Санкт-Петербургский государственный университет

На правах рукописи

Губанов Сергей Александрович

Решение минимаксных задач оптимального планирования проектов с использованием методов идемпотентной алгебры

Научная специальность 1.2.2.
Математическое моделирование, численные методы и комплексы программ

Диссертация
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
доктор физико-математических наук
Кривулин Николай Кимович

Санкт-Петербург
2023

Содержание

Введение

Актуальность темы диссертации

Диссертационная работа посвящена разработке моделей, алгоритмов и программных средств решения задач управления проектами при помощи методов тропической оптимизации. В работе исследуется ряд актуальных задач оптимального планирования сроков выполнения проектов, которые формулируются как новые задачи тропической оптимизации и предлагается их прямое аналитическое решение.

В процессе развития предприятий их деятельность непрерывно усложняется и помимо простых повторяемых действий возникает необходимость в осуществлении неповторяемых взаимосвязанных действий, называемых сложными действиями или проектами. Для управления сложными неповторяемыми действиями, включая временное планирование таких действий, применяются различные подходы, которые объединяются под термином «управление проектами». В качестве примеров таких подходов можно привести методы сетевого планирования, которые опираются на анализ проекта при помощи графовых и сетевых моделей, например метод критического пути (CPM — Critical Path Method) [1] и метод оценки и пересмотра планов (PERT — Program Evaluation and Review Technique) [2]), методы линейного и смешанного целочисленного линейного программирования [3–5]

Хотя в общем случае задачи управления проектами достаточно сложные и являются NP-трудными, задачи временного планирования (из-за отсутствия наложенных ограничений по стоимости и ресурсам), как правило, могут быть сформулированы и решены как задачи линейного программирования. Как следствие, они могут быть численно решены за полиномиальное время при помощи соответствующих вычислительных процедур, таких как алгоритмы Кармаркара и Флойда-Уоршалла.

Применение методов тропической математики является одним из эффективных способов решения практических задач оптимизации, которые возникают в подобных задачах.

Тропическая математика изучает полуполя и полукольца с идемпотент-

ным сложением [6–11], а также различные связанные с ними вычислительные задачи. Часто функции, являющиеся нелинейными и негладкими в обычной математике после их перевода в тропический вид становятся линейными. Такие задачи могут быть решены посредством вычисления тропических собственных чисел и векторов матриц и решения тропических уравнений и неравенств. Более того, многие алгоритмы линейной алгебры (например метод Якоби и алгоритм Гаусса-Зейделя) имеют идемпотентные аналоги, позволяющие создавать эффективные вычислительные алгоритмы. Как следствие, появляются новые возможности для анализа подобных задач, что зачастую приводит к упрощению как процедур их численного решения, так и интерпретации полученных результатов.

Кроме того, при помощи линейных уравнений, сформулированных в терминах идемпотентной алгебры, может быть описано поведение многих динамических систем, которые используются, например, при решении задач планирования и управления, что дает возможность предлагать новые методы имитационного моделирования таких систем.

Методы и алгоритмы тропической математики успешно применяются для решения многих практических задач, включая задачи размещения, принятия решений и задачи сетевого планирования. Для решения таких задач обычно используются конечношаговые вычислительные методы, включая методы линейного и смешанного целочисленного линейного программирования, методы дискретной и комбинаторной оптимизации, в которых используются итерационные процедуры, позволяющие численно получить одно из решений, если решение существует, либо убедиться в его отсутствии.

В отличие от указанных численных методов, методы тропической оптимизации позволяют находить все множество решений аналитически в явном виде в компактной матрично-векторной форме. Аналитическая форма получаемых результатов является более информативной и удобной. Она предоставляет больше возможностей для формального анализа решений математическими методами, может больше рассказать о фундаментальной структуре набора решений и лучше объяснить влияние входных параметров на решение. Аналитическое представление множества решений в матрично-векторной форме может служить ос-

новой для уточнения решений путем нахождения среди них решений, которые удовлетворяют дополнительным ограничениям и критериям оптимальности.

Кроме того, аналитическое решение обычно является более простым для алгоритмической и программной реализации. Процедура решения состоит из конечного числа матрично-векторных операций и имеет невысокую полиномиальную вычислительную сложность. Такая процедура обеспечивает возможность эффективной программной реализации для выполнения как на последовательных, так и на параллельных вычислительных системах. Наконец, решение, полученное в аналитическом виде, является более точным, поскольку позволяет избежать накопления ошибок округления, которые обычно возникают в численных алгоритмах.

Методы тропической оптимизации успешно применялись для решения задач временного планирования проектов с различными типами ограничений и критериями оптимальности. Временные ограничения включают отношения предшествования для времени начала и завершения работ проекта (ограничения типа «старт-финиш», «старт-старт», «финиш-старт» и «финиш-финиш»), а также верхние и нижние границы для времени начала («время выпуска», «время окончания выпуска») и завершения («крайний срок завершения») каждой работы. Критериями оптимальности являются временные характеристики проекта, которые требуется минимизировать (максимизировать), такие как максимальная продолжительность проекта, максимальное отклонение от директивных сроков завершения проекта, максимальный разброс времени начала работ проекта и другие.

Для ряда постановок задач временного планирования, которые определяются выбором критерия оптимальности и совокупности ограничений, известны аналитические решения, полученные с помощью методов тропической оптимизации. Такие решения, однако, известны не для всех критериев оптимальности, которые могут представлять интерес для практических задач планирования. Кроме того, существующие решения обычно учитывают только часть возможных ограничений, в то время как для практических задач обычно является существенным учет всех известных типов ограничений. Поэтому, постановка и решение новых ранее не изученных задач временного планирования проектов

на основе применения и дальнейшего развития моделей и методов тропической оптимизации, включая разработку вычислительных процедур, алгоритмов и программных средств, представляется весьма актуальной.

Степень разработанности темы в литературе

Во многих задачах оптимизации, таких как задачи размещения [8, 12–14] и сетевого планирования [15] целевая функция и ограничения могут быть описаны при помощи операций максимума и минимума, а также арифметических операций. Нахождение решений подобных задач обычно сопряжено с некоторыми трудностями, которые могут быть связаны, в частности, с нелинейностью и негладкостью целевой функции и ограничений. Эффективным способом упрощения подобных задач является их переформулирования на языке тропической математики и последующее использование ее результатов [16–18] для получения решения задачи. Под задачами тропической оптимизации обычно понимаются задачи оптимизации, сформулированные в терминах тропической математики. Такие задачи являются важной составной частью современных исследований, посвященных тропической математике. Начало развитию данного направления положили работы [9, 15, 19], последние результаты исследований можно найти, например, в работах [13, 20–28].

Отдельно необходимо отметить класс оптимизационных задач, которые могут быть сформулированы в терминах тропической математики и заключаются в минимизации нелинейных функционалов, с ограничениями в виде линейных векторных неравенств [29]. Решение подобных задач обычно опирается на экстремальное свойство спектрального радиуса матрицы и связано с его вычислением [20, 23, 25, 30].

Часто при представлении задач временного планирования на языке тропической математики возникают новые задачи тропической оптимизации, требующие особого подхода к их решению [8, 22–24, 29–37]

Настоящая работа посвящена решению ряда задач временного планирования и возникающих при их представлении на языке тропической математики новых задач тропической оптимизации. Исследуемые задачи временного планирования с меньшим набором ограничений уже решались в работах [24, 33, 35, 37]. В представленной диссертации предложено расширение данных результатов на

задачи с большим набором ограничений.

Объектом исследования является разработка моделей и методов тропической оптимизации для решения прикладных задач. **Предметом исследования** является построение на основе методов тропической оптимизации аналитических решений, вычислительных процедур и программных средств для задач временного планирования сроков выполнения проектов.

Цель работы

Целью диссертационной работы является расширение существующего аппарата решения задач составления оптимальных календарных графиков выполнения работ при помощи методов тропической оптимизации на новые ранее не изученные задачи планирования. Рассматриваются задачи с более общим набором ограничений и критериями оптимальности плана, для которых требуется найти аналитические решения, построить вычислительные процедуры и разработать программные средства.

Основные задачи

Для достижения указанной цели необходимо сформулировать и решить следующие задачи:

1. Представить ряд новых задач составления оптимальных графиков выполнения работ проекта в форме минимаксных задач оптимизации с различными видами ограничений. В качестве критериев оптимальности используются:

   - продолжительность проекта;
   - отклонение от заданных директивных сроков выполнения работ проекта;
   - разброс времени завершения работ проекта;
   - разброс времени начала выполнения работ проекта.

   Ограничения заданы при помощи:

   - минимально допустимых временных промежутков между началами работ;

- минимальных временных промежутков между началом и завершением работ;
- минимальных временных промежутков между завершением и началом работ;
- самого раннего и самого позднего допустимого времени начала работ;
- наиболее позднего допустимого времени завершения работ.

2. Сформулировать указанные новые задачи планирования на языке тропической математики (в форме задач тропической оптимизации).

3. Разработать методы нахождения полного решения задач оптимизации с линейными ограничениями на множестве допустимых значений в явном виде в простой аналитической форме, основанные на применении аппарата тропической математики и представлении его в форме теорем.

4. Разработать вычислительные процедуры и программные средства решения задач тропической оптимизации.

5. Разработать приложения полученных результатов для решения актуальных задач планирования работы медицинских учреждений.

Соответствие диссертации паспорту специальности

Содержание диссертационного исследования соответствует следующим пунктам паспорта специальности 1.2.2 - «Математическое моделирование, численные методы и комплексы программ»: разработка новых математических методов моделирования объектов и явлений (пункт 1); развитие качественных и приближенных аналитических методов исследования математических моделей (пункт 2); разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий (пункт 3); реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента (пункт 4);

Научная новизна

Научная новизна заключается в следующем:

- Рассмотрен ряд новых задач планирования сроков выполнения проектов с различными критериями оптимальности и видами ограничений.

- Для рассмотренных задач планирования построено их представление в виде задач тропической оптимизации с различными целевыми функциями и ограничениями.

- Для исследуемых задач найдены новые общие аналитические решения, получено их представление в компактной матрично-векторной форме.

- На основе полученных результаты созданы новые эффективные вычислительные процедуры и разработаны программные средства нахождения решений указанных задач.

- Полученные результаты впервые применены для решения актуальных задач планирования медицинских мероприятий.

Методы исследования

В работе применяются инструменты линейной алгебры, общей теории чисел, математического моделирования. Кроме того, используются методы оптимизации, компьютерного моделирования, построения математических моделей сложных систем и идемпотентной математики. Программирование велось на языке высокого уровня C++ с широким применением подходов объектно-ориентированного программирования, которые естественным образом использовались для описания классов, реализующих матрично-векторные операции идемпотентной алгебры и решение задач тропической оптимизации.

Степень достоверности Достоверность изложенных в работе теоретических результатов обеспечивается их строгими математическими доказательством. В работе приведены полные доказательства для теорем доказанных соискателем. Для прочих использованных результатов представлены ссылки на доказательства.

Теоретическая и практическая ценность работы

Результаты диссертационной работы имеют теоретическую ценность и заключаются в получении полного решения для ряда задач тропической оптими-

зации. Решения имеют матрично-векторную форму, а, следовательно, вычисления могут быть естественным образом распараллелены. Кроме того, решения представлены в удобном виде в простой аналитической форме, что значительно упрощает проведение дальнейших аналитических исследований при помощи математических методов. Полученные теоретические результаты использованы для нахождения решений актуальных практических задач планирования сроков выполнения проектов и предложено их использование для повышения эффективности работы медицинских учреждении, и для ликвидации чрезвычайных ситуаций (ЧС), что особенно актуально в наше время. Для представленных задач составления оптимальных календарных графиков выполнения работ проектов также разработаны программные средства, реализующие предложенные решения.

Результаты диссертационной работы были получены при поддержке грантов Российского гуманитарного научного фонда РГНФ № 13-02-00338а – «Модели и методы тропической математики в прикладных задачах экономики и управления» и РГНФ №16-02-00059 – «Развитие моделей и методов тропической математики в прикладных задачах экономики и управления а также грантов Российского фонда фундаментальных исследований РФФИ №18-010-00723А – «Разработка моделей и методов тропической математики для прикладных задач экономики и управления» и РФФИ №20-010-00145 – «Модели и методы тропической оптимизации в прикладных задачах экономики и управления».

Личный вклад

Автор принимал активное участие в математическом доказательстве, программной реализации и представлении результатов работ на семинарах и конференциях. В опубликованных работах можно выделить следующие результаты, полученные лично автором:

- Прямые полные решения, сформулированные в виде теорем для следующих задач оптимального планирования с ограничениями: задача о минимизации продолжительности проекта, задача минимизации отклонения от заданных директивных сроков выполнения работ проекта, задача минимизации и максимизации разброса времени завершения работ проекта,

задачи минимизации и максимизации разброса времени начала выполнения работ проекта.

- Сформулированные в виде теорем решения вспомогательных задач тропической оптимизации и их строгие математические доказательства.

- Вычислительные процедуры решения и анализ их вычислительной сложности.

- Библиотека классов, написанная на языке высокого уровня C++, реализующая полученные алгоритмы поиска решений.

- Разработка приложений полученных результатов для решения актуальных задач планирования работы медицинских учреждений.

Краткое описание работы

Диссертационная работа состоит из введения, восьми глав, заключения и приложения. Полный объем диссертации составляет 133 страницы машинописного текста. Список литературы содержит 110 наименований. Во введении обосновывается актуальность темы диссертации, представляется обзор соответствующей литературы, задаются цели и задачи работы, и аргументируется их научная ценность.

В главе 1 приведены основные определения, связанные с задачами временного планирования сроков выполнения проекта. В 1.1 определяется понятие управления проектами, приводится краткий обзор развития данной темы и предлагается представление задач управления проектами в виде задачи максимизации или минимизации некоторого критерия оптимальности при ограничениях в форме равенств и неравенств. Далее, в разделе 1.2 формально задаются условия на порядок выполнения работ проекта. Наконец, в разделе 1.3 приведены используемые в исследуемых задачах критерии оптимальности.

Затем, в главе 2 приводятся результаты идемпотентной алгебры и тропической оптимизации, которые потребуются для решения исследуемых задач временного планирования календарных сроков выполнения работ. В разделе 2.1 представлен обзор основных определений тропической математики, необходи-

мых для решения задач управления проектами методами тропической оптимизации. Далее, в разделе 2.2 определяется понятие задачи тропической оптимизации, приводятся примеры различных задач и представлено решение задач оптимизации, которые в дальнейшем будут использованы для решения задач тропической оптимизации с более строгими ограничениями.

В главе 3 представлена задача минимизации продолжительности выполнения проекта с ограничениями на время выполнения его работ. В разделе 3.1 сформулирована и решена задача тропической оптимизации, которая будет использована для решения рассматриваемой в данной главе задачи оптимального планирования. Затем, в разделе 3.2 формально задана задача минимизации продолжительности выполнения проекта с ограничениями и представлено ее решение. Наконец, в разделе 3.3 полученный результат используется для задачи об организации наиболее быстрой вакцинации.

Затем, в главе 4 предложено решение задачи о минимизации максимального отклонения сроков выполнения работ проекта от заданных директивных сроков с известными ограничениями на время выполнения работ. В разделе 4.1 предложено решение задачи тропической оптимизации, к которой впоследствии в разделе 4.2 сводится исследуемая задача сетевого планирования. Наконец, в 4.3 найденный результат используется для решения задачи об организации эвакуации населения в случае чрезвычайной ситуации.

В главе 5 рассматривается задача минимизации разброса времени завершения работ проекта при заданных ограничениях. В 5.1 сформулирована и решена задача тропической оптимизации, к которой в 5.2 сводится исследуемая в главе задача оптимального управления. Далее, в разделе 5.3 приводится пример практической задачи, которая может быть решена с использованием представленных результатов и заключается в оптимизации проведения мероприятий по диспансеризации.

Затем, в главе 6 рассматривается задача максимизации разброса времени завершения работ проекта с заданными ограничениями. В разделе 6.1 предложено решение задачи тропической оптимизации, к которой далее, в разделе 6.2, сводится исследуемая задача сетевого планирования. В разделе 6.3 полученный результат применяется для оптимизации расписания работы медицинского цен-

тра, проводящего медицинские процедуры, связанные со вредным воздействием на организм.

В главе 7 рассматривается решение задачи минимизации разброса времени начала работ проекта с ограничениями на время и последовательность их выполнения с помощью методов идемпотентной алгебры. Далее, в разделе 7.1 формально задается задача сетевого планирования и сводится к известной задаче тропической оптимизации. Затем, в 7.2 полученный результат используется для нахождения решения задачи об организации помощи раненым в случае чрезвычайной ситуации.

