САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра Системного программирования

Пузиков Александр Юрьевич

Разработка алгоритмов и программных средств управления проектами с использованием моделей и методов идемпотентной алгебры

Магистерская диссертация

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

Научный руководитель:
д. ф.-м. н., профессор Кривулин Н. К.

Рецензент:
инженер Смирнов К. К.

Санкт-Петербург
2014

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

Alexandr Puzikov

Development of algorithms and software tools for project control using models and methods of idempotent algebra

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Andrey Terekhov

Scientific supervisor:
professor Nikolai Krivulin

Reviewer:
Engineer Kirill Smirnov

Saint-Petersburg
2014

Оглавление

Введение **4**

1. Обзор существующих алгоритмов **6**
1.1. Управление длительностью проекта 6
1.1.1. График Ганта 6
1.1.2. Метод критического пути 7
1.1.3. MPM 8
1.1.4. PERT 9
1.1.5. GERT 10
1.2. Управление трудовыми ресурсами 11

2. Планирование проектов **13**
2.1. Идемпотентное полуполе 14
2.2. Алгебра матриц 14
2.3. Представление задачи планирования в терминах тропической математики 16
2.3.1. Just-in-time оптимизация проекта 16
2.3.2. Максимизация разброса времени завершения операций 17
2.3.3. Минимизация максимального времени цикла операций 19
2.3.4. Минимальное общее время (makespan) 21
2.3.5. Минимальное отклонение от директивных сроков 22

3. Программная реализация **23**
3.1. Архитектура разрабатываемой системы 23
3.2. Описание программного и технического обеспечения 24
3.2.1. Описание программного и технического обеспечения 24
3.2.2. Требования к программному обеспечению 25
3.2.3. Выбор языка и средств разработки 25
3.2.4. Идемпотентная алгебра 26
3.2.5. Диаграмма Ганта 27
3.2.6. Сетевой график 27
3.3. Руководство пользователя 28

4. Численные примеры **33**

Заключение **37**
Приложение 1. IDEF0 диаграмма ПП 39
Приложение 2. Описание классов пакета Tropical 41
Приложение 3. Описание классов пакета MainPackage 45
Приложение 4. Flow-time оптимизация 48

Введение

Управление проектами - один из важнейших способов квалифицированной организации труда на производстве. Для руководителя проекта это означает рациональное распределение операций проекта с учетом различных критериев оптимальности. Управление проектами, методы которого стали формироваться в середине прошлого века, за прошедшее время сложилось в специфическую область знаний и практическую методологию, широко применяемую в самых разных областях человеческой деятельности. Разработаны международные стандарты управления проектами [13], в соответствии с которыми строятся процессы управления самыми различными проектами от научно-исследовательских до строительных, а также любыми изменениями в компаниях. Проектная модель управления активно внедряется компаниями, работающими в различных отраслях.

Вместе с тем в настоящее время использование рассматриваемых технологий не везде получило большое распространение. Так, в России программное обеспечение по управлению проектом не имеет широкого круга пользователей в связи с тем, что во многих компаниях (исключая сферу ИТ) считается, что данное ПО является дорогостоящим инструментом, использование которого не дает ощутимых результатов ни в краткосрочной, ни в долговременной перспективе [10]. Тем не менее, во всем мире уже признана способность методов управления проектами радикально изменять ситуацию в менеджменте [14].

Основной задачей является разработка средств планирования времени операций в проекте.

В настоящее время существует огромное количество ПО для организации управления проектами. Данное ПО можно разделить по различным критериям – по способу реализации от desktop- до web-версий и по типу рассматриваемых проектов от стохастических до детерминированных. ПО не всегда реализует обширный набор методов, зачастую являясь узкоспециализированные лишь на одном из них. В основе большинства продуктов лежат алгоритмы, созданные еще в 50 - 60х годах прошлого века. К таким алгоритмам относятся "метод критического пути"(CPM), "метод графической оценки и анализа"(GERT), "техника оценки и анализа проектов"(PERT) и другие.

Если рассматривать самые популярные критерии оптимизации проектов, то не составит труда описать их с математической точки зрения. Большинство из них предстанут в виде различных задач оптимизации. Многие классические задачи оптимизации (задачи оптимизации на графах , задачи о назначениях, динамического программирования) представляются в идемпотентной алгебре в виде решения линейных уравнений, нахождения собственных чисел и векторов линейного оператора и прочим вычисления данного характера. За последние десятилетия идемпотентная алгебра превратилась в один из наиболее интенсивно развивающихся разделов математики, роль которого как теоретической дисциплины и эффективного инструмента решения практических задач в экономике, технике, управлении и других областях постоянно растет.

Естественным будет желание рассмотреть задачи управления проектами в свете идемпотентной (тропической) математики.

В связи с этим, целью данной работы является исследование возможности применения алгоритмов идемпотентной алгебры в области управления проектами (сетевого планирования), оценка достоинств и недостатков данных алгоритмов и создание ПО,

в котором будут реализованы описанные методы и алгоритмы.

Достижению поставленной цели способствует реализация частных задач:

1. изучить методы идемпотентной алгебры решения задач планирования;
2. построить на основе изученных методов алгоритмы планирования временных характеристик проекта;
3. реализовать алгоритмы в виде комплекса программных средств с помощью выбранных средств разработки ПО;

Практическая значимость работы состоит в том, что ее результаты могут быть использованы при решении задач планирования проектов в различных сферах трудовой деятельности.

Теоретической базой исследования послужили работы [3-6].

Структура данного исследования соответствует поставленным задачам. Первая глава посвящена обзору существующих на данных момент и наиболее распространенных алгоритмов по управлению проектами, а также рассмотрены различные виды ПО, их реализующие. Во второй главе рассматривается идемпотентная алгебра, а также представление задач оптимизации проектов в терминах идемпотентной математики. Следующая глава содержит основные моменты реализации рассмотренных алгоритмов в виде ПО, систематизируются алгоритмы по критериям оптимизации. Для реализации этой цели был проведён комплексный анализ различных сторонних библиотек и выявлены наиболее подходящие для данной ситуации. Четвертая глава содержит численные примеры использования разработанных алгоритмов. В заключении представлены алгоритмы дисциплины управления проектами, которые пока не представимы в терминах идемпотентной алгебры . Эти задачи могут быть решены в дальнейшем, в качестве перспективного развития ПО.

1. Обзор существующих алгоритмов

В качестве основной технологии визуализации проекта выбрана сетевая. Она наиболее распространена при планировании и контроле реализации сложного мероприятия (проекта). Базируется на теории графов. Структура проекта представляется в форме графа типа "сеть". В сети определим операции как узлы, а дуги это события(окончание либо завершение операции). Методами управления проектами называют алгоритмы сетевого планирования и контроля реализации проектов, а так же алгоритмы оценки трудоемкости и стоимости работы. В основном методы можно разделить на несколько групп:

1. управление интеграцией проекта;
2. управление рисками проекта;
3. управление трудовыми ресурсами;
4. управление длительностью проекта;

и другие. В данной работе рассматриваются только 3 и 4 пункты. Далее рассмотрены некоторые алгоритмы управления проектами.

1.1. Управление длительностью проекта

1.1.1. График Ганта

График Ганта - временная диаграмма, считается одной из старейших технологий планирования. Создан в начале XX в. Г.Л. Гантом в США. Достоинством этой диаграммы является её наглядность. Визуализация работ в виде отрезков, длина которых пропорциональна времени их выполнения облегчает восприятие. Это очень полезно, когда операции в проекте имеют последовательно-параллельную структуру.

В зависимости от характера элементов, обозначенных на оси ординат, приходится иметь дело с двумя видами временных диаграмм:

- диаграммы динамики выполнения работ;
- диаграммы использования исполнительских мощностей.

Для нас интерес представляет второй тип графика. На оси ординат этого графика отмечаются исполнители работ (ресурсы). Эту диаграмму зачастую используют для планирования и контроля использования исполнительских мощностей.

Рис. 1: Диаграмма Ганта

1.1.2. Метод критического пути

CPM (англ. Critical Path Method) – технология создана в 1957 году и возникла на базе графиков Гантта. Она относится к группе детерминированных технологий сетевого планирования. Результатом её использования является особый сетевой граф (представляющий операции проекта) с определенными расчетами. В сетевом графе не должно быть петель или зацикливаний. Результат этих расчетов – план реализации проекта. Сроки старта и завершения проекта определяются в начале процесса планирования с учетом внешних условий. Событием называется наступление некоторого четко определенного состояния. Операцией называется некоторая задача, для решения которой требуется время и которая должна быть выполнена между двумя событиями. Можно выделить следующие основные этапы:

1. Представление проекта в виде сетевого графика;
2. Определение времени, необходимого для выполнения каждой операции;
3. Определение даты (сроков) начала и окончания проекта;
4. Расчет сроков окончания и начала для каждой операции;
5. Расчет резервов времени;
6. Определение критической последовательности операций, от которой будет зависит своевременная реализация проекта;

Работы могут быть реальными, а могут быть фиктивными (с нулевой длительностью).Критический путь - это последовательность операций, наиболее важных для соблюдения установленного срока реализации проекта. Критический путь определяется в процессе анализа длительности выполнения операций в сети. Такой анализ основан на взаимосвязях операций и информации о длительности их выполнения.

Для каждого события в сети можно рассчитать сроки её реализации. Самый ранний срок наступления события рассчитывается по формуле

$$E_j = \max[E_i + t_{ij}]$$

где E_j – срок наступления события j, t_{ij} – длительность операции, лежащей между событиями i и j. Расчет самых ранних возможных сроков наступления событий в сети называется просчетом "вперед". Однако этого не достаточно, необходимо так же знать самые поздние допустимые сроки наступления события в сети.

$$L_j = \min[L_i - t_{ij}]$$

Такой просчет является просчетом "назад". Расчет резерва времени R_i показывает, насколько можно задержаться при выполнении операции оканчивающейся i-ым событием.

$$R_i = L_i - E_i$$

Критический путь образует последовательность операций, объединяющая события с наименьшими резервами времени. Критический путь - самая длинная последовательность операций в сети, и её длительность - это срок завершения проекта в целом. Дан-

ный метод используют такие гранды как Microsoft Project, Oracle Primavera, Gantt Project и другими.

1.1.3. MPM

MPM (англ. Metra Potential Method) – так же как и CPM является сетевой технологией, разработана в 1958 г. во Франции. Проект представляется как обычно в форме сетевого графика, на котором отображены зависимости между операциями. Длительность операций рассчитывается по определенным правилам с учетом имеющегося опыта. При использовании этого метода важным является связь каждой операции с непосредственно ей предшествующими. Поэтому вводится начальная операция "Start", длительность которой равна 0. Введем так же обозначения $ES(B)$ – самый ранний срок начала последующей работы B относительно момента начала предшествующей работы A, $LS(B)$ – самый поздний срок начала последующей работы B относительно момента начала предшествующей работы A. К примеру, если работа B должна начаться одновременно с работой A, то получим:

$$ES(B) = LS(B) = 0$$

Данная технология также как и предыдущая анализирует критические операции. В сетевом графике разрешены обратные связи. Для расчета самых ранних сроков начала операций используется матрица отображения временных взаимоотношений между операциями. Для этого вводят вспомогательные величины $ES(j)^+$ и $ES(j)^-$. Первый параметр учитывает положительные отношения между операциями в проекте. Его значение рассчитывается из составленной ранее матрицы. Для каждой следующей работы в процессе перемещения по столбцам слева направо число $ES(j)^+$ лежащее на пересечении этого столбца со строкой, суммируется со значением $ES(j)^+$ для этой строки. Если в столбце на пересечении с разными строками располагается несколько численных значений, то выбирается максимальная сумма. Для первой операции $ES(j)^+$ принимается равным нулю.

Далее определяется параметр $ES(j)^-$, учитывающий только отрицательные отношения между работами. Процесс вычисления в точности совпадает с $ES(j)^+$, но для каждой операции учитываются только отрицательные значения. Самые ранние сроки окончания работ вычисляются по формуле

$$ES(i)^k = ES(i)^- + t_i$$

а самые поздние

$$LS(i)^k = LS(i)^- + t_i.$$

Резервы времени для конкретных работ можно рассчитать по формуле

$$R_i = LS(i)^k - ES(i)^k$$

. Работы, для которых резервы времени равны нулю, считаются критическими. Задержки с их выполнением вызовут удлинение срока завершения проекта. ПО, реализующее данный метод, обнаружить не удалось.

1.1.4. PERT

PERT (англ. Project Evaluation and Review Technique) – разработана по заказу ВМФ Соединенных Штатов Америки. Использует CPM и позволяет узнать вероятность своевременной реализации проекта с применением стохастических методов. Предоставляет возможность статистической оценки времени выполнения операций и вероятности своевременной реализации каждого этапа в проекте. В процессе планирования выделяются события и операции. Событие - наступление четко определенного состояния в проекте. Событие обозначает окончание и /или начало одной или нескольких операций.

Метод осуществляет:

1. Составление сетевого графика проекта;
2. Определение длительности каждой операции (наиболее вероятной, оптимистической и пессимистической);
3. Расчет мат. ожидания и средне-квадратичное отклонение длительности выполнения операций;
4. Определение критического пути.

Длительность выполнения операции - случайная переменная имеющая β-распределение. Самая оптимистичная длительность выполнения операции обозначается a_n, самая пессимистичная – b_n и наиболее вероятная – m_n. Как правило эти значения берет из головы руководитель проекта.

Расчет ожидаемой длительности выполнения каждой операции n, входящей в состав проекта, выполняется по формуле

$$t_{en} = \frac{a_n + 4m_n + b_n}{6}$$

Параметр, определяющий вероятное отклонение от ожидаемого значения и называемый стандартным отклонением δ_n, рассчитывается для каждой операции по формуле

$$\delta_n = \frac{b_n - a_n}{6}$$

СКО равно 1/6 разности между крайними оценками длительности выполнения работы и считается мерой неопределенности выполнения этой работы за ожидаемое время. Критическим путем будет путь с наиболее длительной реализацией.

По известной ожидаемой длительности реализации проекта и её СКО $\delta T_e = \sqrt{\sum \delta_n^2}$ можно рассчитать вероятность завершения проекта к любому произвольному сроку:

$$z = \frac{T_H - T_e}{\delta T_e}$$

, где z – имеет нормальное распределение, T_H – нормативная длительность, T_e – ожидаемая длительность. Полученное z определит вероятность завершения всех операций в проекте в течении заданного времени.

Метод используется в Microsoft Visio, Project, Orale Primavera, Gantt Project и других ПП.

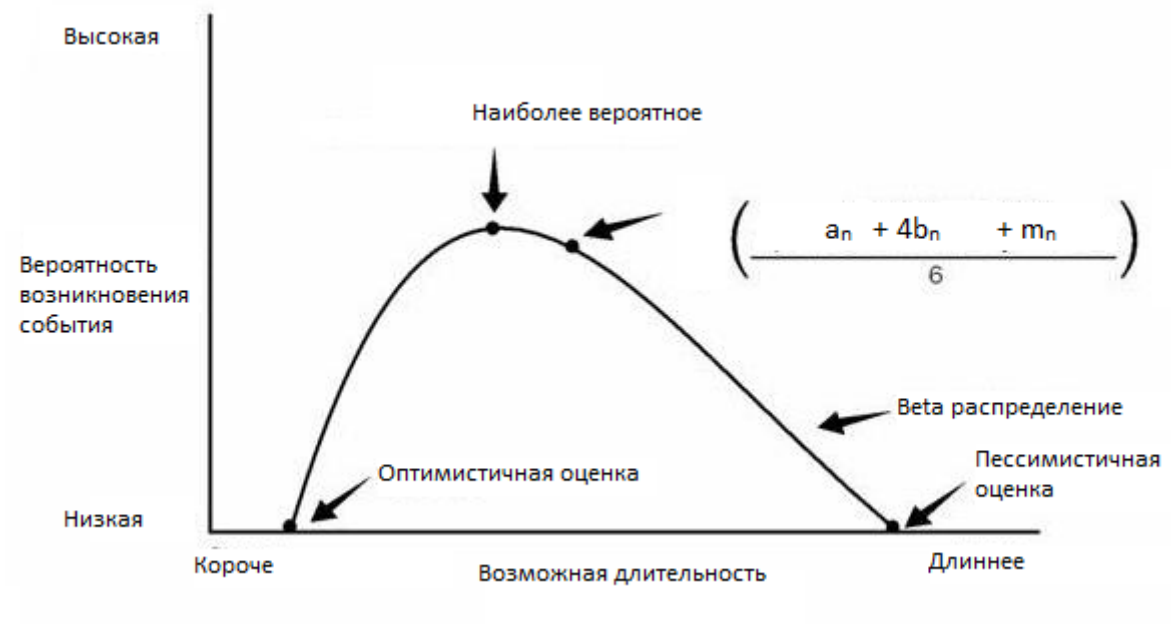

Рис. 2: Ожидаемая длительность операции n

1.1.5. GERT

В отличии от ранее описанных методов, Graphic Evaluation and Review Technique (GERT) позволяет использовать стохастические сети. Эти сети сложнее детерминированных, но с их помощью можно описывать различные варианты зависимостей между событиями в одной и той же сети, а так же выбирать пути развития проекта, отличающиеся от определенных заранее. Технологии, основанные на стохастических сетях, вводят вероятностные типы событий в форме логических объединений операций "или", позволяющие рассматривать альтернативные решения.

