональность которого подходит не только компаниям занимающимся проектированием тепловых сетей, но и эксплуатационным организациям, принимающим участие в развитии теплоснабжения населенных пунктов. Как показала практика, использование программного комплекса приводит к сокращению затрат на развитие тепловой сети до 40%.

«КЕДР» сертифицирован и успешно используется в подразделениях АО «Сибирский ЭНТЦ» уже более 3-х лет. Использование программного комплекса значительно повысило качество выполняемых работ и привело к сокращению сроков разработки проектной документации.

## Заключение

В соответствии с поставленными целями в диссертационной работе получены следующие результаты:

1. Построены математические модели, которые позволяют применить логико-эвристический подход для решения задач различных предметных областей: поддержки организации нормативного содержания сети автомобильных дорог, организации учебного процесса, проектирования схем теплоснабжения.

3. Предложены численные алгоритмы решения задач сетевого планирования, как итерационное построение многоосновных алгебраических систем, удовлетворяющих ограничениям.

4. Разработан программный комплекс, реализующий алгоритмы решения задач сетевого планирования для поддержки организации нормативного содержания сети автомобильных дорог, внедренный в ОГКУ «Дирекция по строительству и эксплуатации автомобильных дорог Иркутской области».

5. Разработан программный комплекс, решающий информационные задачи поддержки организации учебного процесса ВУЗов, внедренный в Иркутском Национальном исследовательском техническом университете (ИРНИТУ) и Иркутском государственном университете путей сообщения (ИрГУПС).

6. Разработан программный комплекс, позволяющий определять параметры тепловых сетей на стадии проектирования схем теплоснабжения, который используется в АО «Сибирский ЭНТЦ».

## Список использованных источников

1. Постановление Правительства РФ от 22 февраля 2012 № 154 О требованиях к схемам теплоснабжения, порядку их разработки и утверждения

2. Распоряжение Правительства РФ от 29.08.2014 № 1662-р.

3. Постановление Правительства РФ от 03.11.2015 № 1191 О некоторых вопросах взимания платы в счет возмещения вреда, причиняемого автомобильным дорогам общего пользования федерального значения транспортными средствами, имеющими разрешенную максимальную массу свыше 12 тонн

4. ГОСТ Р 52289-2004 Технические средства организации дорожного движения. Правила применения дорожных знаков, разметки, светофоров, дорожных ограждений и направляющих устройств

5. ГОСТ Р 52290-2004 Технические средства организации дорожного движения. Знаки дорожные. Общие технические требования.

6. ГОСТ Р 51256-2011 Технические средства организации дорожного движения. Разметка дорожная. Классификация. Технические требования

7. ГОСТ Р 50970-2011 Технические средства организации дорожного движения. Столбики сигнальные дорожные. Общие технические требования. Правила применения

8. ГОСТ 26804-2012 Ограждения дорожные металлические барьерного типа. Технические условия

9. Александров С. Г. Применение системы АДТ для решения задач сетевого планирования / С. Г. Александров, В. И. Мартьянов // Интеллектуализация программных средств : сб. науч. тр. / – Наука. Сиб. отд-ние - Новосибирск, 1990. – С. 160-168.

10. Беллман Р. Динамическое программирование / Р. Беллман – М. : Изд-во иностранной литературы, 1960. – 400 с.

11. Близниченко С. С. Теоретические основы разработки комплексной автоматизации системы управления автомобильными дорогами / С. С. Близниченко // Особенности проектирования, строительства и эксплуатации автомобильных дорог в Восточно-Сибирском регионе : сб. науч. тр. / – Иркутск, 1998. - С. 34-39.

12. Боброва Т. В. Проектно-ориентированное управление производством работ на региональной сети автомобильных дорог / Т. В. Боброва – Омск : СибАДИ, 2006. – 332 с.

13. Васильев С. Н. Логические модели и управление в реальном времени / С. Н. Васильев, А. К. Жерлов // Всесоюзн. научн.-технич. конф. Интеллектуальные системы. – Куйбышев, 1991. – С. 33-38.