Наконец, в главе 8 исследуется задача максимизации разброса времени начала работ проекта с ограничениями на время их выполнения. В разделе 8.1 сформулирована и решена задача тропической оптимизации, к которой затем, в разделе 8.2 сводится рассматриваемая задача оптимального управления. В разделе 8.3 предложено применение полученного результата для оптимизации работы поликлиники в период карантина.

Положения, выносимые на защиту

- Разработана математическая модель задачи минимизации продолжительности проекта, задачи минимизации максимального отклонения от директивных сроков, задач минимизации и максимизации максимального разброса времени завершения работ проекта, а также задач минимизации и максимизации максимального разброса времени начала выполнения работ проекта с ограничениями на время их выполнения.

- Сформулирован и полностью решен ряд задач тропической оптимизации с различными целевыми функциями и линейными ограничениями.

- Получены полные аналитические решения задач планирования сформулированных как задачи тропической оптимизации. Результаты оформлены в виде теорем.

- Разработаны вычислительные процедуры решения задач тропической оптимизации и исследована их вычислительная сложность.

- Разработаны программные средства для решения задач тропической оптимизации.

- Разработаны приложения полученных результатов для решения актуальных задач планирования работы медицинских учреждений.

Апробация результатов

Результаты диссертации докладывались на 6-й Научно-практической internet конференции «Междисциплинарные исследования в области математического моделирования и информатики» (Тольятти, 2015 г); Всероссийской научной конференция по проблемам информатики СПИСОК-2017 (Санкт-Петербург, 2017 г); Всероссийской научной конференция по проблемам информатики СПИСОК-2019 (Санкт-Петербург, 2019 г); Всероссийской научной конференция по проблемам информатики СПИСОК-2022 (Санкт-Петербург, 2022 г); Международной конференции Polinomial Computer Algebra (PCA - 2022); 23-й Всероссийской конференции молодых учёных по математическому моделированию и информационным технологиям (Новосибирск, 2022 г); 2-м Международном форуме «Математические методы и модели в высокотехнологичном производстве» (Санкт-Петербург, 2022 г); 15-й Международной научно-технической конференции «Современные проблемы машиностроения» (Томск, 2022 г); семинаре по стохастическому программированию на кафедре системного программирования, а также на семинарах кафедры статистического моделирования Математико-механического факультета Санкт-Петербургского университета в процессе обучения соискателя в аспирантуре.

Публикации Основные результаты диссертационной работы представлены в печатных работах [38–41], опубликованных в рецензируемых научных изданиях, рекомендованных ВАК при Минобрнауки России. Всего автором опубликовано 12 печатных работ [38–49], посвященных решению различных задач тропической оптимизации и решению задач сетевого планирования и управления, посредством их сведения к известным оптимизационным задачам и применения методов тропической оптимизации. Все решения получены в аналитическом виде в удобной матрично-векторной форме.

Глава 1

Задачи планирования сроков выполнения проектов

В данной главе приведены основные определения, связанные с задачами временного планирования сроков выполнения проекта. В разделе 1.1 определяется понятие управления проектами, приводится краткий обзор развития данной темы и предлагается представление задач управления проектами в виде задачи максимизации или минимизации некоторого критерия оптимальности при ограничениях в форме равенств и неравенств. Далее, в разделе 1.2 формально задаются условия на порядок выполнения работ проекта. Наконец, в разделе 1.3 приведены используемые в исследуемых задачах критерии оптимальности.

1.1 Задачи управления проектами

Управлением проектами является согласование действий, которые выполняются для достижения целей проекта при разумном распределении имеющихся ресурсов. Одними из наиболее распространенных задач управления проектами являются задачи сетевого планирования[50, 51].

Первые методы решения подобных задач были предложенные в конце пятидесятых годов метод критического пути (CPM – Critical Path Method) [1] и метод оценки и пересмотра планов (PERT – Program Evaluation and Review Technique) [2].

Подобные задачи могут быть представлены как задачи оптимизации, для решения которых существуют различные методы и алгоритмы. Одним из перспективных способов решения задач сетевого планирования является предложенный в работах [21, 52, 53], метод, заключающийся в решении задачи с использованием моделей и методов тропической оптимизации.

Тропическая (идемпотентная) математика является областью математики, которая изучает полукольца с идемпотентным сложением [7, 22, 26, 54, 55]. В работах [15, 56, 57] она предложена как естественный инструмент, использую-

щийся для описания и решения реальных практических задач сетевого планирования.

Обычно подобные задачи формулируются как задачи минимизации (максимизации) целевой функции на конечномерных полумодулях над идемпотентным полуполем при наличии различных ограничений, выраженных при помощи линейных уравнений и неравенств.

Далее опишем задачи оптимизации, которые возникают при планировании графиков выполнения работ различных проектов и сводятся к максимизации или минимизации некоторой целевой функции при ограничениях в форме равенств и неравенств. Целевая функция определяется критерием оптимальности плана, а ограничения задаются условиями на время начала и завершения работ проекта. В этих задачах целевая функция и ограничения выражаются при помощи операций максимума, сложения и вычитания.

1.2 Условия на порядок выполнения работ

Рассмотрим проект, который заключается в выполнении n работ при заданных ограничениях на время их выполнения. Будем считать, что ни одна работа не может завершиться, пока не прошло некоторое заданное время после начала других работ (ограничение «старт-финиш»). Также определен минимальный временной интервал между временами начала любых двух работ (ограничение «старт-старт»). Наиболее раннее время начала каждой работы задается при помощи ограничения «время выпуска», а наиболее позднее допустимое время начала работы при помощи ограничения «время прекращения выпуска». Наиболее позднее возможное время завершения работы определяется ограничением «крайний срок завершения». Считаем, что каждая работа завершается сразу же после выполнения всех заданных ограничений на время её завершения.

Представим введенные выше ограничения в более формальной форме в виде линейных уравнений и неравенств. Для этого для каждой работы $i = 1, \ldots, n$ определим следующие обозначения:

x_i – время начала выполнения работы i;

y_i – время завершения выполнения работы i;

b_{ij} – минимальный временной промежуток между началом работы $j = 1, \ldots, n$ и началом работы i;

c_{ij} – минимальный временной интервал между началом работы $j = 1, \ldots, n$ и завершением работы i;

d_{ij} – минимальный временной интервал между завершением работы $j = 1, \ldots, n$ и началом работы i;

В случае, если величина интервала b_{ij}, c_{ij} или d_{ij} не задана, то считаем ее равной $-\infty$;

g_i – самое раннее допустимое время начала работы i;

h_i – самое позднее допустимое время начала работы i;

f_i – наиболее позднее допустимое время завершения работы i.

Ограничения «время выпуска», «время прекращения выпуска» и «крайний срок завершения» обозначаются как g_i, h_i и f_i и задают для величин x_i и y_i нижнюю и верхнюю границы:

$$g_i \leq x_i \leq h_i, \qquad y_i \leq f_i. \tag{1.1}$$

Ограничения «старт-старт» для работы i могут быть определены для всех $j = 1, \ldots, n$ в виде неравенств

$$b_{ij} + x_j \leq x_i. \tag{1.2}$$

Объединение всех неравенств по j дает эквивалентное неравенство

$$\max_{1 \leq j \leq n} (b_{ij} + x_j) \leq x_i. \tag{1.3}$$

Ограничения вида «старт-финиш» для каждой работы i могут быть представлены при помощи неравенства

$$\max_{1 \leq j \leq n} (c_{ij} + x_j) \leq y_i. \tag{1.4}$$

Будем считать, что работа завершается сразу же после выполнения заданных для нее ограничений «старт-финиш», а, следовательно, хотя бы одно из неравенств выполняется как равенство. В таком случае неравенство можно заменить эквивалентным равенством

$$\max_{1\le j\le n}(c_{ij}+x_j)=y_i. \tag{1.5}$$

Для каждой работы i представим все отношения предшествования вида «финиш-старт» в форме неравенства

$$\max_{1\le j\le n}(d_{ij}+y_j)\le x_i. \tag{1.6}$$

1.3 Критерии оптимальности временного планирования

Часто в задачах оптимального управления проектами возникает необходимость минимизировать общую продолжительность выполнения проекта. Для подобных задач может быть использован следующий критерий оптимальности, который требуется минимизировать

$$\max_{1\le i\le n} y_i-\min_{1\le i\le n} x_i=\max_{1\le i\le n} y_i+\max_{1\le i\le n}(-x_i). \tag{1.7}$$

В задачах управления проектами может возникнуть необходимость, по возможности, минимизировать максимальное временное отклонение времени начала или завершения от некоторого заданного интервала. Пусть p_i и q_i обозначают соответственно нижнюю и верхнюю границы допустимых интервалов для сроков начала для каждой работы $i=1,\dots,n$. Считаем, что в задаче требуется по возможности минимизировать максимальный выход за нижнюю границу интервала p_i-x_i и максимальный выход за верхнюю границу x_i-q_i. Определим критерий оптимальности, который требуется минимизировать в следующем виде

$$\max\left(\max_{1\le i\le n}(p_i-x_i),\max_{1\le i\le n}(x_i-q_i)\right). \tag{1.8}$$

При составлении планов производства часто возникает необходимость завершить выполнение всех работ в одно и то же время. Подобный подход называется планированием в соответствии с принципом «точно в срок» (just-in-time)

[50, 51]. Как критерий оптимальности плана может быть использован максимальный разброс между временем завершения всех работ

$$\max_{1\le i\le n} y_i - \min_{1\le i\le n} y_i = \max_{1\le i\le n} y_i + \max_{1\le i\le n} (-y_i)\,, \tag{1.9}$$

который нужно минимизировать. Обратим внимание, что в случае возникновения необходимости избежать одновременного завершения всех работ можно применить тот же критерий, но его необходимо будет максимизировать

В задачах составления оптимального графика может потребоваться избежать избежать одновременного начала выполнения работ. Для подобных задач можно предложить следующий критерий оптимальности

$$\max_{1\le i\le n} x_i - \min_{1\le i\le n} x_i = \max_{1\le i\le n} x_i + \max_{1\le i\le n} (-x_i)\,, \tag{1.10}$$

который нужно максимизировать. При возникновении необходимости обеспечить минимальный разброс времени между началом работ может быть использован тот же самый критерий, но его потребуется минимизировать.

Решаемые в работе задачи сетевого планирования представляют собой различные комбинации из представленных критериев оптимальности и ограничений. Подобные задачи могут быть применены в целом ряде реальных практических задач, например, при оптимизации работы поликлиники или медицинской лаборатории.

Глава 2

Идемпотентная алгебра и тропическая оптимизация

Представим основные определения и результаты идемпотентной алгебры и тропической оптимизации, которые потребуются для решения исследуемых задач временного планирования календарных сроков выполнения работ. В разделе 2.1 приведен обзор основных определений тропической математики, необходимых для решения задач управления проектами методами тропической оптимизации. Затем, в разделе 2.2 определяется понятие задачи тропической оптимизации, приводятся примеры различных задач и представлено решение задач оптимизации, которые в дальнейшем будут использованы для решения задач тропической оптимизации с более строгими ограничениями.

2.1 Элементы тропической математики

Для формулировки и решения рассматриваемых в настоящей работе задач тропической оптимизации потребуются алгебраические определения, обозначения и предварительные результаты идемпотентной алгебры[7, 22, 26, 54, 55].

2.1.1 Введение в тропическую математику

Понятие полукольца, являющееся центральным понятием тропической математики впервые было неявно использовано в книге Р. Дедекинда [58] 1894 года и некоторых других работах. В работах [59, 60] указывается, что явно понятие полукольца было введено Г. Вандивером в 1934 году в работе [61]. В связи с большим количеством связанных теоретических и прикладных задач, полукольца и различные способы их применения на протяжении длительного времени является популярной темой для исследований. В 1950-60 годы началось активное развитие тропической математики. Особенно большой вклад внесли работы С. К. Клини [62] в 1956 и Р. А. Канингхейм-Грина [63] в 1962, Н. Н. Во-

робьева [64–66] в 1963, 1967 и 1970 годах и И. В. Романовского [67, 68] в 1962 и 1964 годах.

Одной из первых российских статей, посвященных тропической математике является работа Н.Н. Воробьева [65], появлению которой предшествовали различные теоретические работы (например по теории структур [69] и $\mathcal{K}$-пространств [70]) и многочисленные практические задачи вычислительной математики. Как пишет Н. Н. Воробьев: «...к построению такой теории приводит и необходимость обобщения ряда конкретных фактов, встречавшихся в различных прикладных математических теориях. Достаточно сослаться на статьи А. Г. Лунца [71] и Н. Г. Поварова [72] по теории релейно-контактных схем, а также А. Шимбела [73], Р. Беллмана и У. Каруша [74] и др., исследовавших вопросы путей в графе.» ([65], стр. 42) Подобные вычислительные задачи возникли не только при проектировании и производстве контактных схем, но и во многих других областях народного хозяйства. (см. например работу Л. В. Канторовича [75] 1942 года). Воробьев отмечает, что «идеи экстремального гармонического анализа... в духе динамического программирования разбирает И. В. Романовский [67, 68]» ([65], стр. 42). А. А. Корбут в своих работах [76, 77], проводит дальнейшие исследование введенных в этих работах «экстремальных векторных пространств». С. Н. Самборский в своей работе [78] исследует «вопрос существования нетривиального спектра эндоморфизма над идемпотентным полумодулем», а также описывает асимптотическое поведение при итерациях и сходимость «рядов Неймана», проявляющихся при решении уравнений $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{y} \oplus \boldsymbol{f}$. В современном виде идемпотентный анализ был разработан научным коллективом под руководством академика В. П. Маслова [79–84] в восьмидесятых годах двадцатого века в Москве [85]. К настоящему времени существуют два основных направления развития тропической математики: использование чисто алгебраических методов, использующихся, например, в работах [7, 8, 10, 20, 26, 29, 55, 86–88], и геометрический подход, основанный на использовании методов тропической геометрии [27, 28, 89–92].

2.1.2 Идемпотентное полуполе

Пусть множество $\mathbb{X}$ замкнуто относительно ассоциативных и коммутативных операций сложения $\oplus$ и умножения $\otimes$. Будем считать, что $\mathbb{X}$ содержит их нейтральные элементы ноль $\mathbb{0}$ и единицу $\mathbb{1}$. Для сложения выполняется свойство идемпотентности (для любого $x \in \mathbb{X}$ выполняется равенство $x \oplus x = x$), а умножение дистрибутивно относительно сложения и обратимо (для любого ненулевого x существует элемент x^{-1} такой, что $x \otimes x^{-1} = \mathbb{1}$). Так как $\mathbb{X}_+ = \mathbb{X} \setminus \{\mathbb{0}\}$ образует группу по умножению, алгебраическую структуру $\langle \mathbb{X}, \mathbb{0}, \mathbb{1}, \oplus, \otimes \rangle$ будем называть идемпотентным полуполем.

При помощи идемпотентного сложения определим следующий частичный порядок: $x \leq y$ тогда и только тогда, когда $x \oplus y = y$. В терминах предложенного частичного порядка для сложения и умножения выполняется свойство монотонности, при котором в случае выполнения условия $x \leq y$ для любого $z \in \mathbb{X}$ справедливы неравенства $x \oplus z \leq y \oplus z$ и $xz \leq yz$ для любых $x, y \in \mathbb{X}$. Для обращения выполняется свойство антитонности, что означает, что для всех $x, y \neq \mathbb{0}$ из неравенства $x \leq y$ следует неравенство $x^{-1} \geq y^{-1}$. Сложение обладает экстремальным свойством, состоящим в том, что неравенства $x \leq x \oplus y$ и $y \leq x \oplus y$ выполняются для всех $x, y \in \mathbb{X}$. Кроме того, неравенство $x \oplus y \leq z$ справедливо тогда и только тогда, когда выполняются неравенства $x \leq z$ и $y \leq z$. Будем считаем, что представленный частичный порядок дополнен до линейного порядка.

Для каждого $x \in \mathbb{X} \setminus \{\mathbb{0}\}$ определим целую положительную степень p следующим образом $x^0 = \mathbb{1}$, $x^p = x^{p-1}x$, $x^{-p} = (x^{-1})^p$, $\mathbb{0}^p = \mathbb{0}$. Считаем, что операция возведения в целую степень может быть естественным образом распространена на случай рационального показателя степени.