Особое внимание следует уделить в проекте альтернативным операциям. Основными этапами являются:

1. Описание стохастической сети;
2. Сбор числовых данных, характеризующих каждую дугу сети;
3. Минимизация построенной стохастической сети;
4. Преобразование сети к форме, которая позволит рассчитать его длительность и вероятность реализации.

Термин "событие" понимается шире чем в детерминированных сетях. Событие может быть не только детерминированными но и стохастическими. Реализованы такие операнды как "И","ИЛИ" и "исключающее ИЛИ". В первом случае событие наступает только тогда, когда все предыдущие события свершились. Во втором - хотя бы одно из предшествующих событий. В третьем - событие произойдет если закончится одна и только одна из взаимоисключающих предшествующих операций. Очевидно, что детерминированные входы и выходы событий имеют форму "И". Можно выделить начальные и конечные события в проекте, начальное событие может быть стохастическим, но конечное всегда детерминировано. При использовании обратных связей или петель всегда указывается количество повторений (счетчик).

Граф проекта определяется как упорядоченная пара $G =< W, T >$ где W – конечное множество вершин w_i, которая определяет событие (состояние), T – конечное множество перемещений t_{ij}. Перемещения определены на конечном множестве упорядоченных пар $< w_i, w_j >$. U_{ij} – функциональные зависимости между вершинами w_i и w_j, называемы дугами. При $i = j$ имеет место петля. S – множество упорядоченных пар $u_{ij}, u_{jk}, u_{kl}, u_{li}$ называется контуром. Минимизация построенной сети заключает-

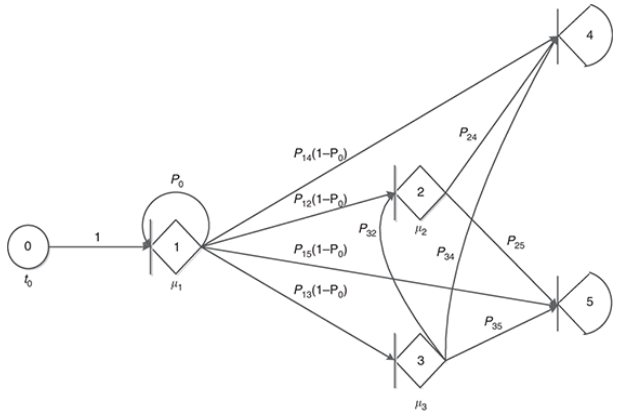

Рис. 3: Пример стохастической сети

ся в упрощении сети с применением различных методов, вплоть до получения менее сложной замещающей сети (либо функции), которая однозначно описывает оригинальную сеть. Каждая дуга (3) представлена вектором $[p_i, t_i]$, где p_i – вероятность реализации i-й дуги при условии, что событие, которому соответствует исходящий узел, свершилось, а t_i – длительность выполнения операции, соответствующее i-й дуге. Основные методы сокращения сети представлены в [14].

На основе рассчитанных вероятностей и длительностей выполнения работ можно утверждать, что реализация проекта продлится t единиц времени с вероятностью p.

1.2. Управление трудовыми ресурсами

Помимо управления временем CPM можно использовать для оптимизации использования ограниченных ресурсов, в том числе и трудовых. Для этого составляется график трудовых ресурсов (ГИР). Основными этапами являются:

1. создание сетевого графика проекта с данными о ресурсах;
2. составление ГИР для самых ранних возможных сроков;
3. составление ГИР для самых поздних возможных сроков;
4. Приведение потребностей ресурсов в соответствии с возможностями их использования.

На рисунке (4) изображен график использования двух ресурсов для семи операций в проекте. Для каждой операции определены самое раннее время начала и самое позднее время окончания(выделено темным фоном). Крестиком обозначены дни простоя.

ГИР с учетом самых ранних и самых поздних возможных сроков показывают уровни потребности в ресурсах в процессе выполнения проектов. При оптимальном

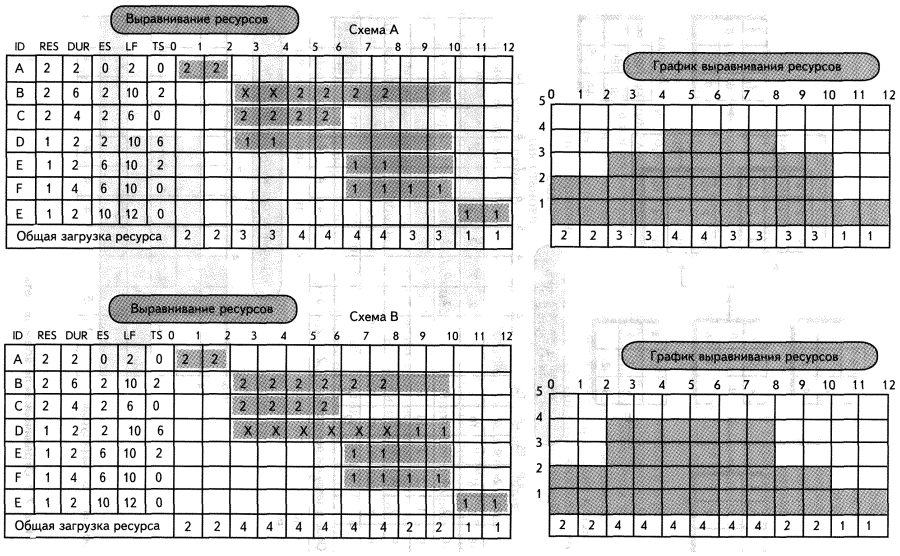

Рис. 4: Графики использования ресурсов и вариант оптимизации загрузки

использовании ресурсов можно минимизировать предельные значения, путём смещения операций или событий в пределах резервного времени.

Следует отметить, рассмотренные методы уже относятся к классическим и для решения реальных задач сетевого планирования применяются методы целочисленного, линейного и нелинейного программирования, которые обычно дают только алгоритмические решения с помощью итеративных вычислительных процедур. Такие решения могут потребовать больших затрат вычислительных ресурсов. Методы идемпотентной алгебры дают решения в явном виде в компактной векторной форме, что гарантирует низкую вычислительную сложность решений.

Вычислительные процедуры способны искать только частные решения или устанавливать, что решений нет. В то время как для многих задач методы идемпотентной алгебры позволяют найти общее (полное) решение задачи, которое в явном виде описывает все множество решений задачи.

Различные алгоритмические решения, как правило, всякий раз требуют разработки новых программных средств. Решения различных задач на основе методов идемпотентной алгебры используют один и тот же математический и вычислительный аппарат (матричные и векторные операции, решение уравнений и неравенств, нахождение собственных чисел и векторов и т.п.), что делает возможным применения одних и тех же унифицированных программных средств для решения разных задач.

Распараллеливание итеративных вычислительных схем обычно затруднено. Получение решения на основе идемпотентной алгебры требует выполнения простых операций (например, умножение матриц), для которых существуют различные эффективные схемы параллельных вычислений.

2. Планирование проектов

Задачи из области планирования проектов обычно представляются в виде задач оптимизации. Такие проблемы возникают на производстве при попытках создать план проекта, который минимизирует максимальное отклонение между временем завершения всех задач в проекте, при различных условиях, которые накладываются на порядок выполнения задач. Рассмотрим проект, состоящий из n операций, которые выполняются при ограничениях типа старт-финиш и старт-старт. Ограничение старт-финиш устанавливает границу для минимального времени задержки между началом одной задачи и окончанием другой. Предположим, что каждая задача немедленно заканчивается, как только указанные ограничения оказываются выполненными. Ограничение старт-старт определяет минимальную задержку между началом двух операций. Одной из задач является нахождение такого плана проекта, который обеспечивает одно общее время завершения для всех операций, не нарушая условий на порядок их выполнения.

Для каждой задачи в проекте $i = 1, \ldots, n$, обозначим через x_i – время начала, y_i – время завершения. Пусть a_{ij} – будет минимально возможная задержка между началом задач $j = 1, \ldots, n$ и завершением i. Ограничения типа "старт-финиш" записываются в виде неравенств:

$$x_j + a_{ij} \leq y_i, \qquad j = 1, \ldots, n. \tag{1}$$

при этом хотя бы одно неравенство должно выполняться как равенство. Заметим, что мы подразумеваем $a_{ii} \geq 0$ для любого i. Если для некоторого j значение c_{ij} не задано, то полагаем $a_{ij} = -\infty$. Теперь мы объединим все эти отношения в одно равенство вида

$$\max_{1 \leq j \leq n} (x_j + a_{ij}) = y_i, \qquad , j = 1, \ldots, n. \tag{2}$$

Кроме того, пусть c_{ij} будет минимально возможной задержкой между началом задач j и i. Будем снова считать, что $c_{ij} = -\infty$, если не определена задержка между j и i. Запись ограничения старт-старт выглядит так

$$x_j + c_{ij} \leq x_i, \qquad j = 1, \ldots, n. \tag{3}$$

Далее можно переписать как одно неравенство

$$\max_{1 \leq j \leq n} (x_j + c_{ij}) \leq x_i, \tag{4}$$

Определим целевую функцию, чтобы сформулировать задачу оптимального планирования, как задачу оптимизации. Критерий оптимальности, который показывает на сколько план обеспечивает одно общее время завершения для всех задач, определяется как максимальная разность между временем завершения задач (интервальная полунорма)

$$\max_{1 \leq i \leq n} y_i - \min_{1 \leq i \leq n} y_i = \max_{1 \leq i \leq n} y_i + \max_{1 \leq i \leq n} (-y_i), \tag{5}$$

Теперь мы можем сформулировать задачу оптимизации, в следующем виде

$$\begin{aligned}
&\max_{1\le i\le n}(y_i) + \max_{1\le i\le n}(-y_i) \to \min \\
&\max_{1\le j\le n}(x_i + a_{ij}) = y_i, \\
&\max_{1\le j\le n}(x_i + c_{ij}) \le x_i, \qquad i = 1, \dots, n.
\end{aligned} \tag{6}$$

Далее будет показано, как задача (6) может быть представлена в терминах тропической (идемпотентной) математики и полностью решена в компактной векторной форме.

2.1. Идемпотентное полуполе

Пусть $\mathbb{X}$ – множество, замкнутое относительно двух ассоциативных и коммутативных операций: сложения $\oplus$ и умножения $\otimes$, с нейтральными элементами: нулем $\mathbb{0}$ и единицей $\mathbb{1}$. Сложение идемпотентно, что означает $x \oplus x = x$ для любого $x \in \mathbb{X}$. Умножение дистрибутивно относительно сложения и обратимо, то есть, для любого $x \in \mathbb{X}_+$, где $\mathbb{X}_+ = \mathbb{X}\backslash\{\mathbb{0}\}$ есть обратное число x^{-1}, которое удовлетворяет равенству $x \otimes x^{-1} = \mathbb{1}$. Таким образом $\mathbb{X}_+$ образует группу по умножению, и структура $(\mathbb{X}, \mathbb{0}, \mathbb{1}, \oplus, \otimes)$ является идемпотентным полуполем.

Возведение в целую степень определяется как обычно. Для любого $x \in \mathbb{X}$ и целого $p > 0$, можно записать $x^0 = \mathbb{1}, \mathbb{0}^p = \mathbb{0}, x^p = x^{p-1} \otimes x$, и $x^{-p} = (x^{-1})^p$.

Далее для краткости не будем указывать знак умножения $\otimes$, как и в обычной математике.

Отметим, что идемпотентное сложение порождает частичный порядок $\le$, такой что $x \le y$ только в том случае, если $x \oplus y = y$. Предполагается, что этот частичный порядок можно расширить до полного порядка на всем $\mathbb{X}$. Далее символы отношений и формулировки задач оптимизации рассматриваются в контексте этого порядка.

Например, полуполе $\mathbb{R}_{\max,+}$ имеет нуль $\mathbb{0} = -\infty$ и единицу $\mathbb{1} = 0$. Для любого $x \in \mathbb{R}$ существует обратный элемент x^{-1}, который в обычных обозначениях совпадает с $-x$. Для любого $x, y \in \mathbb{R}$, степень x^y соответствует арифметическому произведению xy. Порядок, порожденный идемпотентным сложением, соответствует естественному линейному порядку на $\mathbb{R}$.

2.2. Алгебра матриц

Теперь рассмотрим матрицы над $\mathbb{X}$ и обозначим множество матриц с m строками и n столбцами, как $\mathbb{X}^{m\times n}$. Матрица, в которой все элементы равны $\mathbb{0}$, называется нулевой. Матрица регулярна по строкам (столбцам), если она не имеет нулевых строк (столбцов). Матрица регулярна, если она регулярна и по строкам и по столбцам.

Для любых матриц подходящего размера $A = (a_{ij}), B = (b_{ij})$ и $C = (c_{ij})$ и скаляра x, матричное сложение и умножение, а также умножение на скаляр определяется как

$$\{A \oplus B\}_{ij} = a_{ij} \oplus b_{ij}, \qquad \{AB\}_{ij} = \bigoplus_k a_{ik} b_{kj}, \qquad \{xA\}_{ij} = x a_{ij} \tag{7}$$

Для любой матрицы A, её транспонированная обозначается A^T. Для любой ненулевой матрицы $A = (a_{ij}) \in \mathbb{X}^{m\times n}$, введем мультипликативно - сопряженную (или просто сопряженную) матрицу $A^- \in \mathbb{X}^{m\times n}$ с элементами $a_{ij}^- = a_{ji}^{-1}$ если $a_{ji} \neq \mathbb{0}$, и $a_{ij}^- = \mathbb{0}$ в противном случае, для любого $j = 1, \ldots, n$ и $i = 1, \ldots, n$.

Рассмотрим множество квадратных матриц $\mathbb{X}^{n\times n}$. Матрица, которая на диагонали имеет $\mathbb{1}$ и вне диагонали – $\mathbb{0}$, называется единичной и обозначается I. Для любой матрицы A, её след равен

$$\mathrm{tr}A = \bigoplus_{i=1}^{n} a_{ii} \tag{8}$$

Как обычно, матрица, которая состоит их одной строки (столбца) называется строчным (столбцевым) вектором. Обозначим множество столбцевых векторов порядка n как $\mathbb{X}^n$.

Вектор, у которого все компоненты равны $\mathbb{0}$, называется нулевым. Вектор считается регулярным, если у него нет нулевых элементов.

Пусть x – регулярный вектор-столбец и A – матрица. Не трудно заметить, что вектор xA регулярен только тогда, когда матрица A регулярна по строкам. Точно так же вектор-строка x^TA регулярен только тогда, когда матрица A регулярна по столбцам.