14. Васильев С. Н. Об исчислении типово – кванторных формул / С. Н. Васильев, А. К. Жерлов // ДАН, т.343, N 2. – 1995. – С. 583-585.

15. Васильев С. Н. Интеллектное управление телескопом / С. Н. Васильев, Е. А. Черкашин // Сибирский журнал индустриальной математики. – 1998. т. 1, № 2 – С. 81-98.

16. Васильев Ф. П. Линейное программирование / Ф. П. Васильев, А. Ю. Иваницкий – М. : Факториал, 1998. – 176 с.

17. Вяткин И. В. Комбинаторные проблемы в задаче организации учебного процесса / И. В. Вяткин, В. И. Мартьянов // Труды XI международной Байкальской школы-семинара – Иркутск, 1998. – С. 34-37.

18. Габасов Р. С. Основы динамического программирования / Р. С. Габасов, Ф. М. Кириллова – Минск: Изд-во БГУ, 1975. — 262 с.

19. Гончаров С.С. Сигма-программирование / С.С. Гончаров, Д.И. Свириденко // Вычислительные ситемы: Логико-математические аспекты МОЗ : сб. науч. тр. в.107. / – ИМ СО АН СССР – Новосибирск, 1985. С.3-29.

20. Грофф Д. SQL: Полное руководство / Д. Грофф, П. Н. Вайнберг – Киев: BHV, 1998. – 608 с.

21. Губка А. С. Модель построения расписаний работы производственных объектов / А. С. Губка // Авіаційно-космічна техніка і технологія. – 2002 - Вип. 27 – С. 212-217.

22. Данциг Д. Линейное программирование: его применения и обобщения / Д. Данциг – М. : Прогресс, 1966. – 600 с.

23. Жилин Н. С. Информационно-справочные системы для органов управления дорожным хозяйством / Н. С. Жилин // Наука и техника в дорожной отрасли. – 1998. – №3. – С. 2-3.

24. Карманов В. Г. Математическое программирование / В. Г. Карманов – М. : Наука, 1986. – 288 с.

25. Кнут Д. Искусство программирования для ЭВМ: Основные алгоритмы / Д. Кнут – М. : Мир, 1976. – 736 с.

26. Кнут Д. Искусство программирования для ЭВМ: Сортировка и поиск / Д. Кнут – М. : Мир, 1978. – 848 с.

27. Конвей Р. Теория расписаний / Р. Конвей, В. Максвелл, Л. Миллер – М.: Наука, 1975. – 260 с.

28. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест — М. : МЦНМО, 2000. — 960 с.

29. Костенко В. А. Построение расписаний при совместном проектировании аппаратных и программных средств ВС реального времени /

В. А. Костенко // Труды Международной конференции «Параллельные вычисления и задачи управления» (PACO'2001). – М., 2001 – С. 125-141.

30. Костенко В. А. Способы представления и преобразования расписаний в итерационных алгоритмах / В. А. Костенко // «Программные системы и инструменты»: Тематический сборник факультета ВМиК МГУ им. Ломоносова №2 - М., 2001, С. 53-69.

31. Кулик Н. С. Проект системы управления региональной сетью автомобильных дорог (СУРАД) Иркутской области. / Н. С. Кулик, В. И. Мартьянов, Д. В. Пахомов // Вестник ИрГТУ. – 2014. – №4(74). – С. 150-155.

32. Кулик Н. С. Построение графа автомобильных дорог для системы взимания платы с большегрузного транспорта. / Н. С. Кулик, В. И. Мартьянов, Д. В. Пахомов // Вестник ИрГТУ. – 2016. – №4(111). – С. 96-101.

33. Лагоша Б. А. Комплекс моделей и методов оптимизации расписания занятий в вузе / Б. А. Лагоша, А. В. Петропавловская // Экономика и математические методы. – 1993. - Т. 29. Вып. 4. – С. 668-675.