В качестве примеров идемпотентного полуполя $\langle \mathbb{X}, \mathbb{0}, \mathbb{1}, \oplus, \otimes \rangle$ можно рассматривать вещественные полуполя

$$\begin{aligned} &\mathbb{R}_{\max,+} = \langle \mathbb{R} \cup \{-\infty\}, -\infty, 0, \max, + \rangle, && \mathbb{R}_{\min,+} = \langle \mathbb{R} \cup \{+\infty\}, +\infty, 0, \min, + \rangle, \\ &\mathbb{R}_{\max,\times} = \langle \mathbb{R}_+ \cup \{0\}, 0, 1, \max, \times \rangle, && \mathbb{R}_{\min,\times} = \langle \mathbb{R}_+ \cup \{+\infty\}, +\infty, 1, \min, \times \rangle, \end{aligned} \tag{2.1}$$

где $\mathbb{R}$ – множество вещественных чисел, $\mathbb{R}_+ = \{x \in \mathbb{R} | x > 0\}$.

Для полуполя $\mathbb{R}_{\max,+}$ определим ноль $\mathbb{0}$ как $-\infty$, а единицу $\mathbb{1}$ как 0. Для

каждого элемента $x \in \mathbb{R}$ существует обратный элемент x^{-1}, которому соответствует $-x$ в стандартных обозначениях. Для каждого $x, y \in \mathbb{R}$, степень x^y совпадает с арифметическим произведением xy. Отношение частичного порядка, задаваемое идемпотентным сложением, совпадает с обычным отношением линейного порядка на $\mathbb{R}$.

Определим биноминальное тождество как

$$(x \oplus y)^\alpha = x^\alpha \oplus y^\alpha \tag{2.2}$$

для всех $x, y \in \mathbb{X}$ и рационального числа $\alpha \geq 0$.

Для каждого $x_1 \dots x_k \in \mathbb{X}$ справедлив тропический аналог неравенства между средним геометрическим и средним арифметическим

$$(x_1 \dots x_k)^{1/k} \leq x_1 \oplus \dots \oplus x_k. \tag{2.3}$$

2.1.3 Матрицы и векторы

Обозначим как $\mathbb{X}^{m \times n}$ множество матриц, которое состоит из m строк и n столбцов с элементами из $\mathbb{X}$. Определим сложение, умножение и умножение на скаляр для согласованных по размеру матриц $\boldsymbol{A} = (a_{ij})$, $\boldsymbol{B} = (b_{ij})$, $\boldsymbol{C} = (c_{ij})$ при помощи следующих формул:

$$\{\boldsymbol{A} \oplus \boldsymbol{B}\}_{ij} = a_{ij} \oplus b_{ij}, \quad \{\boldsymbol{B} \otimes \boldsymbol{C}\}_{ij} = \bigoplus_k b_{ik} c_{kj}, \quad \{x\boldsymbol{A}\}_{ij} = x a_{ij}. \tag{2.4}$$

Будем считать, что отношение порядка и связанные с ним свойства обобщаются на матрицы и понимаются покомпонентно.

Нулевой в $\mathbb{X}^{m \times n}$ считаем матрицу со всеми элементами, равными $\mathbb{0}$.

Матрица, не имеющая нулевых строк (столбцов), является регулярной по строкам (столбцам). Регулярной будем называть матрицу, которая регулярна и по строкам, и по столбцам.

Как обычно, для любой матрицы $\boldsymbol{A}$ ее транспонированная матрица обозначается как $\boldsymbol{A}^T$.

Для любой матрицы $\boldsymbol{A} = (a_{ij}) \in \mathbb{X}^{m \times n}$ определим мультипликативно сопряженную матрицу $\boldsymbol{A}^- = (a_{ij}^-) \in \mathbb{X}^{n \times m}$, где $a_{ij}^- = a_{ji}^{-1}$, если $a_{ji} \neq \mathbb{0}$ и $a_{ij}^- = \mathbb{0}$ иначе.

Множество векторов-столбцов размерности n обозначим через $\mathbb{X}^n$.

Вектор со всеми компоненами равными $\mathbb{0}$ называется нулевым и обозначается через $\mathbf{0} = (\mathbb{0}, \ldots, \mathbb{0})^T$. Вектор без нулевых компонент является регулярным.

Обозначим через $\mathbf{1} = (\mathbb{1}, \ldots, \mathbb{1})^T$ вектор, состоящий из единиц.

Определим мультипликативно сопряженный вектор для каждого ненулевого вектора $\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \mathbb{X}^n$ как строчный вектор $\boldsymbol{x}^- = (x_1^-, \ldots, x_n^-)$, где $x_i^- = x_i^{-1}$ если $x_i \neq \mathbb{0}$, и $x_i^- = \mathbb{0}$ иначе.

Вектор $\boldsymbol{y} \in \mathbb{X}^n$ является коллинеарным для вектора $\boldsymbol{x} \in \mathbb{X}^n$, если существует скаляр $c \in \mathbb{X}$ такой, что $\boldsymbol{y} = c\boldsymbol{x}$.

Определим идемпотентные аналоги векторной и матричной норм для любого вектора $\boldsymbol{x} \in \mathbb{X}^n$ и матрицы $\boldsymbol{A} \in \mathbb{X}^{m\times n}$:

$$||\boldsymbol{x}|| = \bigoplus_{i=1}^{n} x_i, \quad ||\boldsymbol{A}|| = \bigoplus_{i=1}^{m}\bigoplus_{j=1}^{n} a_{ij}. \tag{2.5}$$

Опишем квадратные матрицы в $\mathbb{X}^{n\times n}$. Будем называть единичной матрица $\boldsymbol{I}$ с элементами, равными $\mathbb{1}$ на главной диагонали и $\mathbb{0}$ вне ее.

Определим неотрицательную целую степень p для квадратной матрицы $\boldsymbol{A} = (a_{ij})$ следующим образом:

$$\boldsymbol{A}^0 = \boldsymbol{I}, \quad \boldsymbol{A}^p = \boldsymbol{A}^{p-1}\boldsymbol{A}. \tag{2.6}$$

Степень суммы матриц может быть представлена при помощи выражения

$$(\boldsymbol{A} \oplus \boldsymbol{B})^p = \bigoplus_{k=1}^{p} \bigoplus_{i_0+\cdots+i_k=p-k} \boldsymbol{B}^{i_0}(\boldsymbol{A}\boldsymbol{B}^{i_1}\cdots\boldsymbol{A}\boldsymbol{B}^{i_k}) \oplus \boldsymbol{B}^p. \tag{2.7}$$

Для любой матрицы $\boldsymbol{A} \in \mathbb{X}^{n\times n}$ определим тропические аналоги определителя и спектрального радиуса матрицы, которые вычисляются соответственно по формулам

$$\operatorname{tr}\boldsymbol{A} = a_{11} \oplus \cdots \oplus a_{nn}, \quad \operatorname{Tr}(\boldsymbol{A}) = \operatorname{tr}\boldsymbol{A} \oplus \cdots \oplus \operatorname{tr}\boldsymbol{A}^n, \quad \lambda = \operatorname{tr}\boldsymbol{A} \oplus \cdots \oplus \operatorname{tr}^{1/n}(\boldsymbol{A}^n). \tag{2.8}$$

Величина $\operatorname{tr} \boldsymbol{A}$, которая называется следом матрицы, обладает следующими свойствами:

$$\operatorname{tr}(\boldsymbol{A} \oplus \boldsymbol{B}) = \operatorname{tr} \boldsymbol{A} \oplus \operatorname{tr} \boldsymbol{B}, \qquad \operatorname{tr}(\boldsymbol{AB}) = \operatorname{tr}(\boldsymbol{BA}), \qquad \operatorname{tr}(x\boldsymbol{A}) = x \operatorname{tr} \boldsymbol{A}. \quad (2.9)$$

В случае выполнения условия $\operatorname{Tr}(\boldsymbol{A}) \leq \mathbb{1}$ определена матрица Клини

$$\boldsymbol{A}^* = \boldsymbol{I} \oplus \boldsymbol{A} \oplus \cdots \oplus \boldsymbol{A}^{n-1}. \quad (2.10)$$

Легко видеть, что из выражения (2.7) следует равенство для матрицы Клини от суммы матриц

$$(\boldsymbol{A} \oplus \boldsymbol{B})^* = \bigoplus_{k=1}^{n-1} \bigoplus_{m=0}^{n-k-1} \bigoplus_{i_0+i_1+\cdots+i_k=m} \boldsymbol{B}^{i_0}(\boldsymbol{AB}^{i_1} \cdots \boldsymbol{AB}^{i_k}) \oplus \boldsymbol{B}^*. \quad (2.11)$$

Матрица называется разложимой, если ее можно привести к блочно-треугольной форме при помощи перестановки строк вместе с такой же перестановкой столбцов. В противном случае матрица является неразложимой.

Обратим внимание, что матрица с регулярными столбцами (строками) имеет только ненулевые элементы и потому является неразложимой. Для каждой неразложимой матрицы ее матрица Клини не имеет нулевых элементов.

Любой матрице $\boldsymbol{A} = (a_{ij}) \in \mathbb{X}^{n \times n}$ может быть сопоставлен ориентированный граф $\langle V, E \rangle$ с множеством вершин $V = \{1, \ldots, n\}$ и множеством дуг $E = \{(i, j) | i, j \in V\}$. Для каждой пары вершин $i, j \in V$ множество E включает дугу (i, j), если $a_{ij} > \mathbb{0}$, и не включает такую дугу иначе.

Матрица неразложима тогда и только тогда, когда ее граф сильно связный.

2.1.4 Решение векторных неравенств

Описанные в данном параграфе неравенства в дальнейшем будут использоваться в качестве неравенств-ограничений в задачах тропической оптимизации, а представленные теоремы будут использоваться для решения задач тропической оптимизации.

Пусть заданы матрица $\boldsymbol{A} \in \mathbb{X}^{m \times n}$ и вектор $\boldsymbol{b} \in \mathbb{X}^m$. Запишем задачу решения относительно неизвестного вектора $\boldsymbol{x} \in \mathbb{X}^n$ неравенства

$$\boldsymbol{Ax} \leq \boldsymbol{b}. \quad (2.12)$$

Данная задача может быть решена при помощи следующей теоремы:

Теорема 1 ([7]). *Для регулярной по столбцам матрицы* $\boldsymbol{A}$ *и регулярного вектора* $\boldsymbol{b}$*, вектор* $\boldsymbol{x}$ *является решением неравенства* (2.12) *тогда и только тогда, когда*

$$\boldsymbol{x} \leq (\boldsymbol{b}^{-}\boldsymbol{A})^{-}. \tag{2.13}$$

Для любой матрицы $\boldsymbol{A} \in \mathbb{X}^{n\times n}$ неравенство относительно неизвестного вектора $\boldsymbol{x} \in \mathbb{X}^n$

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{x} \tag{2.14}$$

будем называть линейным однородным неравенством. Его решение можно найти при помощи следующего результата

Теорема 2 ([36]). *Пусть* $\boldsymbol{x}$ *– общее решение однородного неравенства.*

Тогда, если $\mathrm{Tr}(\boldsymbol{A}) \leq \mathbb{1}$*, то* $\boldsymbol{x} = \boldsymbol{A}^{*}\boldsymbol{u}$ *для всех регулярных векторов* $\boldsymbol{u}$*. В противном случае регулярных решений нет.*

Предыдущую задачу можно усложнить, добавив вектора $\boldsymbol{b} \in \mathbb{X}^n$. Полученное неравенство относительно неизвестного вектора $\boldsymbol{x} \in \mathbb{X}^n$

$$\boldsymbol{A}\boldsymbol{x} \oplus \boldsymbol{b} \leq \boldsymbol{x} \tag{2.15}$$

будем называть линейным неоднородным неравенством. Сформулируем теорему, предлагающую решение такого неравенства.

Теорема 3 ([20]). *Пусть* $\boldsymbol{x} \in \mathbb{X}^n$ *– общее регулярное решения неравенства* (2.15)*. Тогда справедливы следующие утверждения:*

1. *Если* $\mathrm{Tr}(\boldsymbol{A}) \leq \mathbb{1}$*, то* $\boldsymbol{x} = \boldsymbol{A}^{*}\boldsymbol{u}$ *для любого регулярного* $\boldsymbol{u}$ *такого, что* $\boldsymbol{u} \geq \boldsymbol{b}$*.*

2. *Если* $\mathrm{Tr}(\boldsymbol{A}) > \mathbb{1}$*, то регулярных решений не существует.*

Обратим внимание, что представленные в разделе 1.2 ограничения на временные сроки выполнения работ проекта являются рассмотренными выше неравенствами, записанными в терминах полуполя $\mathbb{R}_{\max,+}$, что может быть использовано при решении задач составления оптимальных планов.

2.2 Задачи тропической оптимизации

Задачи тропической оптимизации (задачи оптимизации, сформулированные на языке идемпотентной математики) обычно задаются в терминах минимизации (максимизации) функций на конечномерных полумодулях над идемпотентными полуполями при ограничениях в виде линейных уравнений и неравенств.

Такие задачи находят широкое применение при решении многих практических задач. Например в работах [20, 21, 23, 31, 33, 36, 37, 52, 53, 93–95] приводится решение задач сетевого планирования посредством сведения их к задачам тропической оптимизации. Основным достоинством подобного подхода является то, что по сравнению с методами математического программирования, которые зачастую предоставляют лишь алгоритмическое решение, методы тропической оптимизации дают возможность получить в явном виде полное решение.

Ниже рассматриваются задачи с нелинейной целевой функцией, так как значительная часть задач сетевого планирования относятся к этому классу задач. Информацию о задачах оптимизации с линейными целевыми функциями можно найти в работах [12, 21, 26, 31, 53, 95–102]. В разделе 2.2.1 приведен краткий обзор различных задач оптимизации, начиная с работ [52, 53, 93], где задачи оптимизации рассматривались как поиск вектора, обеспечивающего наилучшую оценку в чебышевской норме ($\|\boldsymbol{x}\| = \max_{1\leq i\leq n}(x_i)$). В работе [94] вводится понятие интервальной полунормы, которая является частным случаем рассматриваемых в работах [36, 37] целевых функций, использующихся для решения задач управления временными графиками выполнения проектов. Далее, в параграфе 2.2.2 более подробно рассмотрены задачи тропической оптимизации, которые составляют основу для поиска решений задач тропической оптимизации с ограничениями.

2.2.1 Примеры задач тропической оптимизации

Сперва рассмотрим следующую задачу

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & (\boldsymbol{A}\boldsymbol{x})^{-} \boldsymbol{d}, \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{d}. \end{aligned} \tag{2.16}$$

Данная задача может быть рассмотрена как поиск вектора $\boldsymbol{x}$, обеспечивающего наилучшую нижнюю оценку для вектора $\boldsymbol{d}$ при помощи $\boldsymbol{A}\boldsymbol{x}$ в чебышевской норме. В работе [52] представлено полное явное решение, основанное на применении абстрактной теории линейных операторов над полуполем $\mathbb{R}_{\max,+}$. Подобное решение также рассматривается в работе [53].

Следующие две задачи, первоначально сформулированные в обычной форме, могут быть представлены на языке идемпотентной алгебры в виде задач тропической оптимизации с нелинейной целевой функцией и линейными ограничениями. В статье [93] исследуется задача минимизации в $\mathbb{R}_{\max,+}$ функции расстояния чебышевского типа, которая решается при помощи полиномиального алгоритма порогового типа. Эта задача может быть записана в форме задачи оптимизации с нелинейной целевой функцией и ограничениями, заданными двусторонним неравенством

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & (\boldsymbol{A}\boldsymbol{x})^{-} \boldsymbol{p} \oplus \boldsymbol{q}^{-} \boldsymbol{A}\boldsymbol{x}, \\ & \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{2.17}$$

Интервальной полунормой называется максимальный разброс между компонентами вектора. Задача минимизации интервальной полунормы рассматривается, например, в работе [94], где предлагается явное решение в смешанной аналитической форме, записанной в терминах полуполя $\mathbb{R}_{\max,+}$ и его двойственного полуполя $\mathbb{R}_{\min,+}$. Заметим, что интервальная полунорма может быть легко представлена в терминах $\mathbb{R}_{\max,+}$ с помощью мультипликативно сопряженных векторов:

$$\min \quad \mathbf{1}^{T} \boldsymbol{A}\boldsymbol{x} \, (\boldsymbol{A}\boldsymbol{x})^{-} \mathbf{1}. \tag{2.18}$$

При помощи описанной в работе [103] связи между решениями двусторонних уравнений (в которых неизвестный вектор присутствует в обоих частях

равенства) в работе [87] для следующей задачи с нелинейной целевой функцией

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{p}^T \boldsymbol{x} \left(\boldsymbol{q}^T \boldsymbol{x}\right)^{-1}, \\ & \boldsymbol{A}\boldsymbol{x} \oplus \boldsymbol{b} \leq \boldsymbol{C}\boldsymbol{x} \oplus \boldsymbol{d}, \end{aligned} \tag{2.19}$$

представлена итерационная вычислительная схема поиска решения в $\mathbb{R}_{\max,+}$.