Как обычно вектор y линейно зависим от векторов $x_1, \ldots, x_n$, если найдутся скаляры $c_1, \ldots, c_n \in \mathbb{X}$ такие, что $y = c_1x_1 \oplus \cdots \oplus c_nx_n$. Вектор y коллинеарен вектору x, если $y = cx$ для какого-нибудь скаляра c.

Для любого ненулевого вектора $x = (x_i) \in \mathbb{X}^n$, его мультипликативно-сопряженный вектор – это строчный вектор $x^- = (x_i^-)$, где $x_i^- = x_i^{-1}$ если $x_i \neq \mathbb{0}$ и $x_i^- = \mathbb{0}$ в противном случае. Нетрудно проверить следующие свойства мультипликативно-сопряженных векторов.

Для любых двух регулярных векторов x и y одинакового размера, покомпонентное неравенство $x \leq y$ подразумевает $y^- \leq x^-$ и наоборот.

Для любого ненулевого вектора выполняется $x^-x = \mathbb{1}$. Кроме того, если вектор регулярен, то $xx^- \geq \mathbb{I}$.

Введём так же понятие собственного значения матрицы в терминах тропической математики. Если существует ненулевой вектор x такой, что выполняется равенство $Ax = \lambda x$, то λ – собственное значение матрицы A, а вектор x – это её собственный вектор. Максимальное собственное число (в смысле порядка на $\mathbb{X}$) называется спектральным радиусом матрицы A, который вычисляется по формуле

$$\lambda = \bigoplus_{i=1}^{m} \mathrm{tr}^{1/m}(A^m) \tag{9}$$

Спектральный радиус любой матрицы $A \in \mathbb{X}^{n\times m}$ имеет экстремальное свойство

$$\min x^-Ax = \lambda$$

.

Введем так же функцию, которая вычисляет у матрицы число

$$\mathrm{Tr}(A) = \bigoplus_{m=1}^{n} \mathrm{tr}A^m$$

При условии что $\mathrm{Tr}(A) \leq \mathbb{1}$, мы определим матрицу

$$A^* = \mathrm{I} \oplus A \oplus \ldots \oplus A^{n-1} \tag{10}$$

при любом $k \geq 0$ получим

$$A^k \leq A^*$$

. Дана матрица $A \in \mathbb{X}^{n \times m}$, рассмотрим задачу нахождения регулярных векторов, которые будут удовлетворять неравенству:

$$Ax \leq x \tag{11}$$

***Лемма* 1.** *Пусть x – решение неравенства (11). Тогда выполняется следующие условия: 1.Если* $\mathrm{Tr(A)} \leq \mathbb{1}$*, тогда $x = A^*u$ для любого регулярного вектора u. 2. Если* $\mathrm{Tr(A)} > \mathbb{1}$*, тогда не существует регулярного решения.*

2.3. Представление задачи планирования в терминах тропической математики

2.3.1. Just-in-time оптимизация проекта

Рассмотрим задачу записанную в форме (6) и заметим, что представление этой задачи в обычных терминах включает в себя только операции сложения и вычисления максимума. Поэтому ее можно представить в терминах полуполя $\mathbb{R}_{\max,+}$. Нахождение минимума целевой функции является just-in-time [8] оптимизацией проекта в терминах управления проектами.

В начале мы представим ограничения, как скалярные равенства и неравенства

$$\begin{aligned} &\bigoplus_{j=1}^{n} a_{ij} x_j = y_i, \qquad j = 1, \ldots, n \\ &\bigoplus_{j=1}^{n} c_{ij} x_j \leq x_i \qquad i = 1, \ldots, n \end{aligned} \tag{12}$$

Если использовать матрично-векторные обозначения

$$A = (a_{ij}) \qquad C = (c_{ij}) \qquad x = (x_i) \qquad y = (y_i)$$

то скалярные ограничения примут форму

$$Ax = y, \qquad Cx \leq x.$$

Кроме того, мы перепишем целевую функцию (5). При условии, что в $\mathbb{1} = (0, \ldots, 0)^T$, целевая функция будет выглядеть как

$$\bigoplus_{i=1}^{n} y_i \bigoplus_{j=1}^{n} y_j^{-1} = \mathbb{1}^T y y^- \mathbb{1} \tag{13}$$

Наконец, объединяя целевую функцию с ограничениями получим задачу

$$\begin{aligned} &\mathbb{1}^T y y^- \mathbb{1} \to \min \\ &Ax = y \\ &Cx \le x \end{aligned} \tag{14}$$

Решение задачи получено в [4] в следующем виде.

Теорема 2. *Предположим, что A регулярна и C матрица с $\mathrm{Tr}(C) \le \mathbb{1}$. Тогда минимум целевой функции равен $\Delta = (AC^*(\mathbb{1}^T AC^*)^-)^- \mathbb{1}$ и достигается при*

$$x = \alpha C^*(\mathbb{1}AC^*)^-, \qquad y = \alpha AC^*(\mathbb{1}^T AC^*)^-, \qquad \alpha > 0.$$

Далее рассмотрим задачу, в которой нет ограничений типа старт-старт, то есть

$$\begin{aligned} &\mathbb{1}^T y y^- \mathbb{1} \to \min \\ &Ax = y. \end{aligned} \tag{15}$$

В таком случае минимум целевой функции достигается при регулярной матрице A и $\Delta = (A(\mathbb{1}^T A)^-)^- \mathbb{1}$, и при $x = \alpha(\mathbb{1}^T A)^-, y = \alpha A(\mathbb{1}^T A)^-$

2.3.2. Максимизация разброса времени завершения операций

Чтобы упростить последующие записи, мы введём, для любого вектора $x \in \mathbb{X}^n$ и матрицы $A \in \mathbb{X}^{m\times n}$, идемпотентные аналоги векторной и матричной нормы

$$\|x\| = \bigoplus_{i=1}^{n} x_i, \qquad \|A\| = \bigoplus_{i=1}^{m} \bigoplus_{j=1}^{n} x_{ij}$$

Теперь мы можем записать

$$\|x\| = \mathbb{1}^T x, \qquad \|A\| = \mathbb{1}^T A \mathbb{1}$$

а так же для любых векторов x и y одинакового размера, выполняется равенство $\|xy^T\| = \|x\|\|y\|$.

Рассмотрим задачу планирования производства, когда главной целью является определение времени начала каждой операции(задачи) так, чтобы максимум разности между временем завершения задач был максимальным. Такие задачи встречаются при составлении расписания движения воздушных судов в аэропорту, где ресурсом будет взлетно - посадочная полоса, и в других задачах, когда ресурса крайне мало и одновременное использование ограниченного ресурса невозможно. Будем использовать те же обозначения, что и в предыдущей задаче.

Рассмотрим задачу

$$\begin{aligned} &\max_{1\le i\le n}(y_i) + \min_{1\le i\le n}(-y_i) \to \max, \\ &\max_{1\le j\le n}(x_i + a_{ij}) = y_i, \\ &\max_{1\le j\le n}(x_i + c_{ij}) \le x_i \end{aligned} \tag{16}$$

Представим задачу в терминах полуполя $\mathbb{R}_{\max,+}$.При условии, что ограничения на старт-финиш отсутствуют

$$\begin{aligned} &\bigoplus_{i=1}^{m} y_i \bigoplus_{i=1}^{n} y_i^{-1} \to \max \\ &\bigoplus_{j=1}^{n} a_{ij} x_j = y_i, \end{aligned} \tag{17}$$

В матрично-векторной форме

$$\begin{aligned} &\|y\|\|y^-\| \to \max \\ &Ax = y \end{aligned} \tag{18}$$

Лемма 3. *Предположим, что матрица A регулярна по столбцам. Определим число $\Delta = \|AA^-\|$. Тогда максимум целевой функции (18) равен Δ и будет достигаться только при векторе $x = (x_i)$ с координатами*

$$x_k = \alpha\|a_k^-\|, \qquad x_j \le \alpha a_{sj}^{-1}, \qquad j \ne k,$$

для любого $\alpha > 0$ и индексов k и s, вычисляемые по формулам

$$k = \arg\max_{1\le i\le n} \|a_i\|\|a_i^-\|$$

$$s = \arg\max_{1\le i\le n} a_{ik}^{-1}$$

Рассмотрим задачу

$$\begin{aligned} &\|x\|\|x^-\| \to \max \\ &Cx \le x \end{aligned} \tag{19}$$

Лемма 4. *Предположим, что C – неразложимая матрица с $\mathrm{Tr}(C) \le \mathbb{1}$. Определим число $\Delta = \|C^*(C^*)^-\|$, тогда задача (19) достигает максимума равного Δ, тогда и только тогда, когда $x = C^*u$, где $u = (u_i)$ – вектор с компонентами*

$$u_k = \alpha\|(c_k^*)^-\|,$$

$$u_j = \alpha(c_{sj}^*)^{-1}, j \ne k,$$

для любого $\alpha > 0$ и индексов k и s, вычисляемых по формулам $k = \arg\max_{1\le i\le n} \|c_i^*\|\|(c_i^*)^-\|$ и $s = \arg\max_{1\le i\le n}(c_{ik}^*)^{-1}$

Теперь рассмотрим задачу, представленную в (16). В матрично-векторной форме она будет выглядеть как

$$\begin{aligned} &\|y\|\|y^-\| \to \max \\ &Cx \le x \\ &Ax = y \end{aligned} \tag{20}$$

Лемма 5. *Предположим, что A регулярная матрица по строкам и C матрица с $\mathrm{Tr}(C) \le \mathbb{1}$ такая, что все столбцы матрицы $D = AC^*$ регулярны. Определим число*

$\Delta = \|DD^-\|$, тогда задача (20) достигает максимума равного Δ, тогда и только тогда, когда $x = C^*u$, где $u = (u_i)$ – вектор с компонентами

$$u_k = \alpha\|d_k^-\|,$$

$$u_j = \alpha d_{sj}^{-1}, j \neq k,$$

для любого $\alpha > 0$ и индексов k и s, вычисляемые по формулам

$$\begin{aligned} k &= \arg\max_{1\le i\le n} \|d_i\|\|d_i^-\| \\ s &= \arg\max_{1\le i\le n} d_{ik}^{-1} \end{aligned} \tag{21}$$

Заметим, что матрица $D = AC^*$ регулярна по столбцам, только когда матрица A регулярна, а матрица C не приводима.

2.3.3. Минимизация максимального времени цикла операций

Рассмотрим теперь задачу планирования производства, когда нужно минимизировать максимальное общее время(время цикла) всех задач в проекте (иногда такую задачу называют flowtime-оптимизация. Подробно данный критерий рассматривается в [1]). В такой задаче помимо ограничений старт-старт и старт-финиш присутствуют ограничения ранний-старт и поздний-финиш. Для каждой операции, ограничения ранний старт и поздний финиш задают соответственно самое ранее по возможности время начала операции, и самое позднее по возможности время завершения. Пусть g_i – нижняя границей время начала операции, и пусть b_{ij} – минимально возможная задержка между началом задач j и i. Тогда время начала каждой операции должен удовлетворять неравенству

$$x_i + b_{ij} \le x_j, \qquad j = 1, \dots, n. \tag{22}$$

Если задержка не указана непосредственно, то она принимается за $-\infty$. Все эти отношения взятые вместе ведут к неравенству вида

$$\max(x_1 + b_{i1}, \dots, x_n + b_{in}) \le x_i \tag{23}$$

Так же, если учитывать ограничение на максимально раннее время старта операции, то операция не может начинать раньше определенного g_i, то есть

$$\max(\max(x_1 + b_{i1}, \dots, x_n + b_{in}), g_i) \le x_i, \qquad j = 1, \dots, n \tag{24}$$

Более того, для любой операции пусть y_i– время завершения a_{ij} – будет минимально возможная задержка между началом задач $j = 1, \dots, n$ и завершением i. Введём так же верхнюю границу времени завершения операции i и обозначим её за h_i. Как и раньше, если задержка не определена, то мы берём её за $-\infty$. Ограничения старт-финиш требуют, чтобы y_i удовлетворял неравенствам

$$x_i + a_{ij} \le y_j, \qquad j = 1, \dots, n. \tag{25}$$

при том хотя бы одно неравенство должно выполняться как равенство.

Совместив эти неравенства и добавив верхнюю границу для завершения операций, получим

$$\begin{aligned} &\max(x_1 + a_{i1}, \ldots, x_n + a_{in}) = y_i, \\ &h_i \geq y_i, , \qquad i = 1, \ldots, n. \end{aligned} \tag{26}$$

Теперь сформулируем проблему планирования как нахождение минимума максимального времени потока по всем операциям проекта. С целевой функцией представленной в виде

$$\max(y_1 - x_1, \ldots, y_n - x_n) \tag{27}$$

мы приходим к задаче оптимизации с ограничениями

$$\begin{aligned} &\max(y_1 - x_1, \ldots, y_n - x_n) \to \min \\ &\max(\max(x_1 + bi1, \ldots, x_n + b_{in}), g_i) \leq x_i \\ &\max(x_1 + a_{i1}, \ldots, x_n + a_{in}) = y_i, \\ &h_i \geq y_i, \qquad i = 1, \ldots, n. \end{aligned} \tag{28}$$

Теперь представим задачу (9) в терминах тропической математики.

$$\begin{aligned} &\oplus_{1 \leq j \leq n} x_i^{-1} y_i \to \min \\ &(b_{i1} x_1 \oplus g_i) \oplus \ldots \oplus (b_{in} x_n \oplus g_i) \leq x_i \\ &a_{i1} x_1 \oplus \ldots \oplus a_{in} x_n = y_i, \\ &h_i \geq y_i, \qquad i = 1, \ldots, n. \end{aligned} \tag{29}$$

Введём матрицы и вектора

$$A = (a_{ij}) \qquad B = (b_{ij}) \qquad g = (g_i) \qquad h = (h_i) \qquad x = (x_i) \qquad y = (y_i)$$

и перепишем скалярное представление задачи в матрично-векторной форме

$$\begin{aligned} &x^- y \to \min \\ &Bx \oplus g \leq x \\ &Ax = y \\ &h \geq y \end{aligned} \tag{30}$$

В [6] представлена теорема, которая позволяет найти общее решение задачи.

Теорема 6. *Пусть x и y – общее решение задачи (30) при условиях, что матрица A имеет ненулевой спектральный радиус, $\Delta = \mathrm{Tr}(B) \oplus h^- A B^* g$ и*

$$\theta = \bigoplus_{k=1}^{n} \bigoplus_{0 \leq i_0 + \ldots + i_k \leq n-k} \mathrm{tr}^{1/k}(B^{i_0}(AB^{i_0} \ldots AB^{i_k})(I \oplus gh^- A)) \tag{31}$$

тогда верны следующие утверждения:

1. Если $\Delta \leq 1$, то θ - минимум в (30), который достигается при $x = S^* u$ и

$y = AS^*u$ где $S = \theta^{-1}A \oplus B$, и u - любой регулярный вектор удовлетворяющий неравенству

$$g \le u \le (h^- AS^*)^-$$

2. Если $\Delta > 1$, то нет регулярного решения.

2.3.4. Минимальное общее время (makespan)

Минимальное общее время выполнения проекта [7] вычисляется как разность между временем начала самой ранней операции проекта и временем завершения самой поздней операции. Если рассматривать множество работ на множестве машин для выполнения (при условии, что каждая машина может выполнять только одну операцию в данный момент времени), то задача представляет собой задачу динамического программирования. При отстранении от условий связанных с ресурсами можно получить описание задачи в терминах тропической алгебры:

$$\max_{1\le i\le n} y_i - \min_{1\le i\le n} x_i = \max_{1\le i\le n} y_i + \max_{1\le i\le n}(-x_i).$$

После подстановки y_i из (2) проблему можно сформулировать

$$\max_{1\le i\le n} \max_{1\le j\le n}(x_j + c_{ij}) + \max_{1\le i\le n}(-x_i) \to \min. \tag{32}$$

В терминах полуполя $\mathbb{R}_{\max,+}$ задача принимает такой вид:

$$\bigoplus_{i=1}^{n} \bigoplus_{j=1}^{n} c_{ij} x_j \bigoplus_{k=1}^{n} x_k^{-1} \to \min.$$

Теперь, введем матрицу $\boldsymbol{C} = (c_{ij})$ и вектор $\boldsymbol{x} = (x_i)$. В этой нотации, задача в векторной форме выглядит как:

$$\text{minimize} \quad \mathbb{1}^T \boldsymbol{C}\boldsymbol{x}\boldsymbol{x}^- \mathbb{1}. \tag{33}$$

Эта проблема рассматривается в [5] в теоремах 3 и 4. Как следствие этих двух теорем имеем дальнейший результат.