34. Лорьер Ж. Л. Системы искусственного интеллекта / Ж. Л. Лорьер — М. : Мир, 1991. – 486 с.

35. Макконнелл С. Совершенный код: практическое руководство по разработке программного обеспечения / С. Макконнелл – 2-е издание – СПб. : Питер, 2010. – 889 с.

36. Мальцев А. И. Алгебраические системы / А. И. Мальцев – М.: Наука, 1967. – 324 с.

37. Мартьянов В. И. Планирование информационных потоков в иерархической системе / В. И. Мартьянов, В. В. Сухорутченко, В. В. Окунцов // Прикладные системы. ИАП РАН. - 1992. – С. 46-58.

38. Мартьянов В. И. Применение логики для решения задач реального времени / В. И. Мартьянов // Логика и семантическое программирование. – Вып.146:Вычислительные системы. – 1992. – С. 170-173.

39. Мартьянов В. И. Теоретический и реализационный аспекты некоторых классов задач сетевого управления / В. И. Мартьянов, И. В. Вяткин, Э. Ю. Могильницкий // Труды Международ. конф. «Проблемы управления и моделирования в сложных системах». – Самара, 1999. – С. 203-208.

40. Мартьянов В. И. Логико-эвристические методы сетевого планирования и распознавание ситуаций / В. И. Мартьянов // Труды Международ. конф. «Проблемы управления и моделирования в сложных системах» – Самара, 2001. – С. 203-215.

41. Мартьянов В. И. Проект системы мониторинга и поддержки управления сетью автомобильных дорог общего пользования Иркутской области /

В. И. Мартьянов, Н. Н. Самбаров, Д. В. Пахомов, А. С. Симонов // Винеровские чтения – Иркутск, 2005, С. 137–138.

42. Мартьянов В. И. Сетевое планирование содержания сети автомобильных дорог Иркутской области / В. И. Мартьянов, Д. В. Пахомов, В. В. Архипов // Новые технологии в инвестиционно-строительной сфере и ЖКХ – Иркутск, 2005. – С. 123–129.

43. Мартьянов В. И. Математические вопросы и программная реализация поддержки организации учебного процесса / В. И. Мартьянов, Д. В. Пахомов // Вестник Иркутского государственного технического университета – 2007. – №1 (29) – С. 143-147.

44. Мартьянов В. И. Организация рационального управления содержанием региональной сети автомобильных дорог / В. И. Мартьянов, Д. В. Пахомов, В. В. Архипов // Современные технологии. Системный анализ. Моделирование. ИрГУПС. – 2009. – №2(22) - С. 66-72.

45. Мартьянов В. И. Обзор приложений логико-эвристических методов решения комбинаторных задач / В. И. Мартьянов, В. В. Архипов, М. Д. Каташевцев, Д. В. Пахомов, А. С. Симонов // Материалы 3-ей Российской школы-семинара «Синтаксис и семантика логических систем». – Иркутск, 2010. - С 43-47.

46. Мартьянов В. И. Обзор приложений логико-эвристических методов решения комбинаторных задач высокой сложности / В. И. Мартьянов, В. В. Архипов, М. Д. Каташевцев, Д. В. Пахомов // Современные технологии. Системный анализ. Моделирование. ИрГУПС. – 2010. №4(28) - С. 205-211.

47. Меренков А. П. Теория гидравлических цепей / А. П. Меренков, В. Я. Хасилев – М. : Наука, 1985. – 276 с.

48. Месарович, М. Общая теория систем: математические основы / М. Месарович, Я. Такахара – М. : Мир, 1978. – 312 с.

49. Новиков Ф. А. Дискретная математика для программистов: учебник для вузов / Ф. А. Новиков – 2-е издание – СПб. : Питер, 2005. – 364 с.

50. Овчинников А. А. Сетевые методы планирования и организации учебного процесса / А. А. Овчинников, В. С. Путинский, Г. Ф. Петров — М. : Высшая школа, 2012. – 156 с.