Представим краткий обзор задач, представленных в терминах общего линейно упорядоченного полуполя, допускающих прямые явные решения в векторной форме. Аналитический метод решения подобных задач, основанный на новых результатах тропической спектральной теории и решении линейных тропических неравенств, был предложен в работах [7, 104, 105].

В статье [36] разработано полное решение задачи без ограничений

$$\min_{\boldsymbol{x}} \quad \boldsymbol{d}^{-} \boldsymbol{A}\boldsymbol{x} \oplus (\boldsymbol{A}\boldsymbol{x})^{-} \boldsymbol{d}. \tag{2.20}$$

В работе [106] в свою очередь представлено явное решение другой задачи без ограничений в форме

$$\min_{\boldsymbol{x}} \quad (\boldsymbol{A}\boldsymbol{x})^{-} \boldsymbol{p} \oplus \boldsymbol{q}^{-} \boldsymbol{A}\boldsymbol{x}. \tag{2.21}$$

Кроме того доказано, что две задачи с ограничениями в виде линейных неравенств

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-} \boldsymbol{p} \oplus \boldsymbol{q}^{-} \boldsymbol{x}, \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{x}; \end{aligned} \tag{2.22}$$

и равенств

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-} \boldsymbol{p} \oplus \boldsymbol{q}^{-} \boldsymbol{x}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}; \end{aligned} \tag{2.23}$$

могут быть сведены к предыдущей задаче без ограничений, что позволяет получить их прямые явные решения.

Наконец в работе [37] предлагается полное, записанное в явном виде решение задачи

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-} \boldsymbol{A}\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x} \end{aligned} \tag{2.24}$$

при условии, что как минимум одна из матриц $\boldsymbol{A}$ и $\boldsymbol{B}$ неразложима.

Описанные в текущем параграфе задачи составляют основу для представленных далее результатов тропической оптимизации, которые во многом являются развитием и усложнением этих задач.

2.2.2 Примеры решения некоторых задач тропической оптимизации

Предположим, что заданы матрица $\boldsymbol{A} \in \mathbb{X}^{m\times n}$ и векторы $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{X}^m$. Необходимо вычислить регулярные векторы $\boldsymbol{x} \in \mathbb{X}^n$, для которых достигается минимум в задаче

$$\min_{\boldsymbol{x}} \quad \boldsymbol{q}^{-}\boldsymbol{x}(\boldsymbol{A}\boldsymbol{x})^{-}\boldsymbol{p}. \tag{2.25}$$

В статье [37] найдено следующее решение задачи (2.25).

Теорема 4. *Предположим, что матрица $\boldsymbol{A}$ – регулярная по строкам, вектор $\boldsymbol{p}$ – ненулевой, а $\boldsymbol{q}$ – регулярный. Тогда в задаче* (2.25) *минимум равен*

$$\Delta = (\boldsymbol{A}\boldsymbol{q})^{-}\boldsymbol{p} \tag{2.26}$$

и достигается на любом векторе

$$\boldsymbol{x} = \alpha\boldsymbol{q}, \qquad \alpha > \mathbb{0}. \tag{2.27}$$

Заметим, что задача минимизации интервальной полунормы (2.18), представленная в предыдущем параграфе, является частным случаем задачи (2.25).

Пусть даны матрицы $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{X}^{n\times n}$ и векторы $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{X}^n$. Требуется найти регулярные векторы $\boldsymbol{x} \in \mathbb{X}^n$, которые решают следующую задачу

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{A}\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}. \end{aligned} \tag{2.28}$$

Определим вспомогательные обозначения

$$\boldsymbol{S}_k = \bigoplus_{0\le i_1+\cdots+i_k\le n-k} \boldsymbol{A}\boldsymbol{B}^{i_1}\cdots\boldsymbol{A}\boldsymbol{B}^{i_k}, \qquad k = 1,\ldots,n; \tag{2.29}$$

$$\boldsymbol{T}_k = \bigoplus_{0\le i_0+i_1+\cdots+i_k\le n-k-1} \boldsymbol{B}^{i_0}(\boldsymbol{A}\boldsymbol{B}^{i_1}\cdots\boldsymbol{A}\boldsymbol{B}^{i_k}), \qquad k = 1,\ldots,n-1. \tag{2.30}$$

В статье [35] предложено следующее решение задачи (2.28).

Теорема 5. *Пусть* $\boldsymbol{A}$ *— матрица с неотрицательным спектральным радиусом* $\lambda > \mathbb{0}$*, а* $\boldsymbol{B}$ *— матрица такая, что* $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$*. Пусть* $\boldsymbol{g}$ *— вектор, а* $\boldsymbol{h}$ *— регулярный вектор такой, что* $\boldsymbol{h}^{-}\boldsymbol{B}^{*}\boldsymbol{g} \leq \mathbb{1}$*.*

Тогда минимальное значение в задаче (2.28) *равно*

$$\theta = \bigoplus_{k=1}^{n} \mathrm{tr}^{1/k}(\boldsymbol{S}_k) \oplus \bigoplus_{k=1}^{n-1} (\boldsymbol{h}^{-}\boldsymbol{T}_k\boldsymbol{g})^{1/k}, \tag{2.31}$$

а все регулярные решения имеют вид

$$\boldsymbol{x} = (\theta^{-1}\boldsymbol{A} \oplus \boldsymbol{B})^{*}\boldsymbol{u}, \tag{2.32}$$

где $\boldsymbol{u}$ *— любой регулярный вектор, который удовлетворяет условию*

$$\boldsymbol{g} \leq \boldsymbol{u} \leq (\boldsymbol{h}^{-}(\theta^{-1}\boldsymbol{A} \oplus \boldsymbol{B})^{*})^{-}. \tag{2.33}$$

Обратим внимание, что задача (2.24) является частным случаем задачи (2.28) с ослабленными ограничениями.

Пусть даны матрица $\boldsymbol{B}$ и векторы $\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{p}$ и $\boldsymbol{q}$. Рассмотрим следующую задачу

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{2.34}$$

Сформулируем теорему, предлагающую решение данной задачи

Теорема 6 ([24])**.** *Пусть* $\boldsymbol{B}$ *– матрица, удовлетворяющая условию* $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$*. Пусть* $\boldsymbol{g}$ *– вектор, а* $\boldsymbol{q}$ *и* $\boldsymbol{h}$ *– регулярные векторы такие, что* $\boldsymbol{h}^{-}\boldsymbol{B}^{*}\boldsymbol{g} \leq \mathbb{1}$*.*

Тогда минимальное значение целевой функции в задаче (2.34) *равно*

$$\theta = (\boldsymbol{q}^{-}\boldsymbol{B}^{*}\boldsymbol{p})^{1/2} \oplus \boldsymbol{h}^{-}\boldsymbol{B}^{*}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{B}^{*}\boldsymbol{g}, \tag{2.35}$$

а все регулярные решения имеют вид

$$\boldsymbol{x} = \boldsymbol{B}^{*}\boldsymbol{u} \tag{2.36}$$

где $\boldsymbol{u}$ *– любой регулярный вектор, который удовлетворяет условиям*

$$\boldsymbol{g} \oplus \theta^{-1}\boldsymbol{p} \leq \boldsymbol{u} \leq ((\boldsymbol{h}^{-} \oplus \theta^{-1}\boldsymbol{q}^{-})\boldsymbol{B}^{*})^{-}. \tag{2.37}$$

Представим результат статьи [33], предлагающий решение для следующей задачи оптимизации. Пусть заданы матрицы $\boldsymbol{A} \in \mathbb{X}^{n\times n}$ и векторы $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{X}^n$. Необходимо найти регулярное решение $\boldsymbol{x} \in \mathbb{X}^n$ задачи

$$\max_{\boldsymbol{x}} \ \boldsymbol{q}^{-}\boldsymbol{x}\,(\boldsymbol{A}\boldsymbol{x})^{-}\boldsymbol{p}. \tag{2.38}$$

Сформулируем теорему, предлагающую прямое полное решение задачи (2.38).

Теорема 7. *Пусть матрица* $\boldsymbol{A} = (\boldsymbol{a_j})$ *имеет только регулярные столбцы* $\boldsymbol{a}_j = (a_{ij})$, *а векторы* $\boldsymbol{p} = (p_j)$ *и* $\boldsymbol{q} = (q_j)$ *регулярные.*

Тогда максимум в задаче (2.38) *равен*

$$\Delta = \boldsymbol{q}^{-}\boldsymbol{A}^{-}\boldsymbol{p} \tag{2.39}$$

и достигается тогда и только тогда, когда вектор $\boldsymbol{x} = (x_j)$ *имеет компоненты*

$$x_k = \alpha a_{sk}^{-1} p_s, \quad x_j \le \alpha a_{sj}^{-1} p_s, \quad j \ne k, \tag{2.40}$$

для всех $\alpha > \mathbb{0}$ *и индексов* k *и* s, *определяемых условиями*

$$k = \arg\max_{1\le j\le n} q_j^{-1}\boldsymbol{a}_j^{-}\boldsymbol{p}, \quad s = \arg\max_{1\le j\le n} a_{jk}^{-1} p_j. \tag{2.41}$$

Заметим, что максимизация интервальной полунормы является частным случаем задачи (2.38).

Глава 3

Минимизация продолжительности проекта

В данной главе сформулирована и решена задача составления оптимального календарного графика выполнения работ проекта, которая заключается в минимизации общей продолжительности проекта с заданными ограничениями на последовательность выполнения работ. Предложенный метод решения заключается в представлении задачи планирования на языке тропической математики и последующем ее решение при помощи методов тропической оптимизации. В разделе 3.2 сформулирована задача оптимального управления и приведено ее решение, основанное на представлении задачи в виде задачи тропической оптимизации. Наконец, в разделе 3.3 представлен пример практического использования полученного результата для решения задачи о наиболее быстром проведении вакцинации.

3.1 Решение задачи тропической оптимизации с ограничениями

Предположим, что заданы матрица $\boldsymbol{B} \in \mathbb{X}^{n\times n}$ и векторы $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{g}, \boldsymbol{h} \in \mathbb{X}^n$. Изучим следующую задачу тропической оптимизации:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{p}\boldsymbol{q}^T\boldsymbol{x}; \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}. \end{aligned} \tag{3.1}$$

Для поиска решения задачи будет использован результат теоремы 5. Полученное решение затем, с учетом особой формы целевой функции задачи, будет приведено к более компактному виду. Сформулируем и докажем следующее.

Теорема 8. *Пусть $\boldsymbol{B}$ – матрица, для которой $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$, $\boldsymbol{g}$ – вектор, а $\boldsymbol{h}$ – регулярный вектор такие, что выполняется неравенство $\boldsymbol{h}^{-}\boldsymbol{B}^{*}\boldsymbol{g} \le \mathbb{1}$. Считаем, что векторы $\boldsymbol{p}$ и $\boldsymbol{q}$ – ненулевые, такие, что $\boldsymbol{q}^T\boldsymbol{p} > \mathbb{0}$.*

Тогда минимальное значение целевой функции в задаче (3.1) *равно*

$$\theta = \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p} \oplus \bigoplus_{0 \le i+j \le n-2} (\boldsymbol{h}^- \boldsymbol{B}^i \boldsymbol{p})(\boldsymbol{q}^T \boldsymbol{B}^j \boldsymbol{g}), \tag{3.2}$$

а все регулярные решения имеют вид

$$\boldsymbol{x} = \boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{G} = \boldsymbol{B}^* \oplus \bigoplus_{0 \le i+j \le n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p} \boldsymbol{q}^T \boldsymbol{B}^j, \tag{3.3}$$

где $\boldsymbol{u}$ *– любой регулярный вектор, который удовлетворяет условиям*

$$\boldsymbol{g} \le \boldsymbol{u} \le (\boldsymbol{h}^- \boldsymbol{G})^-. \tag{3.4}$$

Доказательство. Поскольку $\boldsymbol{q}^T \boldsymbol{p} > \mathbb{0}$ спектральный радиус матрицы $\boldsymbol{p}\boldsymbol{q}^T$ ненулевой. Так как задача (3.1) имеет вид (2.28), воспользуемся теоремой 5. Сперва по формуле (2.31) вычислим минимум θ. Подставляя равенство $\boldsymbol{A} = \boldsymbol{p}\boldsymbol{q}^T$ в выражение $\boldsymbol{A}\boldsymbol{B}^{i_1} \cdots \boldsymbol{A}\boldsymbol{B}^{i_k}$ и выписывая скалярные множители, получим следующее равенство

$$\boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_1} \cdots \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k} = (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k}. \tag{3.5}$$

Таким образом равенство (2.29) принимает вид

$$\boldsymbol{S}_k = \bigoplus_{0 \le i_1 + \cdots + i_k \le n-k} (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k}. \tag{3.6}$$

Вычислим след матрицы под знаком суммы. Учитывая свойства следа имеем

$$\operatorname{tr}((\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k}) = \boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_k} \boldsymbol{p}. \tag{3.7}$$

При помощи полученного результата запишем сумму

$$\bigoplus_{k=1}^{n} \operatorname{tr}^{1/k}(\boldsymbol{S}_k) = \bigoplus_{k=1}^{n} \bigoplus_{0 \le i_1 + \cdots + i_k \le n-k} (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_k} \boldsymbol{p})^{1/k}. \tag{3.8}$$

Рассмотрим первое слагаемое, соответствующее $k = 1$ в суммах слева и справа, более подробно

$$\operatorname{tr}(\boldsymbol{S}_1) = \bigoplus_{0 \le i \le n-1} \boldsymbol{q}^T \boldsymbol{B}^i \boldsymbol{p} = \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p}. \tag{3.9}$$

Докажем, что другие слагаемые не превосходят первое. В самом деле, воспользовавшись неравенством между геометрическим и арифметическим средними, получим

$$(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{p})^{1/k} \leq \boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\oplus\cdots\oplus\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{p}, \tag{3.10}$$

следовательно при всех $k = 2,\ldots,n$ справедливо

$$\bigoplus_{0\leq i_1+\cdots+i_k\leq n-k}(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{p})^{1/k} \leq \bigoplus_{i=0}^{n-k}\boldsymbol{q}^T\boldsymbol{B}^{i}\boldsymbol{p} \leq \bigoplus_{i=0}^{n-1}\boldsymbol{q}^T\boldsymbol{B}^{i}\boldsymbol{p} = \operatorname{tr}(\boldsymbol{S}_1). \tag{3.11}$$

Поскольку первое слагаемое доминирует над остальными имеем

$$\bigoplus_{k=1}^{n}\operatorname{tr}^{1/k}(\boldsymbol{S}_k) = \operatorname{tr}(\boldsymbol{S}_1) = \boldsymbol{q}^T\boldsymbol{B}^*\boldsymbol{p}. \tag{3.12}$$

Затем, воспользовавшись равенством (3.5), представим (2.30) в следующем виде

$$\boldsymbol{T}_k = \bigoplus_{0\leq i_0+i_1+\cdots+i_k\leq n-k-1}(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})\boldsymbol{B}^{i_0}\boldsymbol{p}\boldsymbol{q}^T\boldsymbol{B}^{i_k}. \tag{3.13}$$

Умножив матрицы под знаком суммы слева на $\boldsymbol{h}^-$ и справа на $\boldsymbol{g}$ получим равенство

$$\begin{aligned}\boldsymbol{h}^-(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})\boldsymbol{B}^{i_0}\boldsymbol{p}\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g} = \\ = (\boldsymbol{h}^-\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}).\end{aligned} \tag{3.14}$$

С учетом полученного равенства выпишем сумму

$$\begin{aligned}&\bigoplus_{k=1}^{n-1}(\boldsymbol{h}^-\boldsymbol{T}_k\boldsymbol{g})^{1/k} = \\ &= \bigoplus_{k=1}^{n-1}\bigoplus_{0\leq i_0+i_1+\cdots+i_k\leq n-k-1}((\boldsymbol{h}^-\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}))^{1/k}\end{aligned} \tag{3.15}$$

и рассмотрим первое слагаемое, которое соответствует $k = 1$ и имеет вид

$$\boldsymbol{h}^-\boldsymbol{T}_1\boldsymbol{g} = \bigoplus_{0\leq i_0+i_1\leq n-2}(\boldsymbol{h}^-\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{g}). \tag{3.16}$$

Убедимся, что величина других слагаемых не превосходит сумму $\mathrm{tr}(\boldsymbol{S}_1) \oplus \boldsymbol{h}^{-}\boldsymbol{T}_1\boldsymbol{g}$. Используя неравенство между геометрическим и арифметическим средними, получим

$$((\boldsymbol{h}^{-}\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}))^{1/k} \leq \\ \leq (\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\oplus\cdots\oplus\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})\oplus(\boldsymbol{h}^{-}\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}). \quad (3.17)$$