Теорема 7. *Пусть $\boldsymbol{C}$ – регулярная по строкам матрица, и*

$$\Delta = \mathbb{1}^T \boldsymbol{C} \mathbb{1} = \|\boldsymbol{C}\|.$$

Тогда, минимум в задаче (33) *равен Δ и достигается при*

$$\alpha\mathbb{1} \le \boldsymbol{x} \le \alpha\|\boldsymbol{C}\|(\mathbb{1}^T\boldsymbol{C})^-, \qquad \alpha > \mathbb{0},$$

или, что эквивалетно, если

$$\boldsymbol{x} = (\boldsymbol{I} \oplus \|\boldsymbol{C}\|^{-1}\mathbb{1}\mathbb{1}^T\boldsymbol{C})\boldsymbol{u}, \qquad \boldsymbol{u} > \boldsymbol{0}.$$

2.3.5. Минимальное отклонение от директивных сроков

Пусть для всех работ заданы директивные сроки завершения их выполнения. Найдем сроки начала работ, при которых время завершения совпадает с директивными сроками. Если такие сроки начала работ определить невозможно, то следует найти приближенные решения задачи, которые являются оптимальными с точки зрения минимального нарушения директивных сроков. Обозначим через $b = (b_1, \ldots, b_n)^T$ вектор директивных сроков завершения работ. Тогда рассматриваемая задача планирования сводится к исследованию в полукольце $\mathbb{R}_{max,+}$ уравнения:

$$Ax = b.$$

Наилучшее приближение к решению уравнения находится как решение задачи

$$(Ax)^- b \oplus Ax(b)^- \to \min \tag{34}$$

Обозначив через $\Delta = \Delta(A, b) = (A(b^- A)^-)^- b$, рассмотрим случай равенства $\Delta = \mathbb{1} = 0$.. Если равенство выполняется, решение уравнения существует [3]. Решение имеет вид $x = (b^- A)^-$ и задает время начала работ, которое обеспечивает выполнение директивных сроков завершения работ.

Если же $\Delta > 0$ то решением будет

$$x_0 = \sqrt{\Delta}(b^- A)^- \qquad x_1 = (b^- A)^- \qquad x_2 = \Delta(b^- A)^-$$

приближенные сроки вычисляются по формуле

$$y_0 = Ax_0 \qquad y_1 = Ax_1 \leq b \qquad y_2 = Ax_2 \geq b$$

отклонение в этом случае от директивных сроков запишется в виде

$$\rho(y_0, b) = \sqrt{\Delta}, \qquad \rho(y_1, b) = \rho(y_2, b) = \Delta$$

Тогда для определения новых сроков завершения работ можно взять, например, любой вектор b такой, что $b \in [y1; y2]$. При этом величина отклонения новых сроков от первоначальных не будет превосходить Δ. Минимум отклонения, равный $\sqrt{\Delta}$, достигается при $b = y_0$.

3. Программная реализация

3.1. Архитектура разрабатываемой системы

Программный продукт разрабатывался как десктопное приложение ориентированное на ОС Windows XP и старше, но разработка на языке Java позволяет использовать этот продукт так же на семействе ОС Linux. Диаграмма деятельности

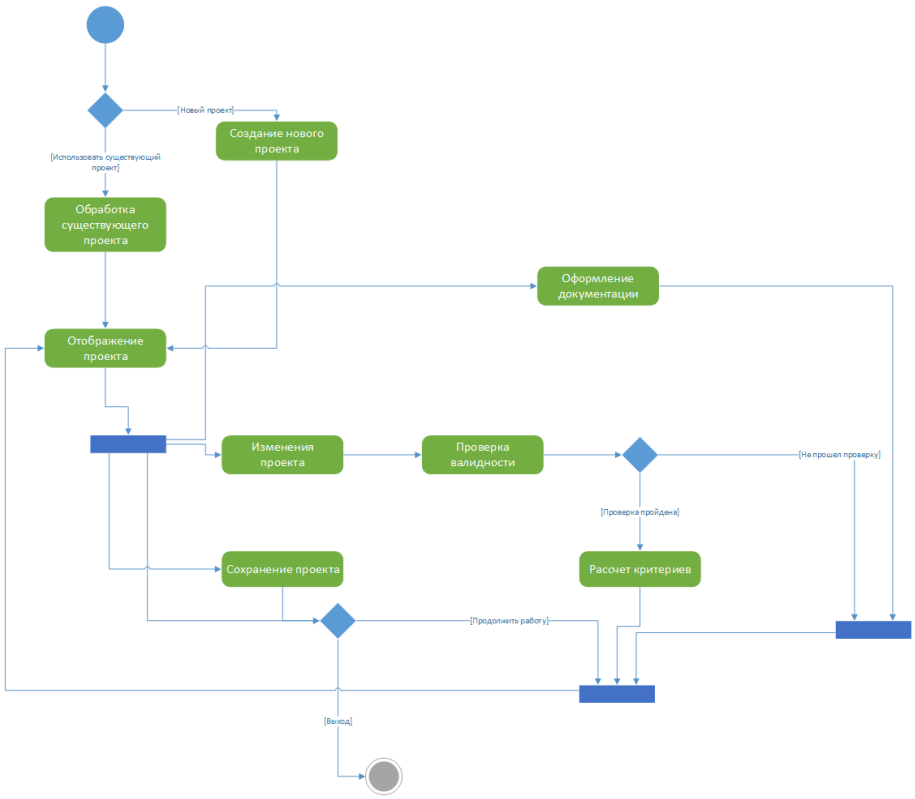

Рис. 5: Диаграмма деятельностей

представлена на рисунке 5.

Структуру разрабатываемой системы можно представить в виде декомпозиции системы на подсистемы, ответственные за определенные задачи. Структурные диаграммы позволяют применить структурный подход к созданию программного продукта (ПП). Все разработанные структурные схемы создаваемого ПП представлены в нотации IDEF0 [9]. Создаваемый программный продукт состоит из подсистем:

1. система обработки проекта;
2. система обработки действий пользователя;
3. система обновления GUI (пользовательского интерфейса);
4. система подготовки документов.

С каждой подсистемой можно точно так же провести декомпозицию. На рисунках 22 – 24 в Приложение 1 представлена декомпозиция структурной схемы "Обработки изменения проекта" на подсистемы и другие.

Диаграмма пакетов разрабатываемого приложения представлена на рисунке (6). Зависимости между пакетами представлены в виде пунктирных направленный линий. Пакет Tropical состоит из двух подпакетов Mathematics и Scheduling. Первый пакет отвечает за простые вычисления, а второй – за нахождение критериев оптимизации проектов. Пакет GUI отвечает за корректное отображение интерфейса приложения, а так же за обеспечение ввода данных пользователем и отображение полученных результатов. Пакет MainPackage представляет собой прослойку между двумя другими, и содержит классы реализующие взаимосвязь между интерфейсом программы и её математической составляющей.

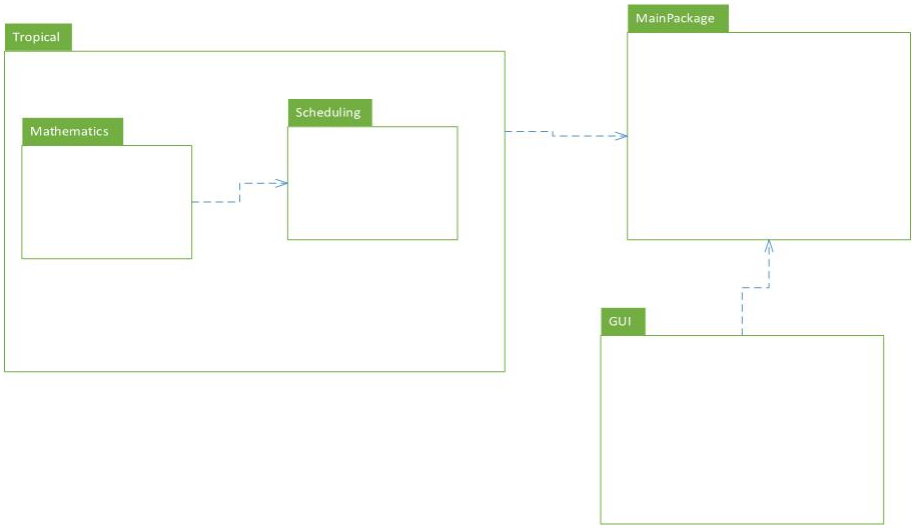

Рис. 6: Диаграмма пакетов UML

Для оптимизации вычислений были созданы несколько классов, отвечающие за различные операции над и между матрицами. В приложении 2 в таблицах представлены методы классов принадлежащих пакету Tropical. На рисунке (7) представлена диаграмма классов пакета Tropical.

Пакет MainPackage состоит из 4х основных классов Job, Project, Resource и Main. Их основные поля и взаимодействие представлено в приложении 3.

3.2. Описание программного и технического обеспечения

3.2.1. Описание программного и технического обеспечения

На текущий момент существует множество операционных систем, основными являются: Microsoft Windows, Ubuntu, Mac OS. В качестве рабочей операционной среды было выбрано семейство Microsoft Windows, а именно Microsoft Windows 7 по следующим причинам:

1. Windows 7 является многозадачной средой, что позволяет пользователю одновременно работать с несколькими приложениями;

2. наличие удобного пользовательского интерфейса, который позволяет вести простой понятный диалог с пользователем и практически не использовать клавиатуру;

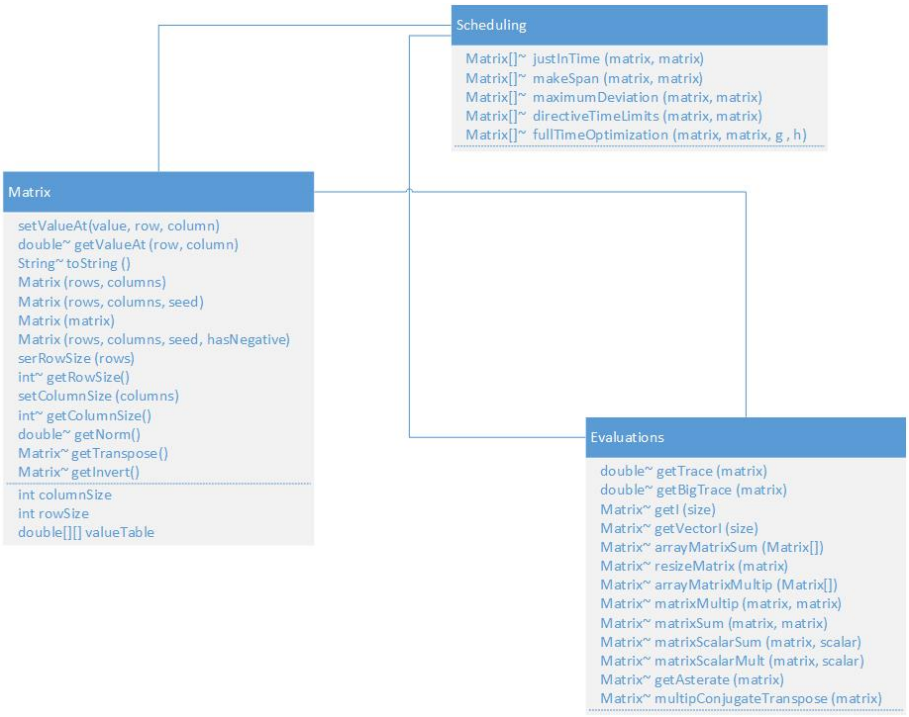

Рис. 7: Взаимодействие основных классов вычислений

3. операционные системы Windows семейства NT обеспечивают достаточную надежность, позволяют использовать все технические средства;

4. наличие огромного количества программного обеспечения, широкое распространение Windows, ,большой объём разработанных программных продуктов для данной системы.

3.2.2. Требования к программному обеспечению

Программное обеспечение должно соответствовать следующим требованиям:

1. программное обеспечение должно обеспечивать продолжительную работу без сбоев и внештатных ситуаций (сбои системного программного обеспечения должны происходить не чаще одного раза в год в режиме круглосуточной эксплуатации);

2. Программное обеспечение должно обеспечивать:

   (a) решение комплексов функциональных задач в режиме реального времени и в интерактивном режиме;

   (b) ведение диалога и обмен информацией между пользователем и машиной;

3.2.3. Выбор языка и средств разработки

В качестве языка программирования был выбрал объектно-ориентированный язык Java [12]. Java-технологии имеют много особенностей, отличающие их от других технологий разработки программного обеспечения. К главным достоинствам языка Java можно отнести:

1. Переносимость;
2. Безопасность;
3. Надежность;
4. Наличие сборщик мусора;
5. Стандартные библиотеки;
6. Самодокументируемый код;
7. Многообразие типов приложений.

Программы, написанные на языке Java, после однократной трансляции в байт- код могут быть исполнены на любой платформе, для которой реализована виртуальная Java-машина. Функционирование программы полностью определяется (и ограничивается) виртуальной Java-машиной. Отсутствуют указатели и другие механизмы для непосредственной работы с физической памятью и прочим аппаратным обеспечением компьютера. В языке Java отсутствуют механизмы, потенциально приводящие к ошибкам: арифметика указателей, неявное преобразование типов с потерей точности и т.п. Присутствует строгий контроль типов, обязательный контроль исключительных ситуаций. Многие логические ошибки обнаруживаются на этапе компиляции. Наличие дополнительных проверок снижает эффективность выполнения Java-программ. Освобождение памяти при работе программы осуществляется автоматически с помощью сборщика мусора, поэтому программировать с использованием динамически распределяемой памяти проще и надежнее. Многие задачи, встречающиеся при разработке программного обеспечения, уже решены в рамках стандартных библиотек [15]. Использование объектно-ориентированного подхода позволяет легко использовать готовые объекты в своих программах. Имеется механизм автоматического генерирования документации на основе комментариев, размещенных в тексте программ. На языке Java возможно реализовать абсолютно разные по способу функционированию и сфере использования программы.

3.2.4. Идемпотентная алгебра

Исследования, проведенные в главе 1, показали, что матриц характеризующих ограничения старт-старт и старт-финиш не достаточно для полного математического описания проекта – не явным образом указываются иерархические взаимосвязи между операциями (окончание каких операций влечет за собой начало других). Для исправления данной ситуации была введена матрица $B \in \mathbb{X}^{\mathbb{n}\times\mathbb{n}}$, принимающая значения либо $\mathbb{0}$ либо $\mathbb{1}$, для каждой операции указывается, после завершения каких операций начинается её выполнение. Данная матрица будет являться идемпотентным аналогом матрицы смежности графа, который описывает порядок выполнения операций. Запишем ограничение финиш-старт

$$By \leq x$$

После подстановки в это неравенство уравнения, описывающего ограничение старт-финиш, получим

$$BAx \leq x$$

Полученное ограничение представляет собой уже известное нам ограничение на старт-старт, а это значит что матрица $C = BA$.

Особое внимание уделено в приложении 4 алгоритму минимизации максимального времени цикла операций (flow-time).

В ходе работы удалось обобщить в некотором смысле критерии оптимальности – для каждого критерия обобщение позволяет вводить все характеристики старт-старт, старт-финиш, самые ранние сроки начала и самые поздние сроки завершения, что не подразумевалось в оригинальных статьях [3-6]. На данный момент можно лишь утверждать лишь о двух свойствах проекта – выполнимости или не выполнимости ограничений, а так же если ограничения выполнимы, то соблюдение ограничений на самый ранний старт операций в проекте.