51. Пахомов Д. В. Свид. 2005611861 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Программа подготовки данных для системы проектирования расписания / Д. В. Пахомов; заявитель и правообладатель ГОУ ВПО ИрГТУ.

52. Пахомов Д. В. Свид. 2005611863 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Программа

ведения учебных планов по дневной, заочной и ускоренной формам обучения / Д. В. Пахомов; заявитель и правообладатель ГОУ ВПО ИрГТУ.

53. Пахомов Д. В. Поддержка информационной составляющей технологии организации учебного процесса / Д. В. Пахомов // Вестник Иркутского государственного технического университета – 2006. – №4 (28) – С. 42-44.

54. Пахомов Д. В. Свид. 2010611861 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Программа обработки и интеграции данных видеопаспортизации для мониторинга сети автомобильных дорог / Д. В. Пахомов; заявитель и правообладатель ГОУ ВПО ИрГТУ

55. Пахомов Д. В. Автоматизация создания проектов организации дорожного движения для автомобильных дорог / Д. В. Пахомов, М. Д. Каташевцев, В. И. Мартьянов, А. А. Степаненко // Современные технологии. Системный анализ. Моделирование. ИрГУПС. – 2012. №3(35). - С. 56-61.

56. Пахомов Д. В. Свид. 2013612462 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Программный комплекс для определения оптимальных параметров тепловых сетей «КЕДР» / Д. В. Пахомов, В. С. Тонгалюк, А. С. Московских, А. С. Бочкарев; заявитель и правообладатель ЗАО "Сибирский ЭНТЦ".

57. Первовский Ю. А. Гидравлический расчет в тепловых сетях: мертвый подход или живая модель / Ю. А. Первовский, О. В. Анподистов // Новости теплоснабжения – 2004, № 1(41) - С. 69.

58. Самбаров Н. Н. О состоянии автомобильных дорог Иркутской области / Н. Н. Самбаров, В. И. Мартьянов, Д. В. Пахомов, А. С. Симонов // Дальний Восток. Автомобильные дороги и безопасность движения. – 2006. – №6 – С. 167-170.

59. Сухарев А. Г. Курс методов оптимизации: учебное пособие / А. Г. Сухарев, А. В. Тимохов, В. В. Федоров – М. : Наука, 1986. – 368 с.

60. Схрейвер А. Теория линейного и целочисленного программирования / А. Схрейвер – Том 1 – М.: Мир, 1991. – 360 с.

61. Сыготина М. В. Современные подходы к организации учебного процесса / М. В. Сыготина // Материалы IV Всероссийской научно-методической конференции «Развитие тестовых технологий в России». М., 2014. – С. 26-33.

62. Танаев В. С. Введение в теорию расписаний / В. С. Танаев, В. В. Шкурба – М. : Наука, 1975. – 256с.

63. Танаев В. С. Теория расписаний: многостадийные системы / В. С. Танаев, Ю. Н. Сотсков, В. А. Струсевич – М. : Наука, 1989. – 328 с.

64. Фаулер М. Архитектура программных приложений. / М. Фаулер – М.: Издательский дом «Вильяме», 2006. – 540 с.

65. Фаулер М. Рефакторинг. Улучшение существующего кода. / М. Фаулер, К. Бек, Д. Брант – СПб.: Символ-плюс, 2003. – 432 с.

66. Харрари Ф. Теория графов. / Ф. Харрари – М.: Мир, 1973. – 360с.

67. Ху Т. Целочисленное программирование и потоки в сетях / Т. Ху – М.: Мир, 1974. – 520 с.

68. Шевченко В. Н. Качественные вопросы целочисленного линейного программирования / В. Н. Шевченко – М.: Физматлит, 1995. – 190 с.

69. Яресь О. Б. Оценка эффективности управления промышленным предприятием в современных условиях: дис. … канд. эк. наук: 08.00.05, Владимирский государственный университет, Владимир, 2000. – 146 с.