Таким образом, для всех $k = 2, \ldots, n$ будет справедливо неравенство

$$\bigoplus_{0\leq i_0+i_1+\cdots+i_k\leq n-k-1}((\boldsymbol{h}^{-}\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}))^{1/k} \leq \\ \leq \bigoplus_{i=0}^{n-1}\boldsymbol{q}^T\boldsymbol{B}^{i}\boldsymbol{p}\oplus\bigoplus_{0\leq i+j\leq n-2}(\boldsymbol{h}^{-}\boldsymbol{B}^{i}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{j}\boldsymbol{g}) = \boldsymbol{q}^T\boldsymbol{B}^{*}\boldsymbol{p}\oplus\boldsymbol{h}^{-}\boldsymbol{T}_1\boldsymbol{g}. \quad (3.18)$$

Учитывая последнее соотношения получим двойное неравенство

$$\boldsymbol{h}^{-}\boldsymbol{T}_1\boldsymbol{g} \leq \bigoplus_{k=1}^{n-1}(\boldsymbol{h}^{-}\boldsymbol{T}_k\boldsymbol{g})^{1/k} \leq \boldsymbol{q}^T\boldsymbol{B}^{*}\boldsymbol{p}\oplus\boldsymbol{h}^{-}\boldsymbol{T}_1\boldsymbol{g}. \quad (3.19)$$

Чтобы получить (3.2), осталось объединить полученные результаты:

$$\theta = \bigoplus_{k=1}^{n}\mathrm{tr}^{1/k}(\boldsymbol{S}_k)\oplus\bigoplus_{k=1}^{n-1}(\boldsymbol{h}^{-}\boldsymbol{T}_k\boldsymbol{g})^{1/k} = \boldsymbol{q}^T\boldsymbol{B}^{*}\boldsymbol{p}\oplus\bigoplus_{0\leq i+j\leq n-2}(\boldsymbol{h}^{-}\boldsymbol{B}^{i}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{j}\boldsymbol{g}). \quad (3.20)$$

Основываясь на формулах (3.3) и (3.4) перейдем к параметрическому представлению всего множества решений задачи. Найдем матрицу Клини $(\theta^{-1}\boldsymbol{A}\oplus\boldsymbol{B})^{*} = (\theta^{-1}\boldsymbol{p}\boldsymbol{q}^T\oplus\boldsymbol{B})^{*}$ и применим тождество (2.11). Как и раньше используем равенство (3.5) для преобразования выражения под знаком суммы и получим

$$(\theta^{-1}\boldsymbol{p}\boldsymbol{q}^T\oplus\boldsymbol{B})^{*} = \boldsymbol{B}^{*}\oplus\bigoplus_{k=1}^{n-1}\bigoplus_{m=0}^{n-k-1}\bigoplus_{i_0+\ldots+i_k=m}\theta^{-m}\boldsymbol{B}^{i_0}\boldsymbol{p}\boldsymbol{q}^T\boldsymbol{B}^{i_k}(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p}). \quad (3.21)$$

Рассмотрим сумму в правой части и запишем слагаемое при $k = 1$ в виде

$$\bigoplus_{m=0}^{n-2}\bigoplus_{i_0+i_1=m}\theta^{-1}\boldsymbol{B}^{i_0}\boldsymbol{p}\boldsymbol{q}^T\boldsymbol{B}^{i_1} = \bigoplus_{0\leq i+j\leq n-2}\theta^{-1}\boldsymbol{B}^{i}\boldsymbol{p}\boldsymbol{q}^T\boldsymbol{B}^{j}. \quad (3.22)$$

Заметим, что из полученного выражения (3.2) для минимума целевой функции задачи следует, что для всех $i = 0, \ldots, n-1$ справедливы неравенства $\theta \geq \boldsymbol{q}^T \boldsymbol{B}^i \boldsymbol{p}$ и, более того, так как векторы $\boldsymbol{p}$ и $\boldsymbol{q}$ ненулевые, выполняется $\theta \geq \boldsymbol{q}^T \boldsymbol{I} \boldsymbol{p} > \mathbb{0}$. Учитывая, что тогда для всех i будут выполняться неравенства $\theta^{-1} \boldsymbol{q}^T \boldsymbol{B}^i \boldsymbol{p} \leq \mathbb{1}$, получим неравенство

$$\theta^{-1} \boldsymbol{B}^{i_0} \boldsymbol{p} \boldsymbol{q}^T \boldsymbol{B}^{i_k} (\theta^{-1} \boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \theta^{-1} \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \leq \theta^{-1} \boldsymbol{B}^{i_0} \boldsymbol{p} \boldsymbol{q}^T \boldsymbol{B}^{i_k}. \tag{3.23}$$

С учетом последнего неравенства для всех $k = 2, \ldots, n-1$ справедливо

$$\bigoplus_{m=0}^{n-k-1} \bigoplus_{i_0+\ldots+i_k=m} \theta^{-m} \boldsymbol{B}^{i_0} \boldsymbol{p} \boldsymbol{q}^T \boldsymbol{B}^{i_k} (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \leq \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p} \boldsymbol{q}^T \boldsymbol{B}^j, \tag{3.24}$$

откуда следует, что слагаемое суммы для $k = 1$ превосходит остальные, которые можно отбросить. Таким образом матрица Клини приобретает вид

$$(\theta^{-1} \boldsymbol{p} \boldsymbol{q}^T \oplus \boldsymbol{B})^* = \boldsymbol{B}^* \oplus \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p} \boldsymbol{q}^T \boldsymbol{B}^j = \boldsymbol{G}. \tag{3.25}$$

После подстановки полученного выражения в формулы (2.32) и (2.33) получим параметрическое описание решений задачи в виде (3.3) и (3.4).

Найдем вычислительную сложность решения задачи. Сперва оценим сложность поиска матрицы

$$\boldsymbol{G} = \boldsymbol{B}^* \oplus \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p} \boldsymbol{q}^T \boldsymbol{B}^j, \tag{3.26}$$

который заключается в вычислении матрицы Клини, требующего не более чем $O(n^4)$ скалярных операций, и суммы произведения следующих матриц

$$\bigoplus_{l=0}^{l=n-2} \boldsymbol{M}_l, \quad \boldsymbol{M}_l = \theta^{-1} \boldsymbol{B}^i \boldsymbol{p} \boldsymbol{q}^T \boldsymbol{B}^j, \quad l = i + j. \tag{3.27}$$

вычисление которой заключаетсся в вычислении $1 + \cdots + n - 2 = (n-2)(n-1)/2$ матриц $\boldsymbol{M}_l$.

Для нахождение всех степеней матрицы $\boldsymbol{B}$ потребуется $O(n^4)$ операций. Вычисление векторов $\boldsymbol{B}^i \boldsymbol{p}$ и $\boldsymbol{q}^T \boldsymbol{B}^j$, после нахождения степеней матрицы $\boldsymbol{B}$,

потребует $O(n^2)$ скалярных операций. Для последующего поиска каждой матрицы $\boldsymbol{M_l}$ также необходимо $O(n^2)$ операций. Следовательно потребуется $O(n^4)$ операций для поиска вспомогательных матриц и векторов и $O(n^4)$ операций для последующего вычисления суммы $\bigoplus_{l=0}^{l=n-2} \boldsymbol{M_l}$.

Вычисление минимума

$$\theta = \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p} \oplus \bigoplus_{l=0}^{l=n-2} \psi_l, \quad \psi_l = (\boldsymbol{h}^- \boldsymbol{B}^i \boldsymbol{p})(\boldsymbol{q}^T \boldsymbol{B}^j \boldsymbol{g}), \quad i+j=l. \tag{3.28}$$

заключается в вычислении n степеней матрицы $\boldsymbol{B}$, поиске матрицы Клини, скаляра $\boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p}$ и $1+\cdots+n-2 = (n-2)(n-1)/2$ скаляров ψ_l. Вычисление $n-1$ степеней матрицы и матрицы Клини потребует не более чем $O(n^4)$ операций.

3.2 Минимизация максимальной продолжительности проекта

Исследуем задачу составления оптимального плана работ в соответствии с критерием минимума максимальной продолжительности проекта (1.7) при всех перечисленных выше ограничениях (при полном наборе ограничений). Сформулируем исследуемую задачу более формально.

$$\begin{aligned} \min_{x_i, y_i} \quad & \max_{1\le i\le n} y_i + \max_{1\le i\le n}(-x_i); \\ & \max_{1\le j\le n}(b_{ij}+x_j) \le x_i, \quad \max_{1\le j\le n}(c_{ij}+x_j) = y_i, \\ & \max_{1\le j\le n}(d_{ij}+y_j) \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i=1,\ldots,n. \end{aligned} \tag{3.29}$$

Легко видеть, что задача (3.29) может быть представлена в виде задачи линейного программирования и решена в численном виде с помощью известных вычислительных процедур. Далее, при помощи методов тропической математики, будет построено прямое аналитическое решение рассматриваемой задачи.

Обратим внимание, что в терминах идемпотентного полуполя $\mathbb{R}_{\max,+}$ целевую функцию

$$\max_{1\le i\le n} y_i + \max_{1\le i\le n}(-x_i) \tag{3.30}$$

можно представить следующим образом:

$$\bigoplus_{1\le i\le n} y_i \bigoplus_{1\le j\le n} x_j^{-1}. \tag{3.31}$$

Перепишем ограничения

$$\max_{1\le j\le n}(b_{ij}+x_j)\le x_i,\ \max_{1\le j\le n}(c_{ij}+x_j)=y_i,\ \max_{1\le j\le n}(d_{ij}+y_j)\le x_i,\ i=1,\dots,n, \tag{3.32}$$

в тропической форме

$$\bigoplus_{1\le j\le n} b_{ij}x_j\le x_i,\quad \bigoplus_{1\le j\le n} c_{ij}x_j=y_i,\quad \bigoplus_{1\le j\le n} d_{ij}y_j\le x_i,\quad i=1,\dots,n. \tag{3.33}$$

Принимая во внимание, что в алгебре $\mathbb{R}_{\max,+}$ справедливо обычное отношение порядка, представим задачу (3.29) в тропической форме:

$$\begin{aligned}\min_{x_i,y_i}\ & \bigoplus_{1\le i\le n} y_i \bigoplus_{1\le j\le n} x_j^{-1},\\ & \bigoplus_{1\le j\le n} b_{ij}x_j\le x_i,\quad \bigoplus_{1\le j\le n} c_{ij}x_j=y_i,\\ & \bigoplus_{1\le j\le n} d_{ij}y_j\le x_i,\quad g_i\le x_i\le h_i,\quad y_i\le f_i,\quad i=1,\dots,n.\end{aligned} \tag{3.34}$$

Определим векторные обозначения

$$\begin{gathered}\boldsymbol{B}=(b_{ij}),\quad \boldsymbol{C}=(c_{ij}),\quad \boldsymbol{D}=(d_{ij}),\\ \boldsymbol{x}=(x_i),\quad \boldsymbol{y}=(y_i),\quad \boldsymbol{f}=(f_i),\quad \boldsymbol{g}=(g_i),\quad \boldsymbol{h}=(h_i)\end{gathered} \tag{3.35}$$

и запишем задачу в матрично-векторном виде

$$\begin{aligned}\min_{\boldsymbol{x},\boldsymbol{y}}\ & \boldsymbol{x}^{-}\mathbf{1}\mathbf{1}^T\boldsymbol{y},\\ & \boldsymbol{B}\boldsymbol{x}\le\boldsymbol{x},\quad \boldsymbol{C}\boldsymbol{x}=\boldsymbol{y},\\ & \boldsymbol{D}\boldsymbol{y}\le\boldsymbol{x},\quad \boldsymbol{g}\le\boldsymbol{x}\le\boldsymbol{h},\quad \boldsymbol{y}\le\boldsymbol{f}.\end{aligned} \tag{3.36}$$

Решим задачу, воспользовавшись результатом теоремы 8.

Теорема 9. *Пусть $\boldsymbol{B}$ и $\boldsymbol{D}$ – матрицы, $\boldsymbol{C}$ – регулярная по столбцам матрица такие, что матрица $\boldsymbol{R}=\boldsymbol{B}\oplus\boldsymbol{D}\boldsymbol{C}$ удовлетворяет условию $\mathrm{Tr}(\boldsymbol{R})\le\mathbb{1}$. Пусть*

$\boldsymbol{g}$ – *вектор, а* $\boldsymbol{f}$ *и* $\boldsymbol{h}$ – *регулярные векторы такие, что вектор* $\boldsymbol{s}^T = \boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-$ *удовлетворяет условию* $\boldsymbol{s}^T \boldsymbol{R}^* \boldsymbol{g} \leq \mathbb{1}$.

Тогда минимальное значение целевой функции в задаче (3.36) *равно*

$$\theta = \|\boldsymbol{C}\boldsymbol{R}^*\| \oplus \bigoplus_{0 \leq i+j \leq n-2} \|\boldsymbol{s}^T \boldsymbol{R}^i\| \|\boldsymbol{C}\boldsymbol{R}^j \boldsymbol{g}\|, \tag{3.37}$$

а все регулярные решения имеют вид

$$\boldsymbol{x} = \boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{G} = \boldsymbol{R}^* \oplus \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{R}^i \boldsymbol{1}\boldsymbol{1}^T \boldsymbol{C} \boldsymbol{R}^j, \tag{3.38}$$

где $\boldsymbol{u}$ – *любой регулярный вектор, который удовлетворяет условиям*

$$\boldsymbol{g} \leq \boldsymbol{u} \leq (\boldsymbol{s}^T \boldsymbol{G})^-. \tag{3.39}$$

Доказательство. Подставив $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ в задачу получим

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^- \boldsymbol{1}\boldsymbol{1}^T \boldsymbol{C}\boldsymbol{x}, \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{3.40}$$

Воспользовавшись теоремой 1, получим, что $\boldsymbol{x} \leq (\boldsymbol{f}^- \boldsymbol{C})^-$. Учитывая, что обращение обладает свйством антитонности перепишем неравенства, ограничивающие вектор $\boldsymbol{x}$ сверху

$$\boldsymbol{x}^- \geq \boldsymbol{f}^- \boldsymbol{C}, \quad \boldsymbol{x}^- \geq \boldsymbol{h}^-. \tag{3.41}$$

Объединив два неравенства в одно получим $\boldsymbol{x}^- \geq \boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-$, или $\boldsymbol{x} \leq (\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)^- = (\boldsymbol{s}^T)^-$.

Обратим внимание, что исходная задача может быть записана в форме

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^- \boldsymbol{1}\boldsymbol{1}^T \boldsymbol{C}\boldsymbol{x}; \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq (\boldsymbol{s}^T)^-. \end{aligned} \tag{3.42}$$

Таким образом задача принимает вид задачи 8, где $\boldsymbol{p} = \boldsymbol{1}$, $\boldsymbol{q}^T = \boldsymbol{1}^T \boldsymbol{C}$, а в роли матрицы $\boldsymbol{B}$ и вектора $\boldsymbol{h}$ выступают соответственно матрица $\boldsymbol{R}$ и вектор $(\boldsymbol{s}^T)^-$.

Так как матрица $\boldsymbol{C}$ является регулярной по столбцам, значение $\boldsymbol{q}^T\boldsymbol{p} = \mathbf{1}^T\boldsymbol{C}\mathbf{1} = \|\boldsymbol{C}\| > \mathbb{0}$ и, следовательно, выполняются условия теоремы 8.

Заметим, что, по определению нормы матрицы,

$$\mathbf{1}^T\boldsymbol{C}\boldsymbol{R}^*\mathbf{1} = \|\boldsymbol{C}\boldsymbol{R}^*\|, \tag{3.43}$$

а, по определению нормы вектора,

$$\boldsymbol{h}^-\boldsymbol{R}^i\mathbf{1} = \|\boldsymbol{h}^-\boldsymbol{R}^i\|, \quad \mathbf{1}^T\boldsymbol{C}\boldsymbol{R}^j\boldsymbol{g} = \|\boldsymbol{C}\boldsymbol{R}^j\boldsymbol{g}\|. \tag{3.44}$$

используя теорему 8 получим искомый результат.

Обратим внимание, что в контексте задач временного планирования в управлении проектами элементы неизвестного вектора $\boldsymbol{x}$ определяют время начала работ, а элементы вектора $\boldsymbol{f}$ задают самое позднее время завершения работ. Так как указанное время определенное (т. е. большее, чем $\mathbb{0} = -\infty$), векторы $\boldsymbol{x}$ и $\boldsymbol{f}$ являются регулярными. Условия $\mathrm{Tr}(\boldsymbol{R}) \leq \mathbb{1}$ и $\boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{g} \leq \mathbb{1}$ обеспечивают непротиворечивость ограничений вида «старт-старт», «старт-финиш» и «финиш-старт» вместе с заданными сроками выпуска и директивными сроками работ.