При условиях, что матрица A имеет ненулевой спектральный радиус, и число

$$\Delta = \mathrm{Tr}(B) \oplus h^{-} AB^{*} g$$

удовлетворяет ограничению $\Delta \leq 1$ то ограничения выполнимы и наоборот. Функция $\mathrm{Tr}(B)$ является определителем матрицы, и позволяет ответить на вопрос имеет ли уравнение тривиальное решение или у него существуют и другие. Матрица B^* позволяет вычислить расстояния(в рассматриваемом случае – время) от каждой вершины(операции в проекте) до всех остальных [2]. Домножив матрицу B^* на матрицу старт-финиш A, у которой на диагонали стоят длительности операций получим матрицу AB^*, которая определяет время между всеми операциями в проекте с учетом длительности самих операций. Домножение этой матрицы слева и справа на самые поздние и самые ранние сроки позволяет определить конфликтуют ли ограничения между собой.

Данное предположение позволяет вводить ограничения на самый ранний старт и самый поздний финиш в критерии оптимальности, которые раннее не учитывали данные временные характеристики. К ним относятся – минимизация (максимизация) разброса времени завершения операций, минимизация общего времени выполнения проекта. Если ограничения описанные выше не выполняются, программный продукт выдаст сообщение о невозможности провести критерии оптимизации при даных условиях и предложит или изменить условия или не принимать во внимания дополнительные ограничения на самое раннее время начала операций и на самое позднее время окончания операций.

3.2.5. Диаграмма Ганта

Данная диаграмма – популярный способ иллюстрации плана, графика работ по какому-либо проекту, изображена на рисунке . Является главным способом визуализации результатов планирования проектов. Диаграмма была создана с помощью библиотеки swiftGanttChart и представлена на рисунке 8 справа.

3.2.6. Сетевой график

Сетевой график представлен на рисунке 8 слева, создан с помощью библиотеки JGraphX. Позволяет пользователю интерактивно работать с проектом – редактировать существующие операции, переопределять их последовательность простым пере-

таскиванием стрелок.

Проверка на петли и циклы проводится при создании операций с помощью алгоритма прохода графа в глубину или с помощью проверки на строгую треугольность матрицы инцидентности сети. Если у проекта несколько стартовых операций или несколько операций окончания, то вводятся фиктивные операции начала (окончания) проекта, объединяющие предшествующие или следующие за всеми операциями проекта.

3.3. Руководство пользователя

Всё взаимодействие пользователя с ПО осуществляется через его интерфейс(8). Для обеспечения простоты и понятности интерфейс разгруппирован на два блока, имеет меню и так же панель инструментов. В центре находятся две вкладки, отвечающие за различные варианты отображение операций в проекте а так же его ресурсов.

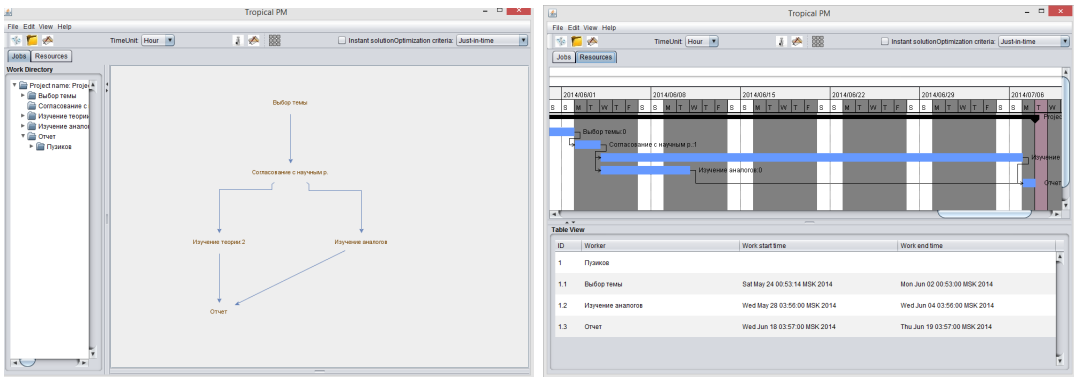

Рис. 8: Основное окно программы

Во вкладке Jobs слева посередине находится граф взаимодействия операций, а слева от него отображено дерево проекта. Вкладка Resources содержит в себе диаграмму Ганта, которая позволяет взглянуть на распределение операций во времени, а так же таблицу ресурсов и выполняемых ими работ.

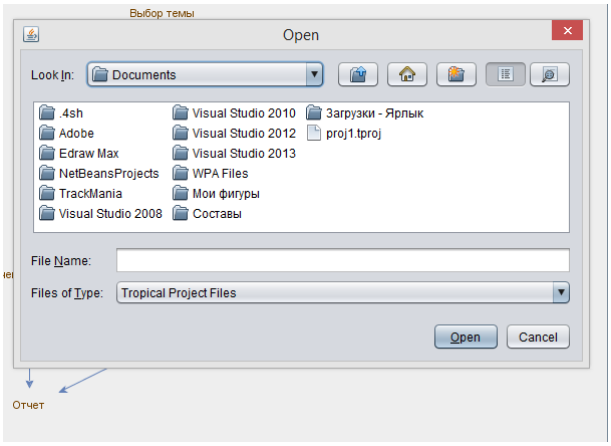

Рис. 9: Выбор уже существующего проекта

При первом запуске программы создаётся пустой проект с пустым множеством ресурсов и операций. Если пользователь хочет загрузить уже существующий проект

ему лишь нужно выбрать соответствующую кнопку на панели инструментов или в меню File - Open Project или нажать комбинацию клавиш (Ctrl+O). После выполнения вышеописанных действий откроется окно выбора файла (9) , выбирать следует файл с расширение .tproj. После этого проект отобразится во всех описанных выше вкладках. В дереве проект а будут представлены все операции и связанные с ними ресурсы. По центру будет расположен детерминированный граф взаимодействия операций. Во вкладке Resources появятся диаграмма Ганта и таблица используемых ресурсов.

Для создания операции пользователю следует выбрать соответствующую кнопку или на панели инструментов, или в меню Edit - Create job, или просто нажать сочетание клавиш (Ctrl+J). После выполнения этих действий появится окно (10) позволяющее задать свойства для новой операции такие как её имя, время начала и окончания, ограничение на эти временных характеристики а так же предыдущие и последующие работы и занятый ресурс, если таковой имеется. Следует заметить, что связи в проекте детерминированны не допускается создание петель и обратных связей. Проверка на ацикличность проводится с помощью поиска в глубину.

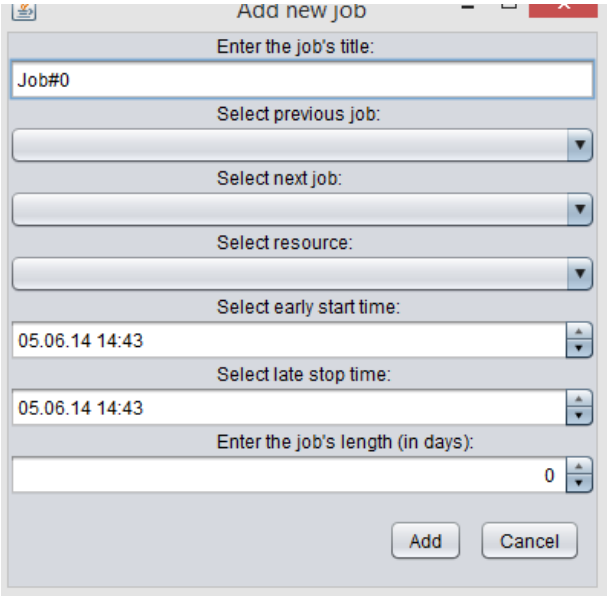

Рис. 10: Создание новой операции

Так же присутствуют ограничения на временные параметры - операция не может начаться позже времени завершения, а ограничения на самое раннее начало и самый позднее окончание так же должны ограничивать временные характеристики, не нарушая естественых ограничений. Длительность операции не может быть больше чем её крайние ограничения.

Если пользователю необходимо отредактировать уже имеющуюся операцию, то ему достаточно выбрать её двойным щелчком в дереве проекта, или на графе взаимодействие операций или в таблице ресурсов во вкладке Resources. После этого откроется окно редактирования операции (11) Интерфейс для создания или редактирования ресурсов выглядит по аналогии с операциями. Для создания ресурса можно либо выбрать на панели инструментов соответствующую кнопку, либо нажать (Ctrl+J), либо выбрать из строки меню Edit - New Resource. Каждой работе может быть назначен только один ресурс, поэтому при создании нового ресурса в выпадающем списке работ будут отсутствовать уже занятые работы. Если нужно назначить ресурсу сразу

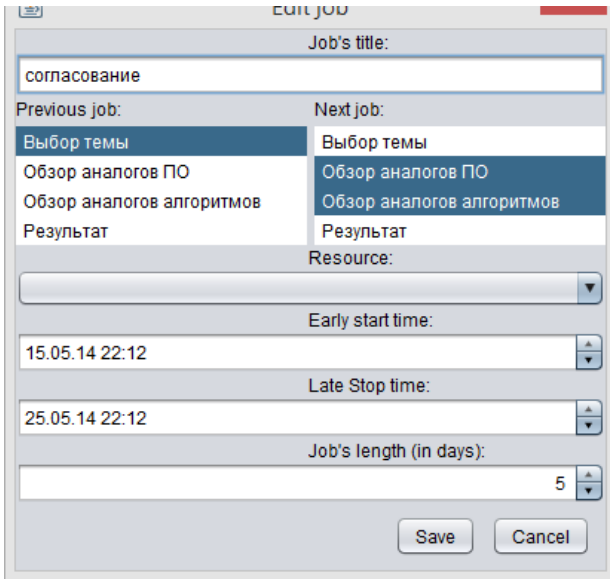

Рис. 11: Редактирование существующей операции

несколько работ, то это легко можно сделать, отредактировав его. Чтобы отредактировать уже существующий ресурс следует произвести двойной щелчок мыши по ресурсу либо в дереве проекта либо в таблице ресурсов во вкладке Resources. Окна для создания и редактирования ресурсов представлены на рисунке (12).

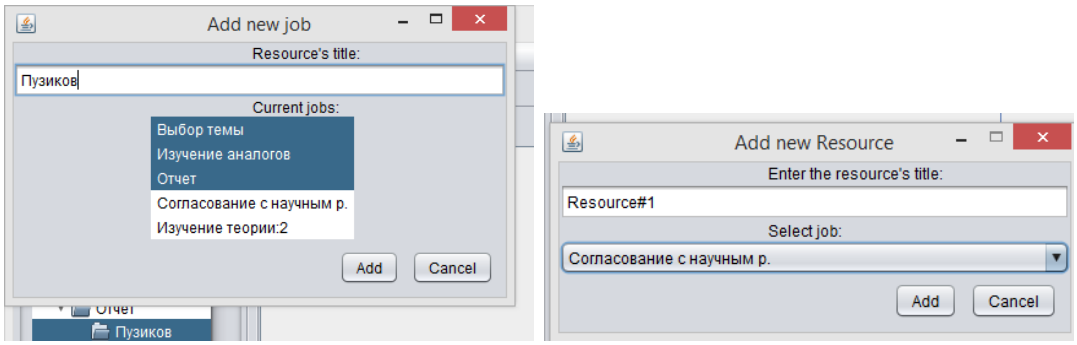

Рис. 12: Создание или редактирование ресурса

Панель инструментов содержит различные полезные для работы компоненты (13). С помощью неё можно задавать точность графика Ганта, критерий оптимизации и условия его вычисления (сразу или после определенного момента), можно создавать новые проекты, операции и ресурсы, а так же сохранять/загружать проекты.

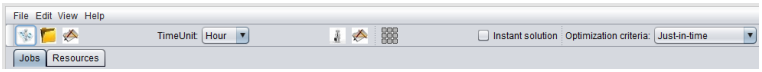

Рис. 13: Панель инструментов

При наведении курсора на любую кнопку на панели инструментов всплывает подсказка с информацией о функции кнопки. Справа находится выпадающий список доступных критериев расчета. Если стоит галочка "Instant solution" то расчет выбранного критерия производится при каждом изменении в проекте.

В центре панели инструментов находится кнопка, нажав которую пользователь может увидеть матрицы ограничений, которые используются в расчета критериев оптимизации. Это матрицы ограничений старт-старт и старт-финиш (рисунок 14) С

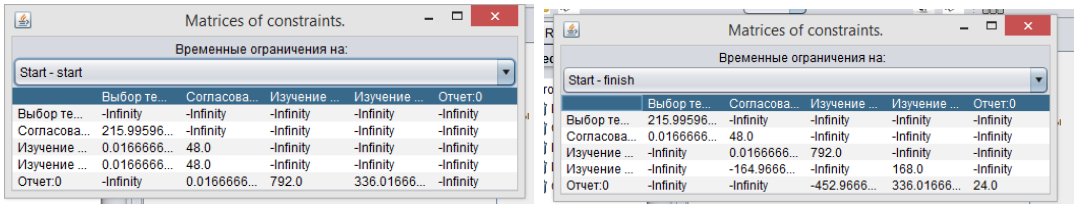

Рис. 14: Матрицы временных ограничений

целью введения новых ограничений или изменения уже существующих можно редактировать любую из ячеек в матрице. Если новое значение является недопустимым, выдаётся предупреждение.

Слева на панели инструментов находится выпадающий список, с помощью которого можно контролировать временную шкалу диаграммы Ганта. Можно выбрать в качестве единичного отрезка год, месяц, неделю или даже день (рисунок 15).

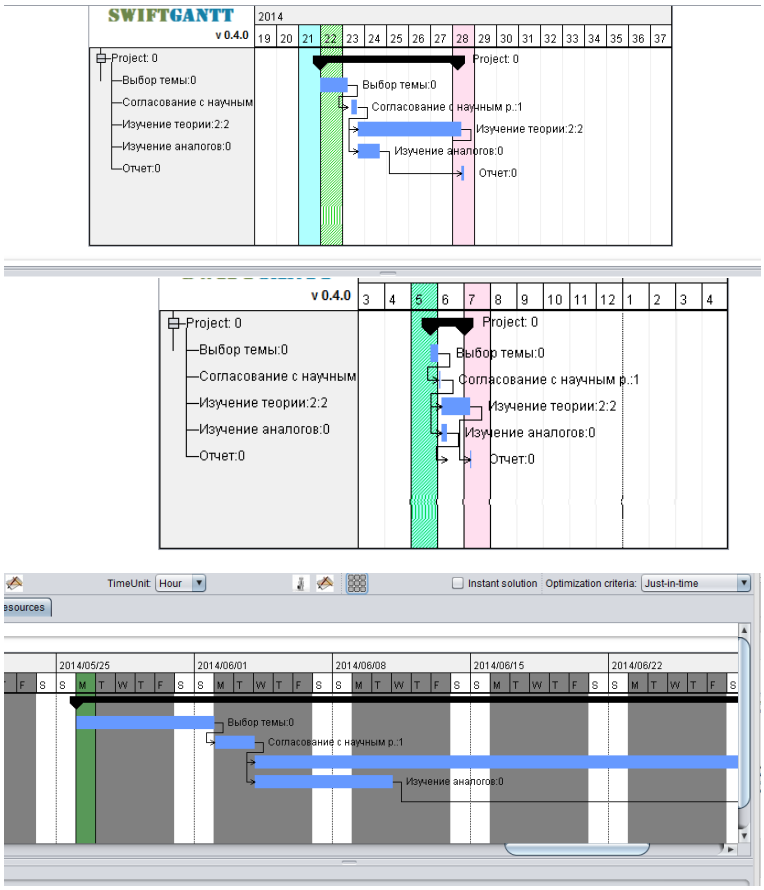

Рис. 15: Различная временная точность диаграмм Ганта

На рисунке 8 справа внизу представлена таблица для отображения ресурсов и закрепленных за ними операций. С помощью этой таблицы можно увидеть временные характеристики операций одного ресурса и исправить различные нестыковки, с помощью двойного щелчка курсором либо по ресурсу либо по операции.