70. Codd E. F. The Relational Model For Database Management Version 2 / E. F. Codd – Massachusetts, USA: Addison-Wesley, 1990. – 538 p.

71. Dasgupta S. Algorithms / S. Dasgupta, C. H. Papadimitriou, U. Vazirani — McGraw-Hill Higher Education, 2006. — 336 p.

72. Garvey M. A. Introduction to Object-Oriented Databases / M. A. Garvey, M. S. Jackson. // Inf. and Software Technol. – 31, № 10. – 1989. – p. 521-528

73. Hentenryck van P. Constraint Satisfaction in Logic Programming / Pascal Van Hentenryck – Cambridge, MA, USA: The MIT Press, 1989. – 356 p.

# Приложение А

Копия справки о внедрении «Системы мониторинга и поддержки управления дорожной сетью» в ОГКУ «Дирекция по строительству и эксплуатации дорог Иркутской области»

СПРАВКА

о внедрении результатов диссертационной работы программиста ИРНИТУ Д.В. Пахомова **ПРОГРАММНЫЕ КОМПЛЕКСЫ ЧИСЛЕННОГО РЕШЕНИЯ РЕСУРСНЫХ ЗАДАЧ НА ОСНОВЕ ЛОГИКО-ЭВРИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ.**

Настоящим подтверждаем, что программный комплекс **«Система мониторинга и поддержки управления дорожной сетью Иркутской области»** (созданный Иркутским национальным исследовательским техническим университетом), включающий компоненты:

- администрирования сетевой БД программного комплекса (автор Д.В. Пахомов);

- генерации отчетов (тех. паспорта, оценка ТЭС, ПОДД и др.) (автор Д.В. Пахомов);

- синхронного просмотра электронных линейных графиков и видеорядов, фотографий объектов (автор Д.В. Пахомов),

используется в производственной деятельности **ОГКУ «Дирекция по строительству и эксплуатации автомобильных дорог Иркутской области»** для решения информационных, организационных и иных задач.

Заместитель директора по ремонту и содержанию автодорог А.В.Зорин

Начальник отдела по содержанию автомобильных дорог Е.Н.Егунова

ОГКУ «Дирекция по строительству и эксплуатации автомобильных дорог Иркутской области» ОТДЕЛ ПО СОДЕРЖАНИЮ АВТОДОРОГ

## Приложение Б

Копия справки о внедрении «Системы поддержки учебного процесса» в Иркутском государственном университете путей сообщения

СПРАВКА

о внедрении результатов диссертационной работы программиста ИРНИТУ Д.В. Пахомова **ПРОГРАММНЫЕ КОМПЛЕКСЫ ЧИСЛЕННОГО РЕШЕНИЯ РЕСУРСНЫХ ЗАДАЧ НА ОСНОВЕ ЛОГИКО-ЭВРИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ.**

Настоящим подтверждаем, что программный комплекс **«Система поддержки учебного процесса»** , включающий компоненты:

- генерации рабочего учебного плана и расчета нагрузки (автор Д.В. Пахомов);

- администрирования сетевой БД программного комплекса (автор Д.В. Пахомов);

- распределенного заполнения учебных поручений преподавателей (автор Д.В. Пахомов);

- формирования рабочего графика учебного процесса для заочной формы обучения (автор Д.В. Пахомов);

- подготовки данных для расчета расписания (автор Д.В. Пахомов);

- генерации отчетов (сетки расписания групп, карточки преподавателей и аудиторий и др.) (автор Д.В. Пахомов);

используется в образовательной деятельности Иркутского государственного университета путей сообщения (ИрГУПС).