Обратим внимание, что, так же как и в предыдущей задаче, вычислительная сложность поиска решения не превышает $O(n^4)$.

3.3 Задача о наиболее быстром проведении вакцинации

События последних нескольких лет показали, что проблема скорейшего проведения мероприятий по вакцинации населения является черезвычайно актуальной и может потребоваться проведение вакцинации с последующим обязательным наблюдением за пациентом. Особенно в случае недостаточных клинических испытаний вакцины или отсутствия времени на проведение работ по снижению побочных эффектов препарата.

Предположим, что есть необходимость проведения мероприятий по вакцинации граждан. Будем считать, что, пациенты некоторое время нуждаются в наблюдении после прививки. Необходимо провести вакцинацию всех граждан за минимальный промежуток времени

Введем следующие обозначения:

x_i – время проведения вакцинации пациента i;

y_i – время окончания периода наблюдения за пациентом i после применения вакцины;

b_{ij} – минимальный период времени между вакцинацией пациента $j = 1, \ldots, n$ и вакцинацией пациента i (может быть обусловлен медицинскими и бюрократическими ограничениями, такими как изоляция пациентов друг от друга, пополнением запасов вакцины, проведением санитарной обработки помещения, оформление документов) ;

c_{ij} – минимальный промежуток времени между вакцинацией пациента $j = 1, \ldots, n$ и завершением наблюдения за пациентом i (обусловлено ограниченностью ресурсов вакцинационного пункта);

d_{ij} – минимальный период времени между завершением наблюдения за пациентом $j = 1, \ldots, n$ и вакцинацией пациента i (обусловлено ограниченностью ресурсов, например, в случае риска госпитализации);

g_i – наиболее раннее время проведения вакцинации пациента i.

h_i – наиболее позднее допустимое время проведения вакцинации пациента i.

f_i – наиболее позднее оправданное время завершения наблюдения за пациентом i.

Обратим внимание, что рассматриваемая задача является частным случаем задачи (3.29), которая может быть представлена в терминах полуполя $\mathbb{R}_{\max,+}$ в виде (3.36).

Рассмотрим условный медицинский центр, которому требуется организовать вакцинацию $n = 3$ граждан. Временые ограничения заданы при помощи

следующих матриц и векторов:

$$\boldsymbol{B} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -2 & -3 & 0 \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} -4 & -5 & -6 \\ \mathbb{0} & -4 & -7 \\ -5 & \mathbb{0} & -4 \end{pmatrix}, \tag{3.45}$$

$$\boldsymbol{g} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix} \quad \boldsymbol{f} = \begin{pmatrix} 6 \\ 6 \\ 6 \end{pmatrix}. \tag{3.46}$$

Для того, чтобы воспользоваться результатом теоремы 9, необходимо проверить выполнение условий.

Найдем вспомогательные матрицы:

$$\boldsymbol{DC} = \begin{pmatrix} 0 & -2 & -1 \\ -1 & -3 & -2 \\ -1 & -3 & -1 \end{pmatrix}, \quad \boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -1 & -3 & 0 \end{pmatrix}. \tag{3.47}$$

Вычислим степени матрицы $\boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC}$:

$$\boldsymbol{R}^2 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & -1 & 2 \\ -1 & -2 & 0 \end{pmatrix}, \quad \boldsymbol{R}^3 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix}. \tag{3.48}$$

Затем вычислим значения следов $\operatorname{tr} \boldsymbol{R} = \operatorname{tr} \boldsymbol{R}^2 = \operatorname{tr} \boldsymbol{R}^3 = 0$. Поскольку величина

$$\operatorname{Tr}(\boldsymbol{R}) = \operatorname{tr} \boldsymbol{R} \oplus \operatorname{tr} \boldsymbol{R}^2 \oplus \operatorname{tr} \boldsymbol{R}^3 = 0 \tag{3.49}$$

условие $\operatorname{Tr}(\boldsymbol{R}) \leq \mathbb{1} = 0$ выполняется. Далее вычислим

$$\boldsymbol{R}^* = \boldsymbol{I} \oplus \boldsymbol{R} \oplus \boldsymbol{R}^2 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix}, \qquad \boldsymbol{R}^* \boldsymbol{g} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \tag{3.50}$$

$$\boldsymbol{f}^- \boldsymbol{C} = \begin{pmatrix} -2 & -4 & -3 \end{pmatrix}, \qquad \boldsymbol{f}^- \boldsymbol{C} \boldsymbol{B}^* \boldsymbol{g} = -1, \qquad \boldsymbol{h}^- \boldsymbol{B}^* \boldsymbol{g} = -1. \tag{3.51}$$

Таким образом получаем, что условие $\boldsymbol{s}^T \boldsymbol{R}^* \boldsymbol{g} = (\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-) \boldsymbol{R}^* \boldsymbol{g} = -1 < \mathbb{1}$ выполнено.

Найдем минимальный разброс θ. Сперва запишем вспомогательные векторы:

$$\boldsymbol{s}^T = \begin{pmatrix} -2 & -3 & -2 \end{pmatrix}, \quad \boldsymbol{s}^T\boldsymbol{R} = \begin{pmatrix} -2 & -3 & -1 \end{pmatrix}, \tag{3.52}$$

$$\mathbf{1}^T\boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \end{pmatrix}, \quad \boldsymbol{Rg} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}. \tag{3.53}$$

Найдем нормы

$$\|\boldsymbol{CR}^*\| = 5, \quad \|\boldsymbol{Cg}\| = 4, \quad \|\boldsymbol{CRg}\| = 5, \quad \|\boldsymbol{s}^T\| = -2, \quad \|\boldsymbol{s}^T\boldsymbol{R}\| = -1. \tag{3.54}$$

Значение параметра θ можно вычислить по формуле

$$\theta = \|\boldsymbol{CR}^*\| \oplus \|\boldsymbol{s}^T\|\|\boldsymbol{Cg}\| \oplus \|\boldsymbol{s}^T\boldsymbol{R}\|\|\boldsymbol{Cg}\| \oplus \|\boldsymbol{s}^T\|\|\boldsymbol{CRg}\| \oplus \|\boldsymbol{s}^T\boldsymbol{R}\|\|\boldsymbol{CRg}\| = 5. \tag{3.55}$$

Представим все решения $\boldsymbol{x}$ задачи в параметрической форме. Найдем вспомогательные матрицы:

$$\mathbf{1}\mathbf{1}^T\boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 4 & 2 & 3 \\ 4 & 2 & 3 \end{pmatrix}, \quad \boldsymbol{R}\mathbf{1}\mathbf{1}^T\boldsymbol{C} = \begin{pmatrix} 5 & 3 & 4 \\ 6 & 4 & 5 \\ 4 & 2 & 3 \end{pmatrix}, \mathbf{1}\mathbf{1}^T\boldsymbol{CR} = \begin{pmatrix} 4 & 3 & 5 \\ 4 & 3 & 5 \\ 4 & 3 & 5 \end{pmatrix}. \tag{3.56}$$

Вычислим следующие матрицы и вектор:

$$\boldsymbol{G} = \boldsymbol{R}^* \oplus (-5)(\mathbf{1}\mathbf{1}^T\boldsymbol{C} \oplus \boldsymbol{R}\mathbf{1}\mathbf{1}^T\boldsymbol{C} \oplus \mathbf{1}\mathbf{1}^T\boldsymbol{CR}) = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix},$$

$$\boldsymbol{CG} = \begin{pmatrix} 4 & 3 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix}, \qquad (\boldsymbol{s}^T\boldsymbol{G})^- = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \tag{3.57}$$

Запишем все решения $\boldsymbol{x} = (x_1, x_2, x_3)^T$ при помощи вектора параметров

$\boldsymbol{u} = (u_1, u_2, u_3)^T$

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix} \boldsymbol{u}, \qquad \boldsymbol{y} = \begin{pmatrix} 4 & 3 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix} \boldsymbol{u}, \qquad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \le \boldsymbol{u} \le \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \tag{3.58}$$

Так как столбцы матриц $\boldsymbol{G}$ и $\boldsymbol{CG}$ являются коллинеарными, запишем их следующим образом

$$\boldsymbol{G} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -1 & -2 \end{pmatrix} \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix}, \quad \boldsymbol{CG} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix}. \tag{3.59}$$

Определим вспомогательный вектор $\boldsymbol{v}$

$$\boldsymbol{v} = \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix} \boldsymbol{u}, \quad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \le \boldsymbol{u} \le \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 0 \end{pmatrix} \le \boldsymbol{v} \le \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{3.60}$$

и с его помощью запишем решение

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -1 & -2 \end{pmatrix} \boldsymbol{v}, \qquad \boldsymbol{y} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \end{pmatrix} \boldsymbol{v}, \qquad \begin{pmatrix} 0 \\ 0 \end{pmatrix} \le \boldsymbol{v} \le \begin{pmatrix} 2 \\ 3 \end{pmatrix}. \tag{3.61}$$

Представим решения в терминах обычной математики

$$\begin{gathered} x_1 = \max(v_1, v_2 - 1), \qquad x_2 = \max(v_1 + 1, v_2), \quad x_3 = \max(v_1 - 1, v_2 - 2), \\ y_1 = \max(v_1 + 4, v_2 + 3), \quad y_2 = \max(v_1 + 3, v_2 + 2), \quad y_3 = \max(v_1 + 2, v_2 + 1), \\ 0 \le v_1 \le 2, \quad 0 \le v_2 \le 3. \end{gathered} \tag{3.62}$$

Глава 4

Минимизации отклонения от директивных сроков

Рассмотрим задачу минимизации максимального отклонения от директивных сроков выполнения работ с ограничениями на сроки их выполнения. В разделе 4.1 представлена задача тропической оптимизации с ограничениями, к которой в разделе 4.2 будет сведена задача оптимального управления. В разделе 4.3 предложено практическое применение полученного результата, заключающееся в составлении оптимального плана эвакуации в случае ЧС.

4.1 Решение задачи оптимизации с ограничениями

Пусть даны матрицы $\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ и векторы $\boldsymbol{g}, \boldsymbol{h}$ и $\boldsymbol{f}$. Представим следующую задачу тропической оптимизации

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{x}^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} = \boldsymbol{y}, \\ & \boldsymbol{D}\boldsymbol{y} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}, \quad \boldsymbol{y} \leq \boldsymbol{f}. \end{aligned} \tag{4.1}$$

Сформулируем и докажем следующую теорему

Теорема 10 ([41]). *Пусть $\boldsymbol{B}$ и $\boldsymbol{D}$ – матрицы, $\boldsymbol{C}$ – регулярная по столбцам матрица такие, что матрица $\boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{D}\boldsymbol{C}$ удовлетворяет условию $\mathrm{Tr}(\boldsymbol{R}) \leq \mathbb{1}$. Пусть $\boldsymbol{g}$ – вектор, а $\boldsymbol{f}$ и $\boldsymbol{h}$ – регулярные векторы такие, что вектор $\boldsymbol{s}^T = \boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-}$ удовлетворяет условию $\boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{g} \leq \mathbb{1}$.*

Тогда минимальное значение целевой функции в задаче (4.1) *равно*

$$\theta = (\boldsymbol{q}^{-}\boldsymbol{R}^*\boldsymbol{p})^{1/2} \oplus \boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{R}^*\boldsymbol{g}, \tag{4.2}$$

а все регулярные решения имеют вид

$$\boldsymbol{x} = \boldsymbol{R}^*\boldsymbol{u}, \quad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{R}^*\boldsymbol{u}, \tag{4.3}$$

где $\boldsymbol{u}$ *– любой регулярный вектор, который удовлетворяет условиям*

$$\boldsymbol{g} \oplus \theta^{-1}\boldsymbol{p} \leq \boldsymbol{u} \leq ((\boldsymbol{s}^T \oplus \theta^{-1}\boldsymbol{q}^-)\boldsymbol{R}^*)^-. \tag{4.4}$$

Доказательство. Подставив $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$, в неравенство $\boldsymbol{D}\boldsymbol{y} \leq \boldsymbol{x}$ и $\boldsymbol{y} \leq \boldsymbol{f}$ получим $\boldsymbol{D}\boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{x}$ и $\boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}$. Объединяя неравенства $\boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{x}$ и $\boldsymbol{D}\boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{x}$ в одно $\boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}$, где $\boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{D}\boldsymbol{C}$, запишем задачу следующим образом:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^-\boldsymbol{p} \oplus \boldsymbol{q}^-\boldsymbol{x}, \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{4.5}$$

Применив теорему 1, получим, что $\boldsymbol{x} \leq (\boldsymbol{f}^-\boldsymbol{C})^-$. Благодаря антитонности обращения, неравенства, которые ограничивают вектор $\boldsymbol{x}$ сверху могут быть переписаны следующим образом: $\boldsymbol{x}^- \geq \boldsymbol{f}^-\boldsymbol{C}$ и $\boldsymbol{x}^- \geq \boldsymbol{h}^-$. Объединяя полученные неравенства в одно, получим $\boldsymbol{x}^- \geq \boldsymbol{f}^-\boldsymbol{C} \oplus \boldsymbol{h}^-$ или $\boldsymbol{x} \leq (\boldsymbol{f}^-\boldsymbol{C} \oplus \boldsymbol{h}^-)^- = (\boldsymbol{s}^T)^-$.

Обратим внимание, что теперь исходная задача может быть записана в виде

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^-\boldsymbol{p} \oplus \boldsymbol{q}^-\boldsymbol{x}, \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq (\boldsymbol{s}^T)^-. \end{aligned} \tag{4.6}$$

Новая задача имеет вид задачи 2.34, где в роли матрицы $\boldsymbol{B}$ и вектора $\boldsymbol{h}$ выступают соответственно $\boldsymbol{R}$ и $(\boldsymbol{s}^T)^-$.

Оценим вычислительную сложность поиска решения задачи. Самой трудоемкой операцией является вычисление матрицы Клини, которое потребует не более чем $O(n^4)$ скалярных операций. Вычисление остальных элементов заключается в фиксированном числе операций над матрицами и векторами и не превосходит $O(n^3)$. Таким образом вычислительная сложность поиска решения задачи не превосходит $O(n^4)$ скалярных операций.

4.2 Минимизация отклонения от директивных сроков

Рассмотрим задачу составления оптимального календарного плана выполнения работ в соответствии с критерием максимального выхода директивных сроков выполнения проекта за пределы заданного временного интервала (1.8)

с ограничениями вида «старт-старт», «старт-финиш», «финиш-старт», «время выпуска», «время прекращения выпуска»и «крайний срок завершения». Запишем задачу минимизации с ограничениями в следующем виде

$$\begin{aligned} \min_{x_i,y_i} \quad & \max\left(\max_{1\le i\le n}(p_i - x_i), \max_{1\le i\le n}(x_i - q_i)\right) \\ & \max_{1\le j\le n}(b_{ij} + x_j) \le x_i, \quad \max_{1\le j\le n}(c_{ij} + x_j) = y_i, \\ & \max_{1\le j\le n}(d_{ij} + y_j) \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1,\dots,n. \end{aligned} \tag{4.7}$$

Обратим внимание, что в случае равенства значений $p_i = q_i$ временные интервалы сужаются до скалярных значений директивных сроков начала работ, а критерий оптимальности можно представить следующим образом

$$\max\left(\max_{1\le i\le n}(p_i - x_i), \max_{1\le i\le n}(x_i - p_i)\right) \tag{4.8}$$

Сформулируем задачу минимизации максимального отклонения от директивных сроков:

$$\begin{aligned} \min_{x_i,y_i} \quad & \max\left(\max_{1\le i\le n}(p_i - x_i), \max_{1\le i\le n}(x_i - p_i)\right) \\ & \max_{1\le j\le n}(b_{ij} + x_j) \le x_i, \quad \max_{1\le j\le n}(c_{ij} + x_j) = y_i, \\ & \max_{1\le j\le n}(d_{ij} + y_j) \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1,\dots,n. \end{aligned} \tag{4.9}$$

Представим задачу (4.7) в терминах полуполя $\mathbb{R}_{\max,+}$. Для этого определим следующие матрично-векторные обозначения:

$$\boldsymbol{B} = (b_{ij}), \quad \boldsymbol{C} = (c_{ij}), \quad \boldsymbol{D} = (d_{ij}), \tag{4.10}$$

$$\boldsymbol{x} = (x_i), \quad \boldsymbol{f} = (f_i), \quad \boldsymbol{g} = (g_i), \quad \boldsymbol{h} = (h_i), \quad \boldsymbol{p} = (p_i), \quad \boldsymbol{q} = (q_i). \tag{4.11}$$

В терминах полуполя $\mathbb{R}_{\max,+}$ максимальный выход за нижний конец интервала имеет вид $x_i^- p_i$, а максимальный выход за верхний конец интервала $q_i^- x_i$. Целевую функцию можно записать в следующем виде:

$$\bigoplus_{1\le i\le n} x_i^- p_i \oplus \bigoplus_{1\le i\le n} q_i^- x_i. \tag{4.12}$$

Сформулируем задачу в терминах идемпотентного полуполя $\mathbb{R}_{\max,+}$

$$\begin{aligned}\min_{x_i,y_i} \quad & \bigoplus_{1\le i\le n} x_i^- p_i \oplus \bigoplus_{1\le i\le n} q_i^- x_i, \\ & \bigoplus_{1\le j\le n} b_{ij}x_j \le x_i, \quad \bigoplus_{1\le j\le n} c_{ij}x_j = y_i, \\ & \bigoplus_{1\le j\le n} d_{ij}y_j \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1,\dots,n\end{aligned} \tag{4.13}$$

и перепишем задачу в матрично-векторной форме

$$\begin{aligned}\min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{x}^-\boldsymbol{p} \oplus \boldsymbol{q}^-\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} = \boldsymbol{y}, \\ & \boldsymbol{D}\boldsymbol{y} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}, \quad \boldsymbol{y} \le \boldsymbol{f}.\end{aligned} \tag{4.14}$$

Таким образом исследуемая задача принимает вид 4.1 и может быть решена при помощи теоремы 10, а задача (4.9) сводится к задаче (4.7) и также решается при помощи теоремы 10, в которой следует положить $\boldsymbol{q} = \boldsymbol{p}$.