Строка меню (рисунок 16) дублирует все основные функции интерфейса. Так с помощью неё можно изменить язык интерфейса. А так же получение справки.

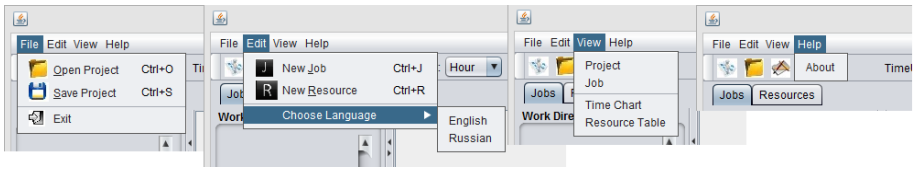

Рис. 16: Строка меню

4. Численные примеры

Рассмотрим проект, состоящий из 5 операций:

1. Выбор темы;
2. Согласование темы;
3. Обзор аналогов ПО;
4. Обзор аналогов алгоритмов;
5. Представление результата;

Для каждой операции введем её длительность: выбор темы длится 4 день, согласование темы – 1 день, обзор аналогов ПО – 5 дней, обзор аналогов алгоритмов –6 дней и представление результата – 1 день.

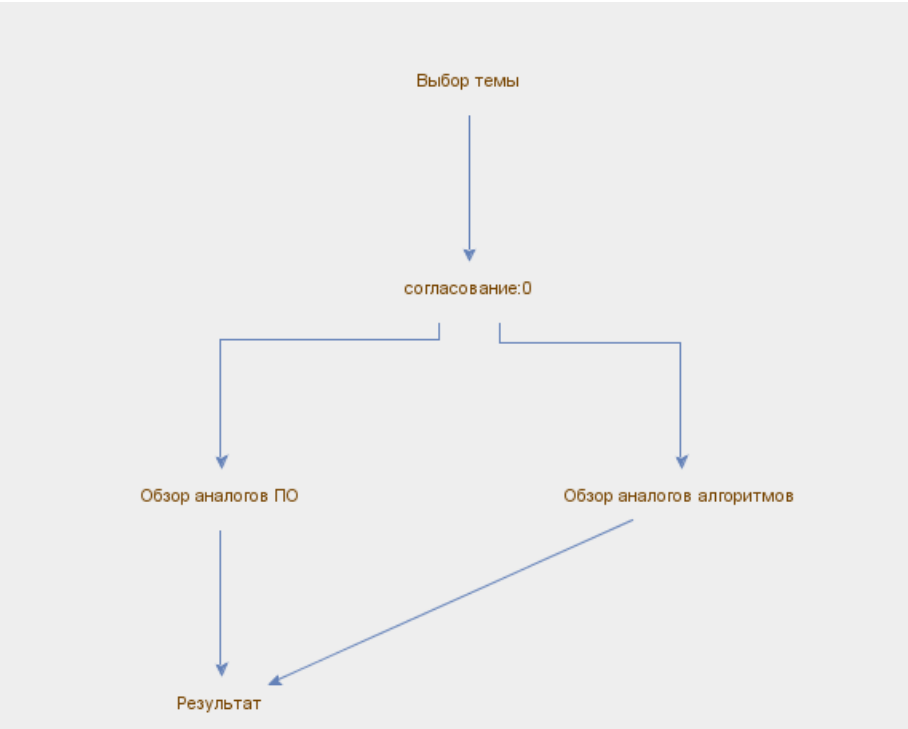

Рис. 17: Сетевой граф проекта

Определим предшествующие операции для всех операций в проекте. У первой операции предшественников нет, для второй операции предшественником является первая, для третьей и четвертой – вторая и для пятой – третья и четвертая. В таком случае можно представить проект в виде графа детерминированных связей между операциями как на рисунке 17. После определения зависимостей между операциями

в графе можно описать его с точки зрения ограничений старт-финиш:

$$A = \begin{pmatrix} 4 & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 5 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & 5 & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & 6 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} & 1 \end{pmatrix}.$$

Следует отметить, что ячейки матрицы инцидентности со значением 0 характеризуют связь между предшествующей и предыдущей операциями.

$$B = \begin{pmatrix} \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ 0 & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 0 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 0 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & 0 & 0 & \mathbb{0} \end{pmatrix}.$$

Значение 0 означает отсутствие промежутка между стартом следующей операции и окончанием предыдущей. Чтобы охарактеризовать наличие промежутка введем ненулевые значения. Матрица инцидентности и ограничения старт-старт

$$B = \begin{pmatrix} \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ 8 & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 10 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 10 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & 10 & 12 & \mathbb{0} \end{pmatrix}. \qquad C = AB = \begin{pmatrix} \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ 12 & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 15 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 15 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & 15 & 18 & \mathbb{0} \end{pmatrix}.$$

Рассмотрим критерий оптимизации описанный в пункте flow-time оптимизации. Для этого для каждой операции следует ввести самые ранние сроки начала, а самые поздние сроки окончания:

$$g = \begin{pmatrix} 0 \\ 10 \\ 19 \\ 19 \\ 29 \end{pmatrix}, \qquad h = \begin{pmatrix} 14 \\ 20 \\ 25 \\ 28 \\ 33 \end{pmatrix}.$$

Заполнив, таблицы представленные на рисунке 14 и выбрав критерий оптимизации full-time, получим следующий результат

$$x = \begin{pmatrix} 0.0 \\ 10.014 \\ 20.014 \\ 20.014 \\ 32.014 \end{pmatrix}, \qquad y = \begin{pmatrix} 4.0 \\ 15.014 \\ 25.014 \\ 26.014 \\ 33.014 \end{pmatrix}.$$

Единицей времени является день, но это изменяемый параметр на панели инструментов. Если принять значение 0.0 за дату 05 мая 21:52:00 2014 года, то получим следующий график:

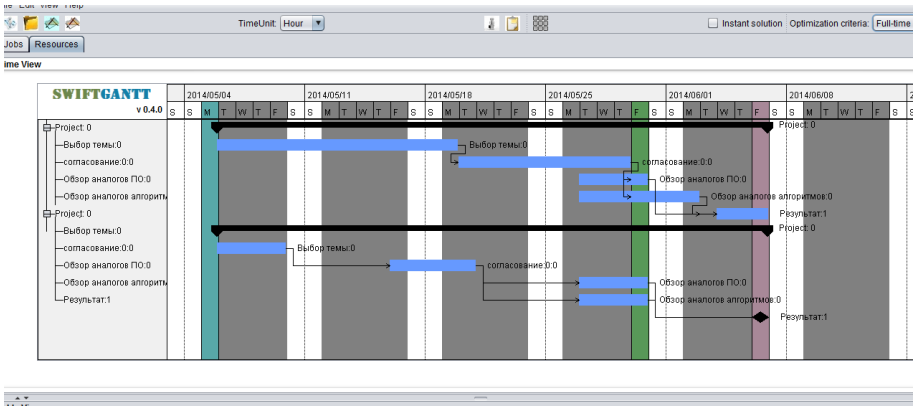

Рис. 18: Минимизация максимального времени по всем операциям в проекте

На рисунке 18 представлен исходный график в котором границы операций определялись их самыми ранними и самыми поздними значениями, и полученный график в результате оптимизации.

Рассмотрим ту же самую задачу с ограничениями но с just-in-time критерием оптимизации. После выбора данного критерия оптимизации из выпадающего списка на панели инструментов, получим следующий результат

$$x = \begin{pmatrix} 0.0 \\ 8.0 \\ 10.0 \\ 12.0 \\ 30.0 \end{pmatrix}, \qquad y = \begin{pmatrix} 4.0 \\ 13.0 \\ 25.0 \\ 24.0 \\ 31.0 \end{pmatrix}.$$

Как и в предыдущем примере примем начальное время за 05 мая 21:52:00 2014 года и получим следующий график:

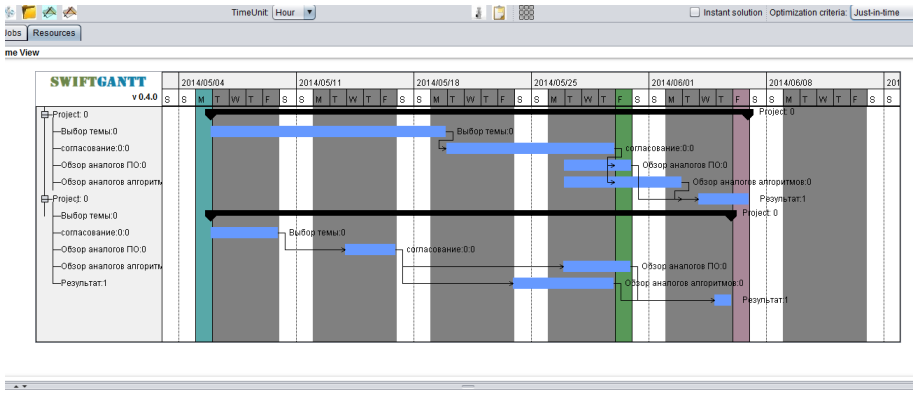

Рис. 19: Окончание всех операций в один срок

Следует отметить, что окончания в один срок всех операций в данном случае невозможно из-за слишком больших ограничений во времени между операциями и зафиксированной последовательностью выполнения.

Минимизация общего времени выполнения проекта осуществляется с путем выбора критерия makespan из выпадающего списка на панели инструментов. При исходных данных получим следующий результат

$$x = \begin{pmatrix} 0.0 \\ 10.0 \\ 19.0 \\ 19.0 \\ 29.0 \end{pmatrix}, \qquad y = \begin{pmatrix} 4.0 \\ 15.0 \\ 24.0 \\ 25.0 \\ 30.0 \end{pmatrix}.$$

Как и в предыдущем примере примем начальное время за 05 мая 21:52:00 2014 года и получим следующий график:

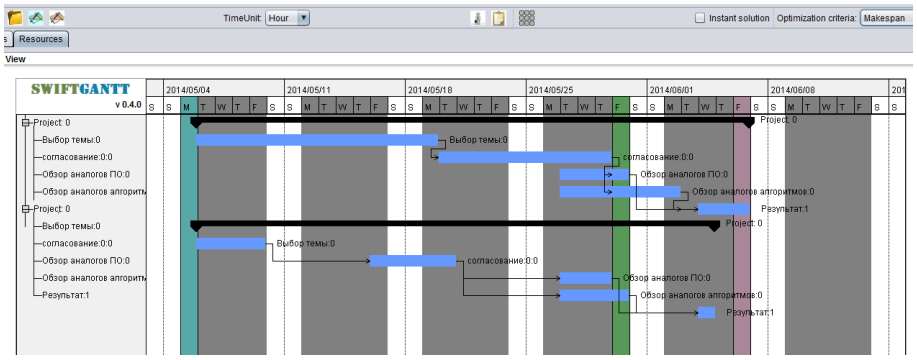

Рис. 20: Минимизация общего времени выполнения проекта

Заключение

В ходе выполнения магистерской диссертации была изучена дисциплина управления проектами и в частности класс задач сетевого планирования.

Изучены существующие методы решения данных задач и способы их программной реализации.

Изучены основы идемпотентной алгебры для решения задач сетевого планирования. Изучены модели и задачи на основе идемпотентной алгебры.

Выявлены методы дисциплины управления проектами, которые пока невозможно представить в терминах идемпотентной алгебры - оптимальное распределение операций среди ресурсов и управление издержками проекта.

Разработаны алгоритмы и программные средства на языке Java SE 1.7, использующие разработанные методы сетевого планирования.

Результат представлен в магистерской диссертации, которая была составлена согласно [11].

Список литературы

[1] Bansal Nikhil. Algorithms for Flow Time Scheduling. — Carnegie Mellon University Pittsburgh, 2003. — P. 86.

[2] Carre B.A. An algebra for network routing problems. — IMA J Appl.Math 7 no. 3, 1971. — P. 273–294.

[3] Krivulin N. *Methods of Idempotent Algebra for Problems in Modeling and Analysis of Complex Systems.* — Saint Petersburg University Press, Saint Petersburg, 2009. — (in Russian).

[4] Krivulin N. Explicit solution of a tropical optimization problem with application to project scheduling in *Mathematical Methods and Optimization Techniques in Engineering.* — arXiv:1303.5457 [math.OC], 2013. — http://arxiv.org/abs/1303.5457.

[5] Krivulin N. *A Tropical Optimization Problem with Application to Project Scheduling with Minimum Makespan.* — arXiv:1403.0268 [math.OC], 2013. — http://arxiv.org/abs/1403.0268.

[6] Krivulin N. *A constrained tropical optimization problem: Complete solution and application example.* — arXiv:1305.1454 [math.OC], 2013. — http://arxiv.org/abs/1305.1454.

[7] Wikipedia. Job shop scheduling — Wikipedia, The Free Encyclopedia. — 2014. — [Online; accessed 1-June-2014]. URL: `http://en.wikipedia.org/w/index.php?title=Job_shop_scheduling&oldid=608552369`.

[8] Wikipedia. Just in time (business) — Wikipedia, The Free Encyclopedia. — 2014. — [Online; accessed 1-June-2014]. URL: `http://en.wikipedia.org/w/index.php?title=Just_in_time_(business)&oldid=609375789`.

[9] Википедия. IDEF — Википедия, свободная энциклопедия. — 2014. — [Online; accessed 1-июнь-2014]. URL: `http://ru.wikipedia.org/?oldid=61646158`.

[10] Воропаев В.И. Управление проектами в современном обществе. — 2005. — P. 4–21. — Управление проектами и программами.

[11] ГОСТ 7.32 – 2001. Отчет о научно-исследовательской работе. — М: Изд-во стандартов, 2004. — P. 19.

[12] Портянкин И. Swing. Эффектные пользовательские интерфейсы. — 2 edition. — издательство Лори, 2011. — P. 608.

[13] Руководство к Своду знаний по управлению проектами. Четвертое издание (Руководство PMBOK) Американский национальный стандарт ANSI/PMI 99-001-2004. — Fourth edition. — Project Management Institute, 2008. — P. 380.

[14] Троицкий М. *Управление проектами/ М. Троицкий, Б.Груча, К. Огонек.* — М., Финансы и статистика, 2011.

[15] Эккель Б. Философия Java. — 4 edition. — СПб.,издательство Питер, 2014. — P. 640.

Приложение 1. IDEF0 диаграмма ПП

Декомпозиция системы с использованием нотации IDEF0.