Начальник учебного отдела ИрГУПС И.А.Исакова

## Приложение В

Копия справки о внедрении «Системы поддержки учебного процесса» в Иркутском национальном исследовательском техническом университете

СПРАВКА

о внедрении результатов диссертационной работы программиста ИРНИТУ Д.В. Пахомова **ПРОГРАММНЫЕ КОМПЛЕКСЫ ЧИСЛЕННОГО РЕШЕНИЯ РЕСУРСНЫХ ЗАДАЧ НА ОСНОВЕ ЛОГИКО-ЭВРИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ.**

Настоящим подтверждаем, что программный комплекс **«Система поддержки учебного процесса»** , включающий компоненты:

- администрирования сетевой БД программного комплекса (автор Д.В. Пахомов);

- распределенного заполнения учебных поручений преподавателей (автор Д.В. Пахомов);

- подготовки данных для расчета расписания (автор Д.В. Пахомов);

- генерации отчетов (сетки расписания групп, карточки преподавателей и аудиторий и др.) (автор Д.В. Пахомов);

используется в образовательной деятельности Иркутского национального исследовательского технического университета (ИРНИТУ).

Начальник учебного отдела ИРНИТУ О.В.Солдатова

Подпись Солдатовой О.В
Верно
Специалист ОК
«20» 06 2017 г.

## Приложение Г

Копия акта о внедрении программного комплекса «КЕДР» в АО «Сибирский ЭНТЦ»

УТВЕРЖДАЮ
И.О. директора Иркутского филиала
АО «Сибирский ЭНТЦ»
_____ Н.А. Абрамов
«___» _____

АКТ

О внедрении программного комплекса для определения оптимальных параметров тепловых сетей «КЕДР»

Комиссия в составе: В. Е. Сычёва, начальника департамента по информационным технологиям Иркутского филиала, П. В. Коротких, руководителя группы отдела автоматизации проектных работ Иркутского филиала, В. С. Тонгалюка, руководителя группы отдела перспективного проектирования института «СибВНИПИЭнергопром», составила настоящий акт о том, что программный комплекс для определения оптимальных параметров тепловых сетей «КЕДР», автором которого является Д. В. Пахомов, внедрён в производственный процесс и используется отделами тепловых сетей и перспективного проектирования института «СибВНИПИЭнергопром» при разработке схем теплоснабжения и проектов тепловых сетей.

Использование программного комплекса «КЕДР», являющегося результатом диссертационного исследования Д. В. Пахомова «Программные комплексы и численные решения ресурсных задач на основе логико-эвристического моделирования», позволило повысить качество и скорость проектирования схем теплоснабжения и проектов тепловых сетей. Программный комплекс «КЕДР» (свидетельство о государственной регистрации программы для ЭВМ № 2013612462) прошёл сертификацию на соответствие требований нормативных документов: СНиП 41-02-2003 «Тепловые сети», ГОСТ Р ИСО 9127-94 «Документация пользователя и информация на упаковке потребительских программных пакетов», ГОСТ Р ИСО/МЭК 12119-2000 «Информационная технология. Пакеты программ. Требования к качеству и тестирование» о чём имеется сертификат соответствия № РОСС RU.СП15.Н00610 от 06.05.2013. Программа ориентирована на непосредственное использование специалистами, имеет удобный интерфейс, высокую степень реактивности в режиме диалога.