4.3 Задача об организации эвакуации

Рассмотрим процесс организации эвакуации населения в случае черезвычайной ситуации (например аварии на атомной станции). Будем считать, что есть заданное количество населенных пунктов, которые требуется эвакуировать. Слишком раннее время эвакуации населенного пункта может вызвать проблемы, связанные с перегрузкой дорожной сети (которая может потребоваться, например, для сил ЧС, организовывающих ликвидацию последствий) или с тем, что местные власти не успеют собрать всех граждан, подлежащих эвакуации. Слишком позднее время эвакуации также нежелательно, поскольку может представлять опасность для граждан. Следовательно у каждого пункта есть желаемое время начала эвакуации. Требуется организовать эвакуацию минимально отклонившись от заданных директивных сроков. Запишем вспомогательные обозначения.

x_i – время начала эвакуации населенного пункта i;

y_i – время завершения эвакуации i;

b_{ij} – минимальный временной интервал между началом эвакуации пункта $j = 1, \ldots, n$ и началом эвакуации i (обусловлен возможной нехваткой персонала и транспорта для одновременного начала эвакуации) ;

c_{ij} – минимальный период времени между началом эвакуации пункта $j = 1, \ldots, n$ и завершением эвакуации пункта i (может быть обусловлено последовательностью эвакуации. Например небольшие населенные пункты могут быть эвакуированы одним транспортом);

d_{ij} – минимальный период времени между завершением эвакуации $j = 1, \ldots, n$ и началом эвакуации i (может быть обусловлено расстоянием от пункта эвакуации до населенного пункта, необходимостью санитарной обработки транспорта и отдыха персонала);

g_i – наиболее раннее возможное время начала эвакуации пункта i;

h_i – наиболее позднее допустимое время начала эвакуации пункта i;

f_i – наиболее позднее допустимое время завершения эвакуации пункта i.

Представленная задача является частным случаем задачи (4.7), которую можно представить в терминах полуполя $\mathbb{R}_{\max,+}$ в виде (4.9).

Рассмотрим условную задачу эвакуации $n = 3$ населенных пунктов при помощи следующих матриц и векторов

$$\boldsymbol{B} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -2 & -3 & 0 \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} -4 & -5 & -6 \\ \mathbb{0} & -4 & -7 \\ -5 & \mathbb{0} & -4 \end{pmatrix}, \quad (4.15)$$

$$\boldsymbol{g} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} 6 \\ 6 \\ 6 \end{pmatrix}, \quad \boldsymbol{p} = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}. \quad (4.16)$$

Проверим, что выполняются условия теоремы 10. Найдем матрицы

$$\boldsymbol{DC} = \begin{pmatrix} 0 & -2 & -1 \\ -1 & -3 & -2 \\ -1 & -3 & -1 \end{pmatrix}, \quad \boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -1 & -3 & 0 \end{pmatrix}, \tag{4.17}$$

и затем вычислим степени

$$\boldsymbol{R}^2 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & -1 & 2 \\ -1 & -2 & 0 \end{pmatrix}, \quad \boldsymbol{R}^3 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix}. \tag{4.18}$$

Принимая во внимание, что выполняется условие $\mathrm{Tr}(\boldsymbol{R}) = \mathrm{tr}\,\boldsymbol{R} \oplus \mathrm{tr}\,\boldsymbol{R}^2 \oplus \mathrm{tr}\,\boldsymbol{R}^3 = 0 \leq \mathbb{1}$, последовательно вычисляем

$$\boldsymbol{R}^* = \boldsymbol{I} \oplus \boldsymbol{R} \oplus \boldsymbol{R}^2 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix}, \qquad \boldsymbol{R}^*\boldsymbol{g} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \tag{4.19}$$

$$\boldsymbol{f}^-\boldsymbol{C} = \begin{pmatrix} -2 & -4 & -3 \end{pmatrix}, \qquad \boldsymbol{f}^-\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{g} = -1, \qquad \boldsymbol{h}^-\boldsymbol{B}^*\boldsymbol{g} = -1. \tag{4.20}$$

Убедимся, что справедливо условие

$$\boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{g} = (\boldsymbol{f}^-\boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{R}^*\boldsymbol{g} = -1 < \mathbb{1}. \tag{4.21}$$

Найдем скаляры

$$\boldsymbol{p}^-\boldsymbol{R}^*\boldsymbol{p} = 1, \quad \boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{p} = 1, \quad \boldsymbol{p}^-\boldsymbol{R}^*\boldsymbol{g} = 0. \tag{4.22}$$

Подставляя найденные величины, вычислим минимум целевой функции

$$\theta = (\boldsymbol{p}^-\boldsymbol{R}^*\boldsymbol{p})^{1/2} \oplus \boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{p} \oplus \boldsymbol{p}^-\boldsymbol{R}^*\boldsymbol{g} = 1. \tag{4.23}$$

Найдем векторы:

$$\boldsymbol{s}^T \oplus \theta^{-1}\boldsymbol{p}^- = \begin{pmatrix} -2 & -3 & -2 \end{pmatrix},$$

$$\boldsymbol{g} \oplus \theta^{-1}\boldsymbol{p} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad ((\boldsymbol{s}^T \oplus \theta^{-1}\boldsymbol{p}^-)\boldsymbol{R}^*)^- = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \tag{4.24}$$

Представим все решения $\boldsymbol{x} = (x_1, x_2, x_3)$ и $\boldsymbol{y} = (y_1, y_2, y_3)$ при помощи вектора параметров $\boldsymbol{u} = (u_1, u_2, u_3)$ в виде

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix} \boldsymbol{u}, \quad \boldsymbol{y} = \begin{pmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix} \boldsymbol{x}, \quad \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \leq \boldsymbol{u} \leq \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \quad (4.25)$$

Так как все столбцы матрицы $\boldsymbol{R}^*$ коллинеарны она может быть записана следующим образом:

$$\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 1 \end{pmatrix}. \quad (4.26)$$

Вычислим значение скаляра $u = u_1 \oplus (-1)u_2 \oplus (1)u_3$. Представим решение, заменив вектор параметров $\boldsymbol{u}$ на введенный скаляр и вычислив границы для u

$$\boldsymbol{x} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} u, \quad \boldsymbol{y} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix} u, \quad 1 \leq u \leq 2. \quad (4.27)$$

Представим решение в терминах обычных арифметических операции

$$x_1 = u,\ x_2 = u + 1,\ x_3 = u - 1,\ y_1 = u + 4,\ y_2 = u + 3,\ y_3 = u + 2.\ 1 \leq u \leq 2 \quad (4.28)$$

Глава 5

Минимизация разброса времени завершения работ

В данной главе сформулирована и решена задача составления оптимального календарного графика выполнения работ проекта, которая заключается в минимизации разброса времени завершения работ проекта с заданными ограничениями на последовательность выполнения работ. В разделе 5.1 сформулирована и решена задача тропической оптимизации, которая в дальнейшем будет использована в разделе 5.2 для решения исследуемой задачи оптимального управления.

5.1 Решение задачи оптимизации с ограничениями

Рассмотрим следующую задачу. Пусть заданы матрицы $\boldsymbol{A}, \boldsymbol{C} \in \mathbb{X}^{m\times n}$, $\boldsymbol{B} \in \mathbb{X}^{n\times n}$, векторы $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{f} \in \mathbb{X}^m$ и вектор $\boldsymbol{g} \in \mathbb{X}^n$. Запишем задачу минимизации целевой функции при заданных ограничениях относительно неизвестного $\boldsymbol{x} \in \mathbb{X}^n$

$$\begin{aligned} \min \quad & \boldsymbol{q}^- \boldsymbol{x} (\boldsymbol{A}\boldsymbol{x})^- \boldsymbol{p}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}, \quad \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{5.1}$$

которая является дальнейшим усложнением задачи (2.25), полученным при помощи добавления новых неравенств-ограничений. Сформулируем и докажем следующую теорему:

Теорема 11 ([38]). *Пусть заданы регулярная по строкам матрица $\boldsymbol{A}$, регулярная по столбцам матрица $\boldsymbol{C}$ и матрица $\boldsymbol{B}$ такая, что $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$. Предположим, что вектор $\boldsymbol{p}$ является ненулевым, векторы $\boldsymbol{q}$, $\boldsymbol{f}$ и $\boldsymbol{h}$ – регулярные, и удовлетворяют условию $\boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{g} \leq ((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{B}^* (\boldsymbol{q}^- \boldsymbol{B}^*)^-)^{-1}$.*

Тогда минимум в задаче (5.1) *равен $\Delta = (\boldsymbol{A}\boldsymbol{B}^*(\boldsymbol{q}^-\boldsymbol{B}^*)^-)^-\boldsymbol{p}$ и достигается при*

$$\boldsymbol{x} = \alpha \boldsymbol{B}^* \left(\boldsymbol{q}^- \boldsymbol{B}^*\right)^-, \qquad \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{g} \leq \alpha \leq ((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{B}^*(\boldsymbol{q}^- \boldsymbol{B}^*)^-)^{-1}. \tag{5.2}$$

Доказательство. Обратим внимание, что для решения первого ограничения-неравенства $\boldsymbol{Bx} \oplus \boldsymbol{g} \leq \boldsymbol{x}$ можно воспользоваться теоремой 3. Запишем вместо неравенства его решение $\boldsymbol{x} = \boldsymbol{B}^*\boldsymbol{u}$, где $\boldsymbol{u}$ – регулярный вектор такой, что $\boldsymbol{u} \geq \boldsymbol{g}$. После подстановки получим новую задачу оптимизации относительно неизвестного вектора $\boldsymbol{u}$:

$$\begin{aligned} \min \quad & \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{u} (\boldsymbol{A}\boldsymbol{B}^* \boldsymbol{u})^- \boldsymbol{p}, \\ & \boldsymbol{B}^* \boldsymbol{u} \leq \boldsymbol{h}, \quad \boldsymbol{C}\boldsymbol{B}^* \boldsymbol{u} \leq \boldsymbol{f}, \quad \boldsymbol{u} \geq \boldsymbol{g}. \end{aligned} \tag{5.3}$$

Так как векторы $\boldsymbol{f}$ и $\boldsymbol{h}$ регулярные, а матрица $\boldsymbol{C}$ регулярна по столбцам и $\boldsymbol{B}^* \geq \boldsymbol{I}$, матрицы $\boldsymbol{B}^*$ и $\boldsymbol{C}\boldsymbol{B}^*$ также регулярны по столбцам. Таким образом, для неравенств $\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{u} \leq \boldsymbol{f}$ и $\boldsymbol{B}^*\boldsymbol{u} \leq \boldsymbol{h}$ справедливы условия теоремы 1. После решения неравенства получаем задачу относительно вектора $\boldsymbol{u}$, записанную следующим образом

$$\begin{aligned} \min \quad & \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{u} (\boldsymbol{A}\boldsymbol{B}^* \boldsymbol{u})^- \boldsymbol{p}, \\ & \boldsymbol{g} \leq \boldsymbol{u} \leq (\boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^*)^-, \quad \boldsymbol{u} \leq (\boldsymbol{h}^- \boldsymbol{B}^*)^-. \end{aligned} \tag{5.4}$$

Изучим данную задачу без ограничений. Поскольку матрица $\boldsymbol{A}$ регулярна по строкам, а вектор $\boldsymbol{q}$ – регулярный, матрица $\boldsymbol{A}\boldsymbol{B}^*$ также регулярна по строкам, а вектор $(\boldsymbol{q}^- \boldsymbol{B}^*)^-$ является регулярным. Воспользовавшись теоремой 4, находим, что при отсутствии ограничений минимум в задаче равен $\Delta = (\boldsymbol{A}\boldsymbol{B}^*(\boldsymbol{q}^- \boldsymbol{B}^*)^-)^- \boldsymbol{p}$ и достигается на любом векторе

$$\boldsymbol{u} = \alpha \left(\boldsymbol{q}^- \boldsymbol{B}^*\right)^-, \qquad \alpha > \mathbb{0}. \tag{5.5}$$

Определим, какие из найденных решений удовлетворяют имеющимся ограничениям. Обратим внимание, что ограничения $\boldsymbol{u} \leq (\boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^*)^-$ и $\boldsymbol{u} \leq (\boldsymbol{h}^- \boldsymbol{B}^*)^-$ могут быть представлены при помощи неравенств $\boldsymbol{u}^- \geq \boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^*$ и $\boldsymbol{u}^- \geq \boldsymbol{h}^- \boldsymbol{B}^*$, или в виде одного равносильного неравенства $\boldsymbol{u}^- \geq \boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^* \oplus \boldsymbol{h}^- \boldsymbol{B}^*$. Полученное неравенство эквивалентно условию $\boldsymbol{u} \leq (\boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^* \oplus \boldsymbol{h}^- \boldsymbol{B}^*)^-$. Чтобы найти значения α, при которых выполняются ограничения задачи, потребуется решить следующее двойное неравенство

$$\boldsymbol{g} \leq \alpha(\boldsymbol{q}^- \boldsymbol{B}^*)^- \leq (\boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^* \oplus \boldsymbol{h}^- \boldsymbol{B}^*)^-. \tag{5.6}$$

После решения левого и правого неравенств получим

$$\boldsymbol{q}^{-}\boldsymbol{B}^{*}\boldsymbol{g} \leq \alpha \leq ((\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})\boldsymbol{B}^{*}(\boldsymbol{q}^{-}\boldsymbol{B}^{*})^{-})^{-1}. \tag{5.7}$$

По условиям теоремы множество значений α, которые удовлетворяют неравенству – не пусто. После возвращения к исходной задаче относительно вектора $\boldsymbol{x}$ получим искомый результат.

Оценим вычислительную сложности найденого решения. В исследуемой задаче самой сложной операцией является вычисление матрицы Клини, которое состоит из вычисления первых $n-1$ степеней матрицы $\boldsymbol{B}$ и требует не более чем $O(n^4)$ скалярных операций. Вычисление прочих элементов решения заключается в алгебраических операциях над матрицами и векторами со сложностью не превышающей $O(n^3)$. Следовательно поиск решения задачи требует не более чем чем $O(n^4)$ скалярных операций.