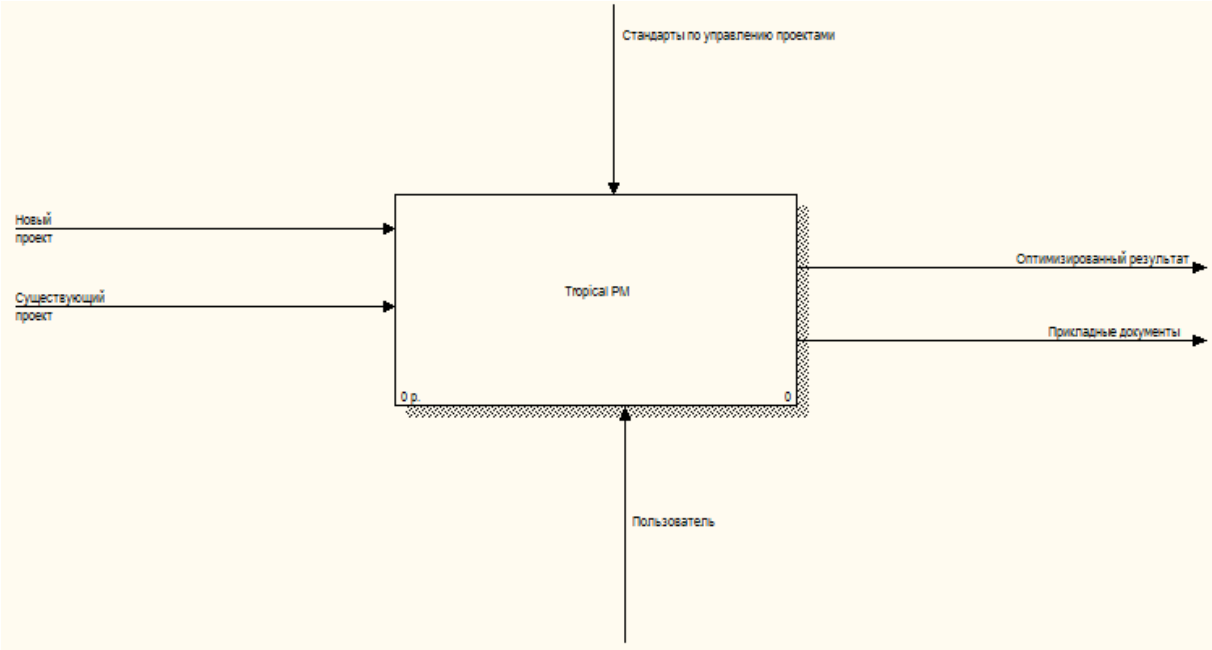

Рис. 21: IDEF0 диаграмма системы

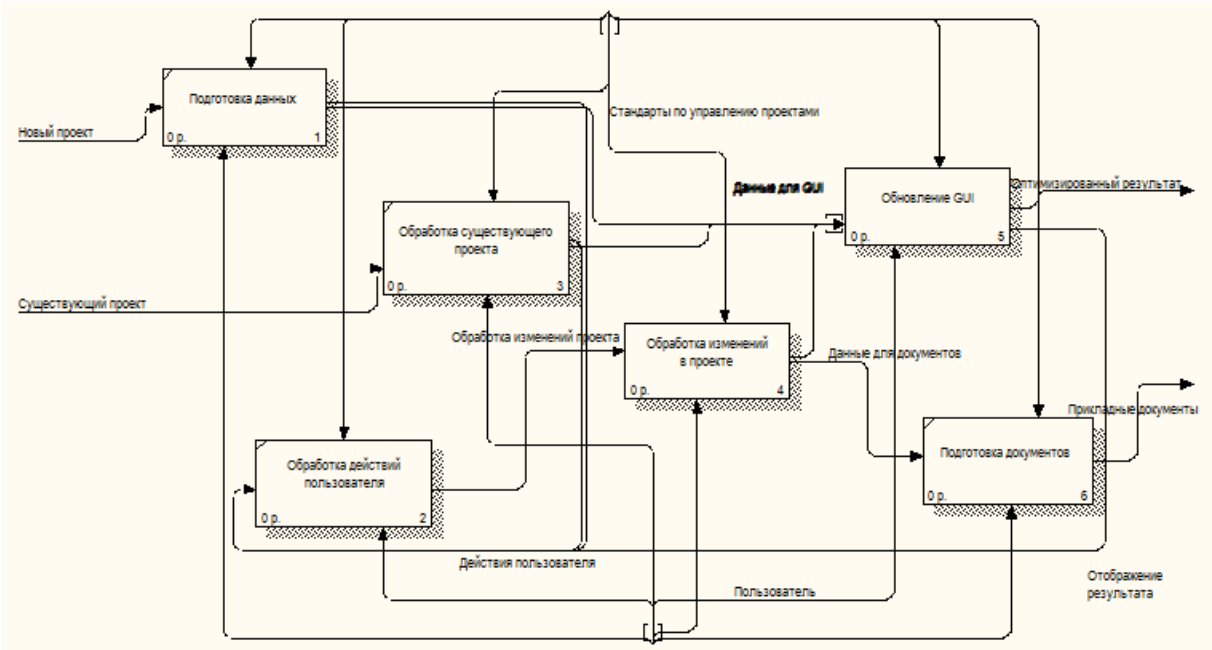

Рис. 22: IDEF0 диаграмма декомпозиции системы

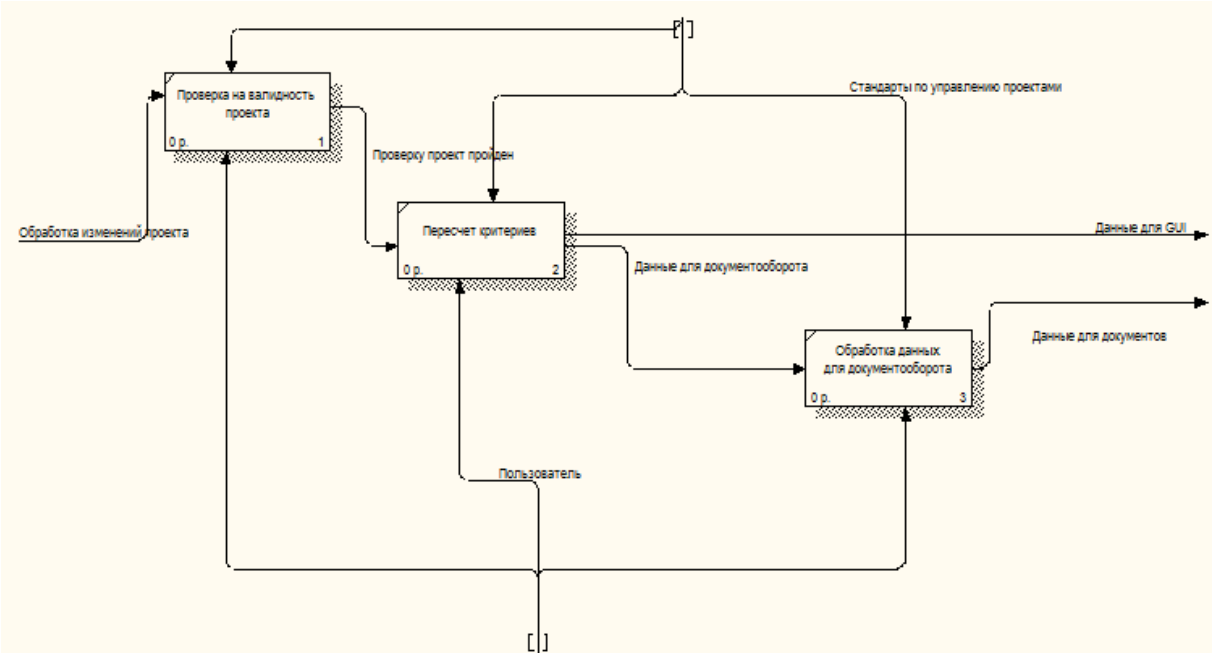

Рис. 23: IDEF0 диаграмма декомпозиции подсистемы обработки изменений

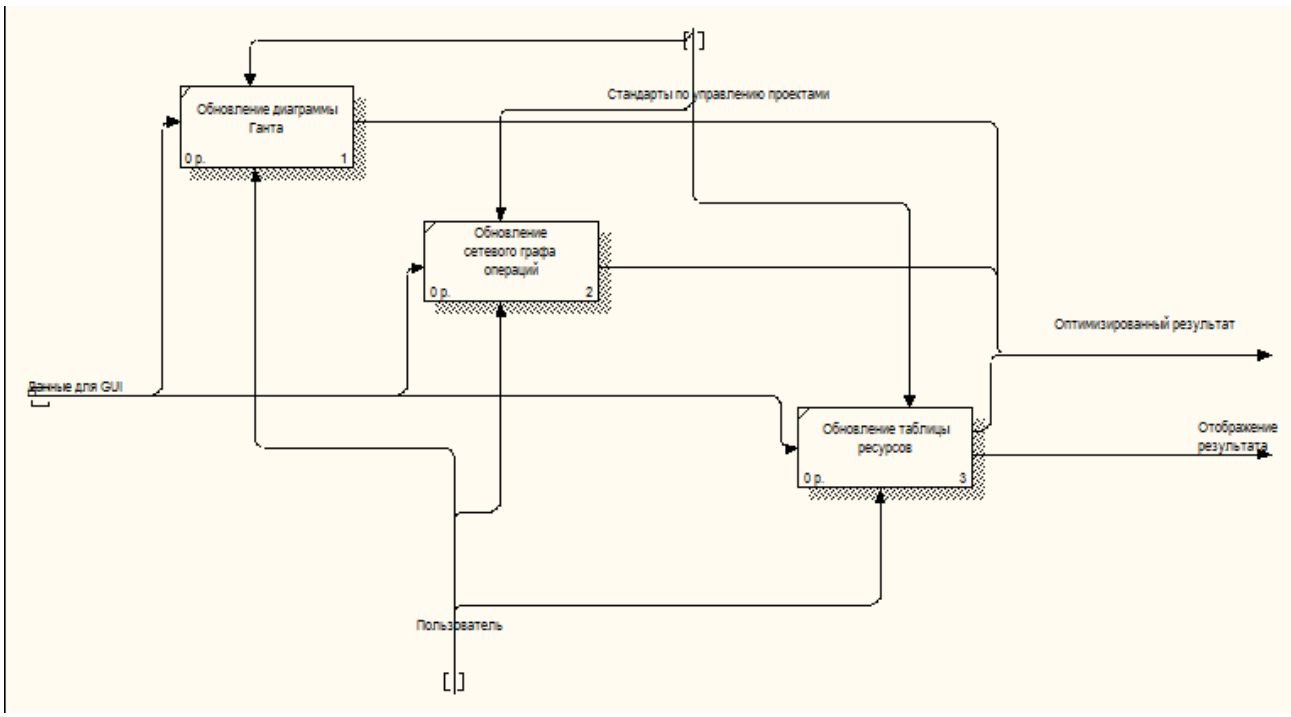

Рис. 24: IDEF0 диаграмма декомпозиции подсистемы обновления GUI

Приложение 2. Описание классов пакета Tropical

Класс Scheduling использует другие классы Matrix и Evaluation для рассчета временных критериев оптимизаци проекта. Статические методы принимают набор матриц и векторов в качестве аргументов (ограничения на старт-старт, старт-финиш, самое ранне начало операции и самое позднее её окончание). Возвращаемым значением является набор из двух матриц $\boldsymbol{x}$ и $\boldsymbol{y}$ матрица времени начала каждой операций и матрица времени окончания.

Таблица 1: Перечень методов класса Scheduling.

Наименование	Тип	Пеоедаваемые параметры	Краткая характеристика
flowTime	Collection <Matrix>	Collection <Matrix>	Находит временные характеристики проекта по критерию (28).
justInTime	Collection <Matrix>	Collection <Matrix>	Находит временные характеристики проекта по критерию (14).
makeSpan	Collection <Matrix>	Collection <Matrix>	Находит временные характеристики проекта по критерию (32).
maximumDeviation	Collection <Matrix>	Collection <Matrix>	Находит временные характеристики проекта по критерию (20).
plannedTerms	Collection <Matrix>	Collection <Matrix>	Находит временные характеристики проекта по критерию (34).

Класс Evaluations построен для проведения различных математических операций с и над матрицами. Все его методы как и в классе Scheduling являются статическими потому что не представляют с точки зрения ООП никакой динамической составляющей.

Таблица 2: Перечень методов класса Matrix.

Наименование	Тип	Пеоедаваемые параметры	Краткая характеристика
getColumnSize	int	-	Возвращает количество столбцов в матрице.
getInvert	Matrix	-	Возвращает инвертированную матрицу от данной.
getNorm	double	-	Возвращает норму данной матрицы.
getRowSize	int	-	Возвращает количество строк в матрице.
getTransponse	Matrix	-	Возвращает транспонированную матрицу от данной.
getValueAt	double	int r, int c	Возвращает значение по координатам r и c.
Matrix	конструктор	int rs, int cs	Конструктор создает матрицу заданного размера.
setColumnSize	-	int cs	Устанавливает определенное количество столбцов матрицы.
setRowSize	-	int rs	Устанавливает определенное количество строк матрицы.
setValueAt	-	double val, int r, int c	Устанавливает новое значение по координатам r и c.
toString	String	-	Возвращает полное описание матрицы.

Таблица 3: Перечень методов класса Evaluations.

Наименование	Тип	Пеоедаваемые параметры	Краткая характеристика
arrayMatrixMultip	Matrix	Collection <Matrix>	Производит перемножение нескольких матриц и возвращает результат.
arrayMatrixSum	Matrix	Collection <Matrix>	Производит перемножение нескольких матриц и возвращает результат.
getAsterate	Matrix	Matrix m	Возвращает матрицу для данной матрицы m по формуле (10).
getBigTrace	double	Matrix m	Возвращает тропическую сумму следа матрицы m.
getI	Matrix	int s	Возвращает еденичную матрицу I квадратного типа размера s×s.
getVectorI	Matrix	int s	Возвращает еденичный вектор-столбец размера s.
getTrace	double	Matrix m	Возвращает тропический след матрицы m.
matrixMultip	Matrix	Matrix m1, Matrix m2	Производит тропического произведение двух матриц.
matrixScalarMult	Matrix	Matrix m, double s	Производит тропическое умножение матрицы со скаляром.
matrixScalarSum	Matrix	Matrix m, double s	Производит тропическое суммирование матрицы со скаляром.
matrixSum	Matrix	Matrix m1, Matrix m2	Производит тропического произведение двух матриц.
multipConjugate Transpose	Matrix	Matrix m	Возвращает сопряженно-транспонированную матрицу от данной матрицы m.
resizeMatrix	Matrix	Matrix m	Проверяет матрицу на наличие нулевых строи стобцов.

Приложение 3. Описание классов пакета MainPackage

Таблица 4: Перечень методов класса Job.

Наименование	Тип	Пеоедаваемые параметры	Краткая характеристика
Job	Конструктор	String name, Date start, Date stop, Job next, Job previous, Resource res	Создает объект класса Job.
toString	String	-	Возвращает краткое описание класса.
getFullInfo	String	-	Возвращает полное описание класса.
getStartTime	Date	-	Возвращает время начала операции.
getStopTime	Date	-	Возвращает время окончания операции.
hasWorker	boolean	-	Проверка наличия ресурса.
setPreviousJob	boolean	Job j	Добавление работы в пул предыдущих операций.
removePrevious Jobs	-	-	Очищение пула предыдущих операций.
setNextJob	boolean	Job j	Добавление работы в пул следующих операций.
removeNextJobs	-	-	Очищение пула следующих операций.
setResource	boolean	Resource	Закрепление ресурса за текущей операцией.
removeResource	-	-	Открепление ресурса от текущей операции.
setStartTime	-	Date	Устанавливает время начала операции.
setStopTime	-	Date	Устанавливает время окончания операции.
getName	String	-	Возвращает имя класса.
setName	-	String	Устанавливает имя класса.
getNextJob	Collection <Job>	-	Получает список последующих операций.
getPrevious Job	Collection <Job>	-	Получает список предыдущих операций.
getResource	Resource	-	Получает ресурс текущей операции.

Таблица 5: Перечень методов класса Resource.

Наименование	Тип	Пеоедаваемые параметры	Краткая характеристика
Resource	Конструктор	-	Создает объект класса Resource.
Resource	Конструктор	String name, Job j	Создает объект класса Resource.
toString	String	-	Метод возвращает краткое описание класса.
addJob	boolean	Job j	Добавление выбранной операции в списка.
removeJob	boolean	Job j	Удаление выбранную операцию из списка.
getFullInfo	String	-	Возвращает полное описание класса.
getJobs	Collection <Job>	-	Получает список операций.
setName	-	String	Устанавливает имя класса.
removeAllJobs	-	-	Очищение пула следующих операций.

Класс Project создан для организаци взаимодествий между логикой приложения и его интерфейсом.

Таблица 6: Перечень методов класса Project.