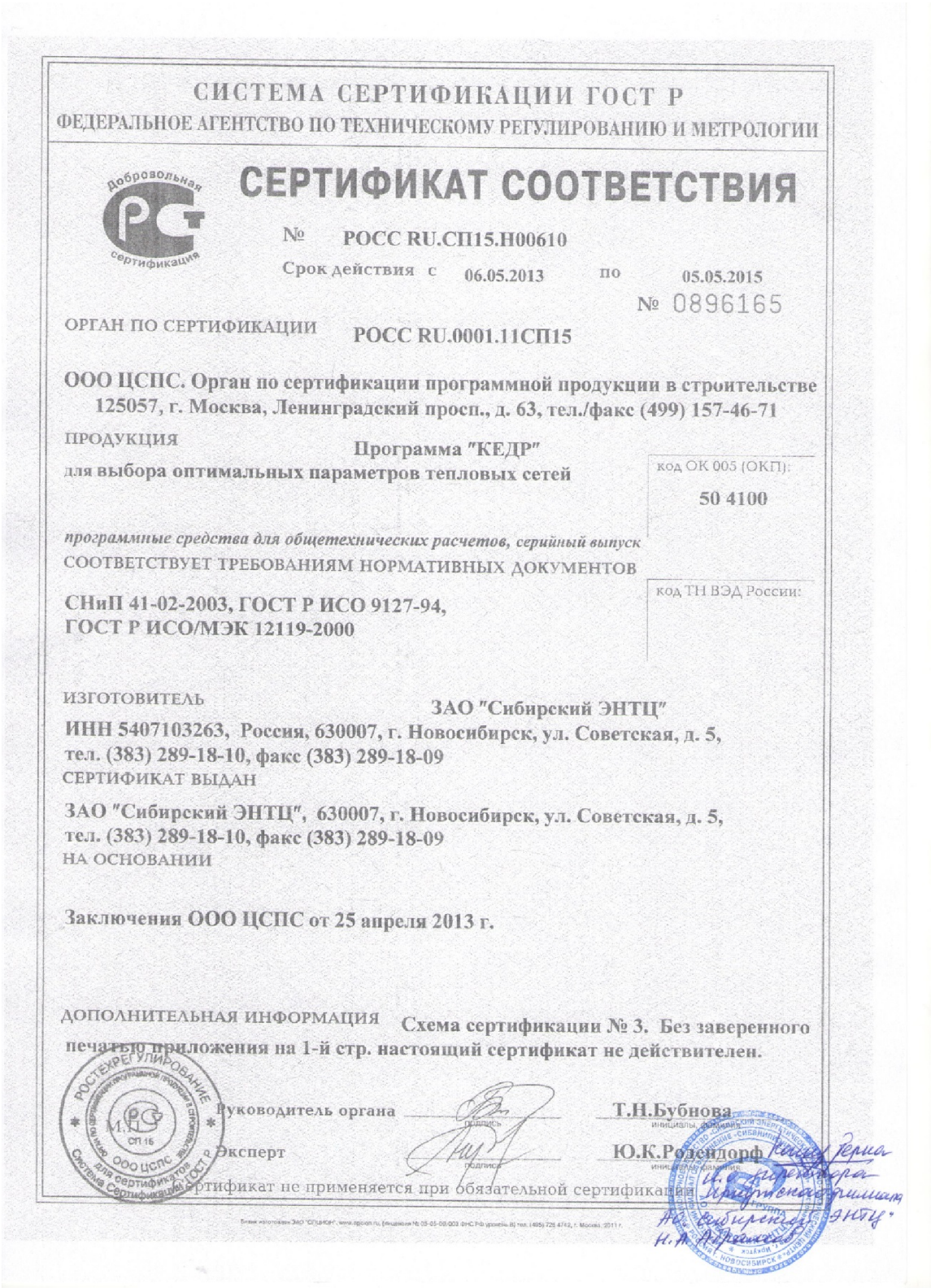
Председатель комиссии _____ В. Е. Сычёв

Члены комиссии _____ П. В. Коротких

_____ В. С. Тонгалюк

**Приложение Д**

Копия сертификата соответствия программного комплекса «КЕДР»



СИСТЕМА СЕРТИФИКАЦИИ ГОСТ Р

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

Добровольная сертификация

# СЕРТИФИКАТ СООТВЕТСТВИЯ

№ РОСС RU.СП15.Н00610

Срок действия с 06.05.2013 по 05.05.2015

№ 0896165

ОРГАН ПО СЕРТИФИКАЦИИ РОСС RU.0001.11СП15

ООО ЦСПС. Орган по сертификации программной продукции в строительстве
125057, г. Москва, Ленинградский просп., д. 63, тел./факс (499) 157-46-71

ПРОДУКЦИЯ Программа "КЕДР"
для выбора оптимальных параметров тепловых сетей

код ОК 005 (ОКП): 50 4100

*программные средства для общетехнических расчетов, серийный выпуск*

СООТВЕТСТВУЕТ ТРЕБОВАНИЯМ НОРМАТИВНЫХ ДОКУМЕНТОВ

код ТН ВЭД России:

СНиП 41-02-2003, ГОСТ Р ИСО 9127-94,
ГОСТ Р ИСО/МЭК 12119-2000

ИЗГОТОВИТЕЛЬ ЗАО "Сибирский ЭНТЦ"
ИНН 5407103263, Россия, 630007, г. Новосибирск, ул. Советская, д. 5,
тел. (383) 289-18-10, факс (383) 289-18-09

СЕРТИФИКАТ ВЫДАН
ЗАО "Сибирский ЭНТЦ", 630007, г. Новосибирск, ул. Советская, д. 5,
тел. (383) 289-18-10, факс (383) 289-18-09

НА ОСНОВАНИИ
Заключения ООО ЦСПС от 25 апреля 2013 г.

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ Схема сертификации № 3. Без заверенного печатью приложения на 1-й стр. настоящий сертификат не действителен.

РОСТЕХРЕГУЛИРОВАНИЕ ООО ЦСПС Система сертификации ГОСТ Р Для сертификатов М.П. СП 15

Руководитель органа Т.Н.Бубнова
подпись инициалы, фамилия

Эксперт Ю.К.Роздорф
подпись инициалы, фамилия

Сертификат не применяется при обязательной сертификации

Бланк изготовлен ЗАО "СПЦ-Юг", www.rpcsp.ru, (лицензия № 05-05-09/003 Ф-С РФ уровень В) тел. (495) 726 4742, г. Москва, 2011 г.

**ПРИЛОЖЕНИЕ**
**к сертификату соответствия № РОСС RU.СП15.Н00610 от 06.05.2013**
**(без сертификата не действительно)**

1. **Обозначение программной продукции**
Программа "КЕДР".

2. **Название программной продукции**
Программа выбора оптимальных параметров тепловых сетей.

3. **Версия 3**

4. **Состав решаемых задач:**
- выбор оптимальных параметров в кольцевых тепловых сетях;
- определение диаметров новых участков, способов реконструкции существующих участков, напоров и мест расположения насосных станций, напора сетевых насосов на источнике, оптимальных схем трассировки сети в нормальном и аварийных режимах;
- построение пьезометрических графиков для нормального и аварийных режимов.

5. **Соответствует требованиям пунктов нормативных документов по состоянию на 25 апреля 2013г.**
**СНиП 41-02-2003 "Тепловые сети":**
- раздел 4. Классификация, пп.4.1, 4.2;
- раздел 6. Схемы теплоснабжения и тепловых сетей. Надежность, п.6.29. Резервирование, пп.6.33 - 6.36;
- раздел 8. Гидравлические режимы, пп.8.1 - 8.13, 8.15, 8.17.

**ГОСТ Р ИСО 9127-94 "Документация пользователя и информация на упаковке потребительских программных пакетов":**
- раздел 6. Справочная документация (ОБ). Подраздел 6.1. Обозначение пакета (ОБ), п.6.1.1. Подраздел 6.3. Функциональное описание программного средства (ОБ), пп.6.3.1, 6.3.3. Подраздел 6.5. Использование программного средства (ОБ), пп.6.5.1 – 6.5.3.

**ГОСТ Р ИСО/МЭК 12119-2000 "Информационная технология. Пакеты программ. Требования к качеству и тестированне":**
- раздел 3. Требования к качеству. Подраздел 3.1. Описание продукта, пп.3.1.1, 3.1.3. Подраздел 3.2. Документация пользователя, пп.3.2.1 – 3.2.5.

6. **Программная документация**
Руководство пользователя графической оболочки "КЕДР", Иркутск, 2012, 84 с.
Техническая документация по программному комплексу "КЕДР", Иркутск, 2012, 34 с

**Руководитель органа по сертификации программной продукции в строительстве** **Т.Н.Бубнова**