Наименование	Тип	Пеоедаваемые параметры	Краткая характеристика
Project	Конструктор	-	Создает объект класса Project.
Project	Конструктор	String name	Создает объект класса Project.
Project	Конструктор	String name, JFrame window	Создает объект класса Project.
getAllJobs	Collection <Job>	-	Получает список операций.
getAllResources	Collection <Resource>	-	Получает список ресурсов проекта.
getAllUnused Resources	Collection <Resource>	-	Получает список неиспользованных ресурсов проекта.
setStartTime	-	-	Устанавливает время начала проекта.
setStopTime	-	-	Устанавливает время окончания проекта.
getStartTime	Date	-	Возвращает время начала проекта.
getStopTime	Date	-	Возвращает время окончания проекта.
addGUI	-	JFrame window	Сохранение ссылки на интерфейс.
addResource	-	Resource r	Добавление нового ресурса в проект.
addJob	-	Job j	Добавление новой операции в проект.
valueChanged	-	-	Вызывается при любом изменении в проекте.
validateProject	-	-	Проверка валидности проекта.
toString	String	-	Метод возвращает краткое описание класса.
getFullInfo	String	-	Возвращает полное описание класса.

Приложение 4. Flow-time оптимизация

В данном зафиксированы алгоритмы нахождения минимума целевой функции при нахождения минимума максимального времени цикла операций.

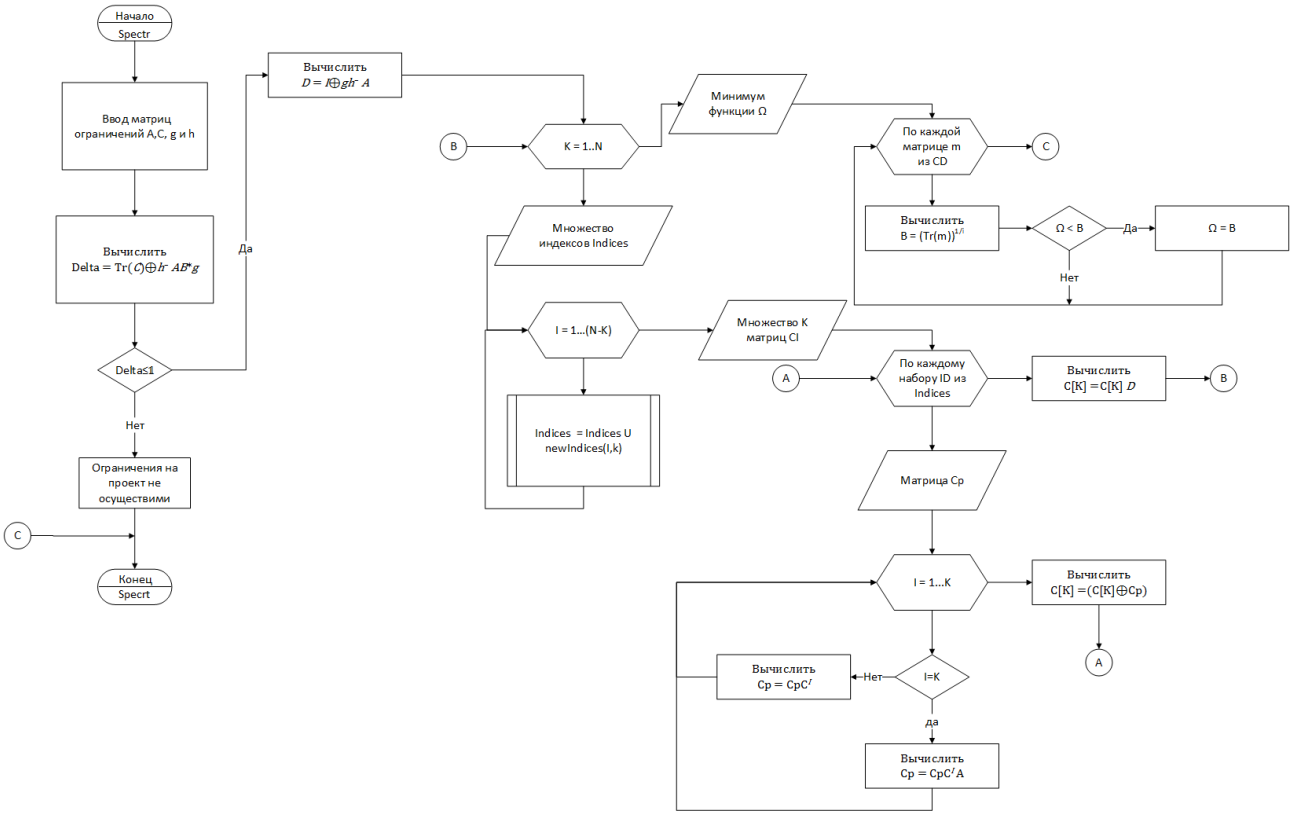

Рис. 25: Алгоритм нахождения минимум flow-time оптимизации

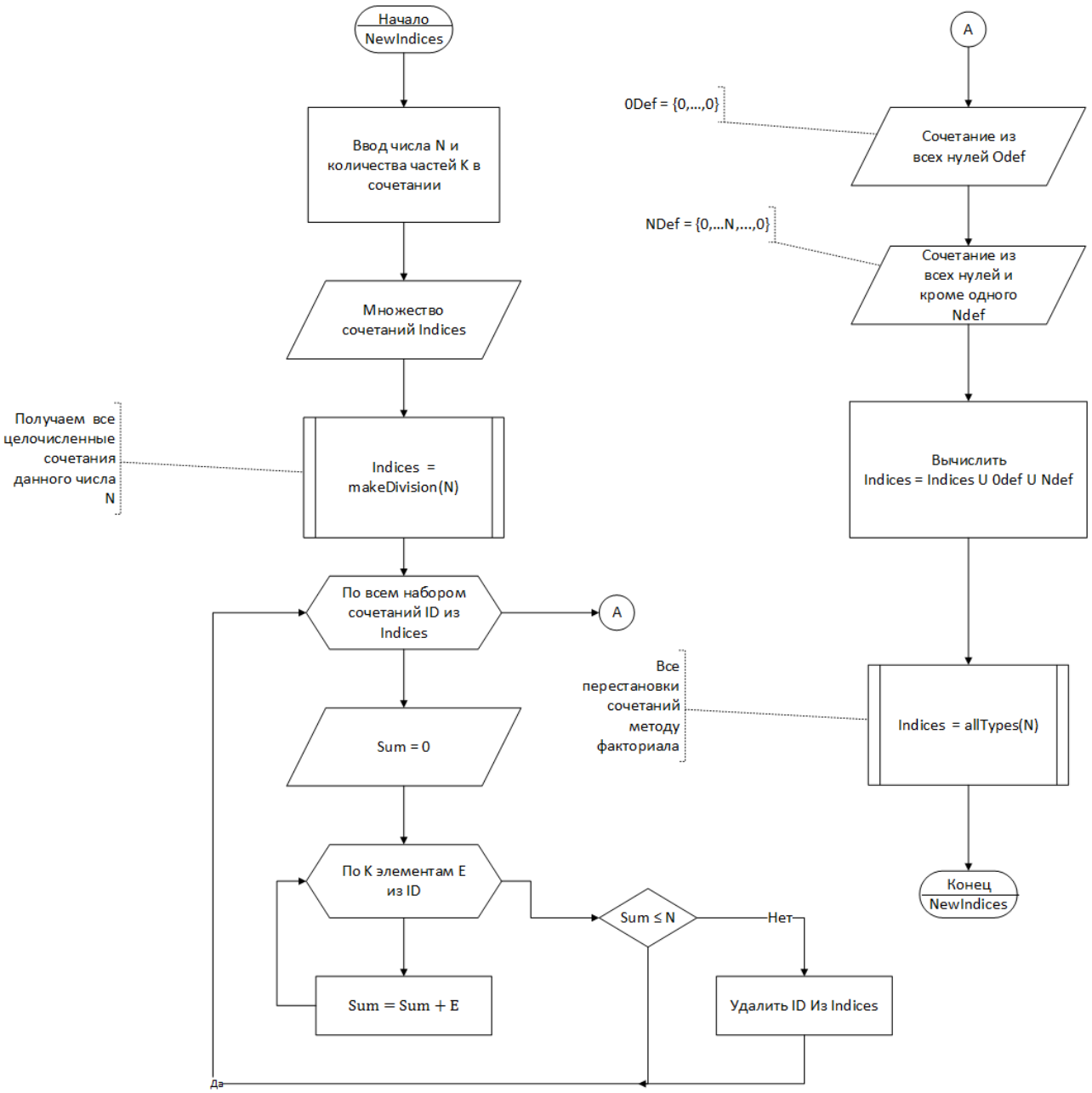

Рис. 26: Алгоритм нахождения всех возможных перестановок сочетаний числа n

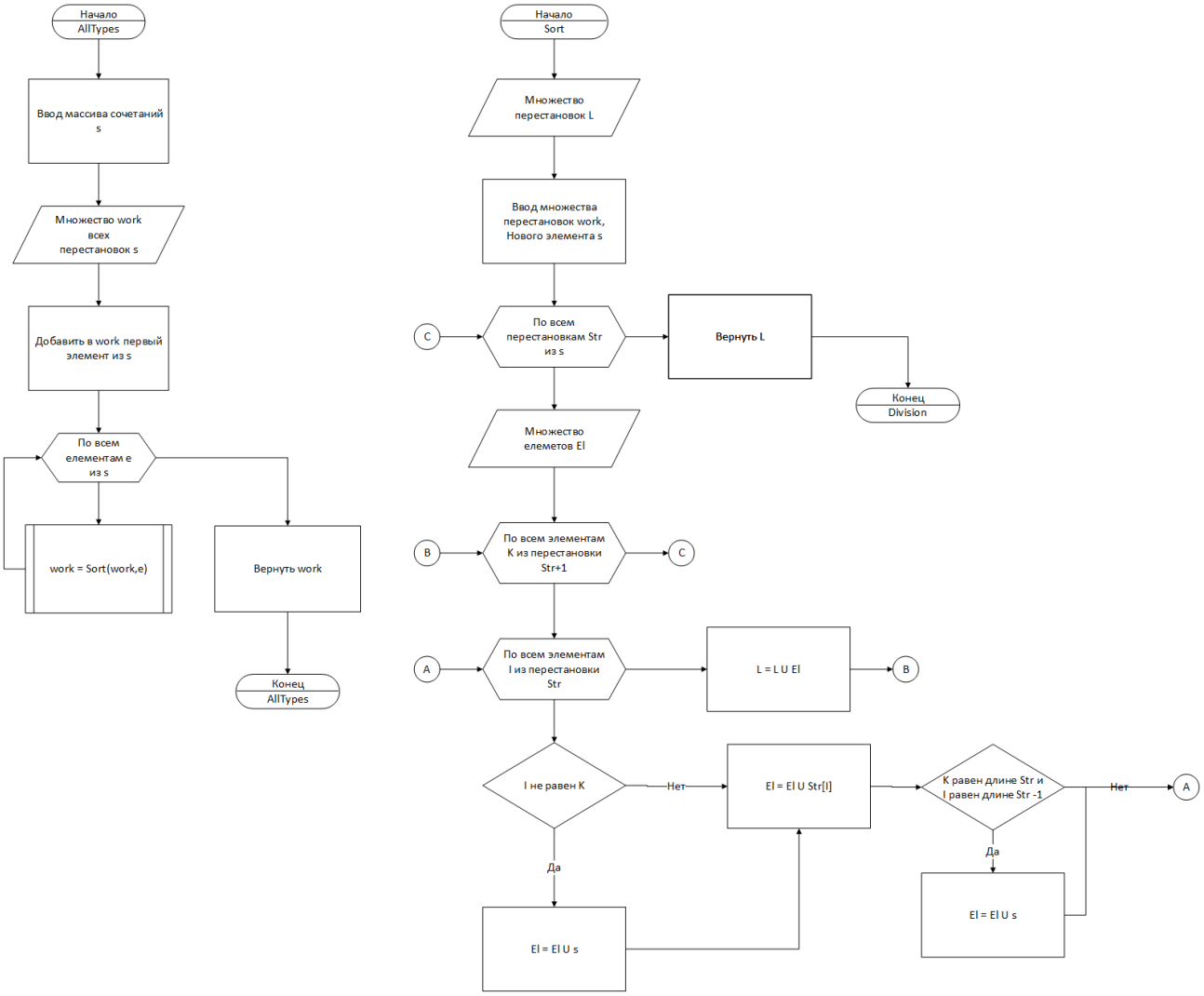

Рис. 27: Алгоритм нахождения всех возможных сочетаний числа n

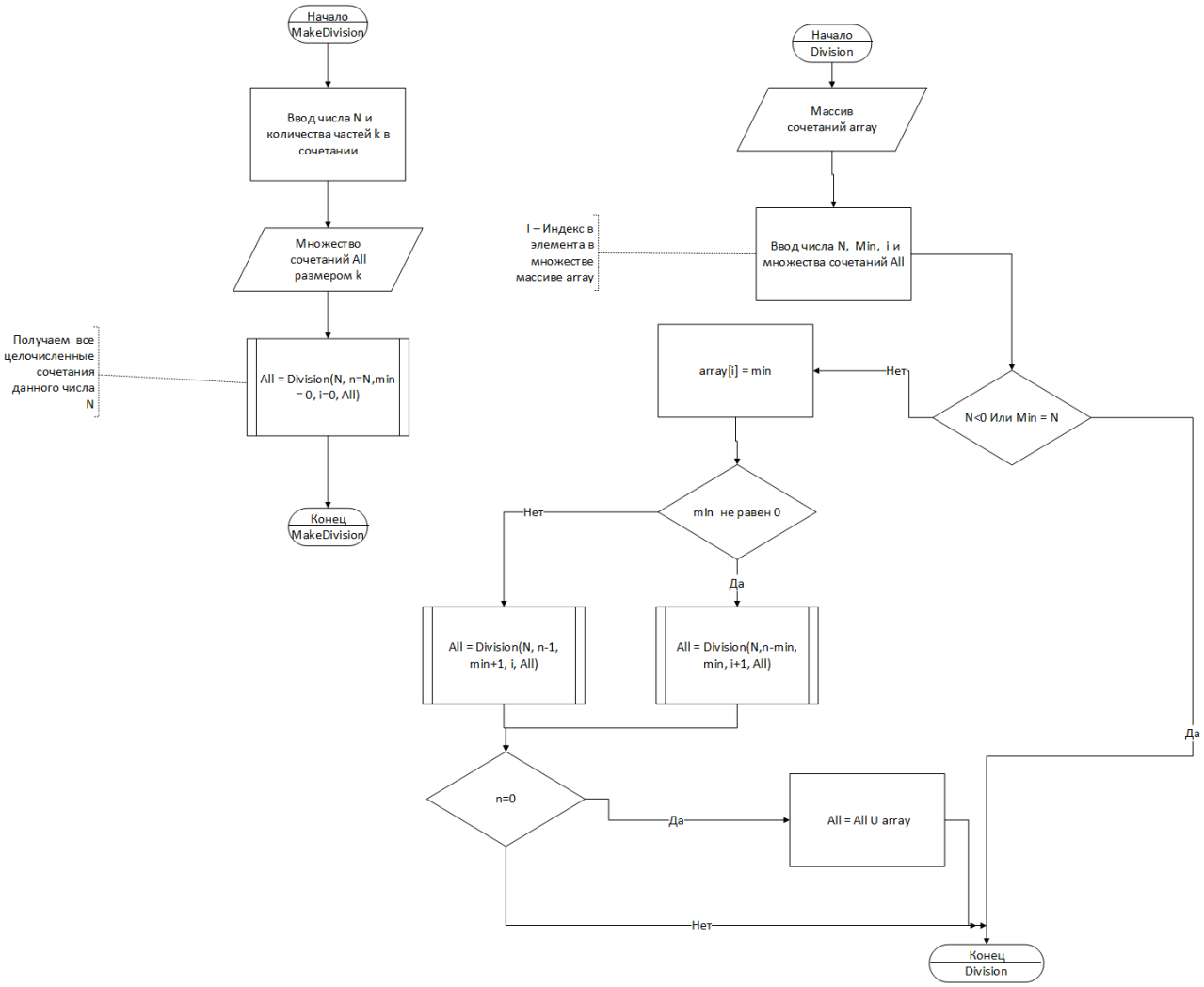

Рис. 28: Алгоритм нахождения всех возможных перестановок набора чисел