5.2 Минимизация разброса времени завершения работ

В данном разделе представлено решение задачи сетевого планирования, заключающейся в минимизации критерия максимального разброса времени выполнения всех работ проекта (1.9) с добавлением ограничений на время их выполнения вида «старт-старт», «старт-финиш», «время выпуска», «время прекращения выпуска» и «крайний срок завершения»

$$\begin{aligned} \min_{x_i,y_i} \quad & \max_{1\leq i\leq n} y_i + \max_{1\leq i\leq n} (-y_i)\,, \\ & \max(\max_{1\leq j\leq n} (b_{ij}+x_j)\,, g_i) \leq x_i, \quad \max_{1\leq j\leq n} (c_{ij}+x_j) = y_i, \\ & y_i \leq f_i \quad x_i \leq h_i, \quad i=1,\ldots,n. \end{aligned} \tag{5.8}$$

Обратим внимание, что при записи задачи (5.8) в обычных обозначениях используются только операции сложения, вычитания и определения максимума и, следовательно, задача легко может быть представлена в тропическом виде в терминах полуполя $\mathbb{R}_{\max,+}$. Уравнения и неравенства ограничений в терминах

полуполя $\mathbb{R}_{\max,+}$ могут быть записаны следующим образом:

$$\bigoplus_{j=1}^{n} b_{ij}x_j \oplus g_i \le x_i, \quad \bigoplus_{j=1}^{n} c_{ij}x_j = y_i, \quad x_i \le h_i, \quad y_i \le f_i \quad i = 1, \dots, n. \tag{5.9}$$

Определим следующие матрично-векторные обозначения

$$\boldsymbol{B} = (b_{ij}), \qquad \boldsymbol{C} = (c_{ij}), \qquad \boldsymbol{y} = (y_i), \qquad \boldsymbol{g} = (g_i), \qquad \boldsymbol{h} = (h_i), \qquad \boldsymbol{f} = (f_i) \tag{5.10}$$

и представим ограничения в матрично-векторной форме

$$\boldsymbol{Bx} \oplus \boldsymbol{g} \le \boldsymbol{x}, \quad \boldsymbol{Cx} = \boldsymbol{y}, \quad \boldsymbol{x} \le \boldsymbol{h}, \quad \boldsymbol{y} \le \boldsymbol{f}. \tag{5.11}$$

Запишем целевую функцию задачи в тропическом виде

$$\bigoplus_{i=1}^{n} y_i \bigoplus_{j=1}^{n} y_j = \mathbf{1}^T \boldsymbol{y}\boldsymbol{y}^{-}\mathbf{1}. \tag{5.12}$$

Таким образом, задача (5.8) целиком может быть записана следующим образом:

$$\begin{aligned} \min \quad & \mathbf{1}^T \boldsymbol{y}\boldsymbol{y}^{-}\mathbf{1}, \\ & \boldsymbol{Bx} \oplus \boldsymbol{g} \le \boldsymbol{x}, \quad \boldsymbol{Cx} = \boldsymbol{y}, \\ & \boldsymbol{x} \le \boldsymbol{h}, \quad \boldsymbol{y} \le \boldsymbol{f}. \end{aligned} \tag{5.13}$$

Теорема 12 ([38]). *Пусть заданы регулярная матрица $\boldsymbol{C}$ и матрица $\boldsymbol{B}$ такая, что* $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$. *Будем считать, что $\boldsymbol{f}$ и $\boldsymbol{h}$ регулярны и удовлетворяют следующему условию*

$$\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^{*}\boldsymbol{g} \le \left((\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})\boldsymbol{B}^{*}(\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^{*})^{-}\right)^{-1}. \tag{5.14}$$

Тогда минимум в задаче (5.13) *равен*

$$\Delta = (\boldsymbol{C}\boldsymbol{B}^{*}(\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^{*})^{-})^{-}\mathbf{1} \tag{5.15}$$

и достигается при

$$\boldsymbol{x} = \alpha \boldsymbol{B}^{*}(\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^{*})^{-}, \qquad \boldsymbol{y} = \alpha \boldsymbol{C}\boldsymbol{B}^{*}(\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^{*})^{-}, \tag{5.16}$$

где

$$\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^{*}\boldsymbol{g} \le \alpha \le \left((\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})\boldsymbol{B}^{*}(\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^{*})^{-}\right)^{-1}. \tag{5.17}$$

Доказательство. После подстановки равенства $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ в целевую функцию и ограничения задача принимает следующий вид

$$\begin{aligned} \min \quad & \mathbf{1}^T\boldsymbol{C}\boldsymbol{x}(\boldsymbol{C}\boldsymbol{x})^{-}\mathbf{1}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x}, \\ & \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}, \\ & \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{5.18}$$

Заметим, что полученная задача имеет вид задачи (5.1), в которой вектор $\boldsymbol{q}^{-}$ заменяется на единичный вектор $\mathbf{1}^T\boldsymbol{C}$ и $\boldsymbol{C}$ на $\boldsymbol{C}$. Обратим внимание, что так как матрица $\boldsymbol{C}$ – регулярна, то вектор $\mathbf{1}^T\boldsymbol{C}$ также регулярный. Более того, так как $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$, а векторы $\boldsymbol{f}$ и $\boldsymbol{h}$ являются регулярными и удовлетворяют ограничению $\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{g} \leq \big((\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})\boldsymbol{B}^*(\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*)^{-}\big)^{-1}$, выполняются условия теоремы 11. Воспользовавшись теоремой находим минимум в задаче $\Delta = (\boldsymbol{C}\boldsymbol{B}^*(\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*)^{-})^{-}\mathbf{1}$, который достигается при $\boldsymbol{x} = \alpha\boldsymbol{B}^*(\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*)^{-}$, где параметр α задается условием $\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{g} \leq \alpha \leq \big((\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})\boldsymbol{B}^*(\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*)^{-}\big)^{-1}$. Принимая во внимание, что $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$, получим требуемый результат.

5.3 Оптимизация проведения диспансеризации

Регулярное проведение диспансеризации позволяет осуществлять мониторинг состояния здоровья граждан и выявлять большое количество заболеваний на ранних стадиях, что в конечном итоге, позволяет увеличить продолжительность жизни населения и снизить нагрузку на систему здравоохранения в долгосрочной перспективе. Тем не менее сами по себе мероприятия по регулярному проведению качественной диспансеризации достаточно затратны и создают значительную нагрузку на систему здравоохранения. Перечисленные факты делают проблему оптимизации проведения диспансеризации важной и актуальной. Сама по себе диспансеризация заключается в проведении некоторого количества медицинских процедур. Сформулируем задачу прохождения диспансеризации более подробно.

Будем считать, что необходимо минимизировать разброс времени завершения процедур, что может потребоваться, например, при осуществлении выезд-

ного осмотра или для последующего производства необходимых работ, связанных с общей приостановкой деятельности (например при дезинфекции). Введем следующие вспомогательные обозначения:

x_i – время начала процедуры i (взятие анализов, осмотр специалистом, регистрация);

y_i – время завершения проведения процедуры i;

b_{ij} – минимальный период времени между началом процедуры $j = 1, \ldots, n$ и началом процедуры i. В качестве практического примера подобного ограничения можно рассмотреть например запрет на слишком частый забор крови;

c_{ij} – минимальный временной интервал между началом процедуры $j = 1, \ldots, n$ и завершением i. Может быть обусловлено например тем, что последующая консультация врача имеет смысл только после получения результатов анализов;

g_i – самое раннее возможное время начала проведения процедуры i;

h_i – самое позднее возможное время начала процедуры i;

f_i – наиболее позднее допустимое время завершения процедуры i.

Обратим внимание, что представленная задача является частным случаем рассмотренной выше задачи сетевого планирования (5.8) и она может быть записана в форме (5.13).

Для более наглядного описания полученного результата, рассмотрим условное проведение диспансеризации, заключающееся в осуществлении $n = 3$ процедур с ограничениями на время их выполнения, заданными в терминах полуполя $\mathbb{R}_{\max,+}$ с помощью следующих матриц:

$$\boldsymbol{B} = \begin{pmatrix} \mathbb{0} & -2 & 1 \\ 0 & \mathbb{0} & 2 \\ -1 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \qquad \boldsymbol{C} = \begin{pmatrix} 4 & 0 & \mathbb{0} \\ 2 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}, \tag{5.19}$$

где $\mathbb{0} = -\infty$, и векторов

$$\boldsymbol{g} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \qquad \boldsymbol{h} = \begin{pmatrix} 5 \\ 4 \\ 4 \end{pmatrix}, \qquad \boldsymbol{f} = \begin{pmatrix} 8 \\ 8 \\ 10 \end{pmatrix}. \tag{5.20}$$

Убедимся, что условия теоремы 12 выполняются. Легко видеть, что матрица $\boldsymbol{A}$ регулярна. Проверим, что $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$. Для этого потребуется вычислить следующие матрицы:

$$\boldsymbol{B}^2 = \begin{pmatrix} 0 & \mathbb{0} & 0 \\ 1 & -2 & 1 \\ \mathbb{0} & -3 & 0 \end{pmatrix}, \qquad \boldsymbol{B}^3 = \begin{pmatrix} -1 & -2 & 1 \\ 0 & -1 & 2 \\ -1 & \mathbb{0} & -1 \end{pmatrix}. \tag{5.21}$$

После вычисления следов найдем значения $\mathrm{Tr}(\boldsymbol{B}) = \mathrm{tr}\,\boldsymbol{B} \oplus \mathrm{tr}\,\boldsymbol{B}^2 \oplus \mathrm{tr}\,\boldsymbol{B}^3 = 0 = \mathbb{1}$. Таким образом определена матрица Клини

$$\boldsymbol{B}^* = \boldsymbol{I} \oplus \boldsymbol{B} \oplus \boldsymbol{B}^2 = \begin{pmatrix} 0 & -2 & 1 \\ 1 & 0 & 2 \\ -1 & -3 & 0 \end{pmatrix}. \tag{5.22}$$

Убедимся, что верно неравенство

$$\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* \boldsymbol{g} \leq \left((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-) \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^- \right)^{-1}. \tag{5.23}$$

Сначала найдем

$$\begin{gathered} \boldsymbol{C}\boldsymbol{B}^* = \begin{pmatrix} 4 & 2 & 5 \\ 4 & 3 & 5 \\ 2 & 1 & 3 \end{pmatrix}, \qquad \mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* = \begin{pmatrix} 4 & 3 & 5 \end{pmatrix}, \\ \mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* \boldsymbol{g} = 5, \qquad \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^- = \begin{pmatrix} -4 \\ -3 \\ -5 \end{pmatrix}. \end{gathered} \tag{5.24}$$

Вычислим вспомогательные векторы

$$\boldsymbol{f}^- \boldsymbol{C} = \begin{pmatrix} -4 & -5 & -7 \end{pmatrix}, \qquad \boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^- = \begin{pmatrix} -4 & -4 & -4 \end{pmatrix}. \tag{5.25}$$

Используя полученные векторы, найдем

$$((\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})\boldsymbol{B}^{*}(\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{B}^{*})^{-})^{-1} = 7, \tag{5.26}$$

следовательно выполняется условие $\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{B}^{*}\boldsymbol{g} \leq \left((\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})\boldsymbol{B}^{*}(\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{B}^{*})^{-}\right)^{-1}$.

Воспользовавшись теоремой 12, получаем минимум задачи Δ

$$\boldsymbol{C}\boldsymbol{B}^{*}(\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{B}^{*})^{-} = \begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix}, \qquad \Delta = (\boldsymbol{C}\boldsymbol{B}^{*}(\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{B}^{*})^{-})^{-}\mathbf{1} = 2, \tag{5.27}$$

который достигается при

$$\boldsymbol{x} = \alpha\boldsymbol{B}^{*}(\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{B}^{*})^{-} = \alpha \begin{pmatrix} -4 \\ -3 \\ -5 \end{pmatrix}, \ \boldsymbol{y} = \alpha\boldsymbol{C}\boldsymbol{B}^{*}(\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{B}^{*})^{-} = \\ = \alpha \begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix}, \ 5 \leq \alpha \leq 7. \tag{5.28}$$

Обратим внимание, что при значении $\alpha = 5$ мы имеем план с самыми ранними сроками начала процедур, который задается векторами

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} 5 \\ 5 \\ 3 \end{pmatrix}, \tag{5.29}$$

а при значении $\alpha = 7$ имеем план с наиболее поздними сроками начала процедур, для которого

$$\boldsymbol{x} = \begin{pmatrix} 3 \\ 4 \\ 2 \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} 7 \\ 7 \\ 5 \end{pmatrix}. \tag{5.30}$$

Глава 6

Максимизация разброса времени завершения работ

Рассмотрим задачу составления оптимального календарного плана выполнения работ проекта, которая заключается в максимизации разброса времени завершения выполнения работ проекта при заданных ограничениях. В разделе 6.1 представлено решение задачи тропической оптимизации, которая в дальнейшем будет использована в разделе 6.2 для решения задачи сетевого планирования. В разделе 6.3 предложено практическое применение полученного результата для оптимизации составления графика медицинских процедур, связанных со вредным воздействием на организм.

6.1 Решение задачи тропической оптимизации с ограничениями

Пусть, даны матрицы $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{X}^{n\times n}$ и векторы $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{g}, \boldsymbol{h} \in \mathbb{X}^n$. Рассмотрим следующую задачу оптимизации

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \boldsymbol{q}^{-}\boldsymbol{x}(\boldsymbol{A}\boldsymbol{x})^{-}\boldsymbol{p}, \\ & \boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{6.1}$$

Представим теорему, которая предлагает решение исследуемой задачи.

Теорема 13 ([39]). *Предположим, что матрицы $\boldsymbol{A}$ и $\boldsymbol{B}$ – такие, что $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$, матрица $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^*$ имеет регулярные столбцы $\boldsymbol{d}_j = (d_{ij})$, где $\boldsymbol{B}^* = (\boldsymbol{b}_j^*)$ – матрица Клини, а матрица $\boldsymbol{C}$ является регулярной по столбцам. Пусть векторы $\boldsymbol{p} = (p_j)$, $\boldsymbol{q}$ и $\boldsymbol{h}$ – регулярные, а вектор $\boldsymbol{r} = (r_j)$ определен как $\boldsymbol{r} = (\boldsymbol{h}^{-}\boldsymbol{C}\boldsymbol{B}^*)^{-}$.*

Тогда максимум в задаче (6.1) *равен*

$$\Delta = \boldsymbol{q}^{-}\boldsymbol{B}^*\boldsymbol{D}^{-}\boldsymbol{p} \tag{6.2}$$

и достигается тогда, когда $\boldsymbol{x} = \boldsymbol{B}^*\boldsymbol{u}$, *где* $\boldsymbol{u} = (u_j)$ – *любой вектор с компонентами*

$$u_k = \alpha d_{sk}^{-1} p_s, \quad u_j \le \alpha d_{sj}^{-1} p_s, \quad j \ne k, \tag{6.3}$$

при условии, что

$$\alpha \le \min_{1\le j\le n} d_{sj} p_s^{-1} r_j, \quad k = \arg\max_{1\le j\le n} \boldsymbol{q}^- \boldsymbol{b}_j^* \boldsymbol{d}_j^- \boldsymbol{p}, \quad s = \arg\max_{1\le j\le n} d_{jk}^{-1} p_j. \tag{6.4}$$

Доказательство. Заменяя неравенство $\boldsymbol{Bx} \le \boldsymbol{x}$ на его решение $\boldsymbol{x} = \boldsymbol{B}^*\boldsymbol{u}$, где $\boldsymbol{u}$ – любой регулярный вектор подходящего размера получим новую задачу относительно вектора $\boldsymbol{u}$ в следующем виде

$$\begin{gathered} \max_{\boldsymbol{u}} \ \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{u} \left(\boldsymbol{Du}\right)^- \boldsymbol{p}, \\ \boldsymbol{CB}^* \boldsymbol{u} \le \boldsymbol{h}. \end{gathered} \tag{6.5}$$

Используя теорему 7 для вычисления максимума в задаче без ограничений получим максимум равный $\Delta = \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{D}^- \boldsymbol{p}$, который достигается тогда и только тогда, когда вектор $\boldsymbol{u}$ имеет компоненты, определяемые условиями (2.40)-(2.41).

Выясним, какие из найденных решений удовлетворяют неравенству $\boldsymbol{CB}^*\boldsymbol{u} \le \boldsymbol{h}$. Так как матрица $\boldsymbol{C}$ регулярна по столбцам и $\boldsymbol{B}^* \ge \boldsymbol{I}$, матрица $\boldsymbol{CB}^*$ является регулярной по столбцам. Следовательно рассматриваемое неравенство удовлетворяет условию теоремы 1 и может быть решено следующим образом $\boldsymbol{u} \le (\boldsymbol{h}^- \boldsymbol{CB}^*)^- = \boldsymbol{r}$.

Найдем при каких значениях α любой вектор $\boldsymbol{u}$, который задается условиями (2.40), удовлетворяет неравенству $\boldsymbol{u} \le \boldsymbol{r}$. Для этого достаточно, чтобы для всех $j = 1, \ldots, n$ были справедливы неравенства $\alpha d_{sj}^{-1} p_s \le r_j$. После решения этих неравенств, получим для всех j неравенства $\alpha \le d_{sj} p_s^{-1} r_j$, которые можно заменить эквивалентным неравенством

$$\alpha \le \min_{1\le j\le n} d_{sj} p_s^{-1} r_j. \tag{6.6}$$

Параметр α, удовлетворяющий данному условию, задает векторы $\boldsymbol{u}$, при которых все векторы $\boldsymbol{x} = \boldsymbol{B}^*\boldsymbol{u}$, являются решениями задачи (6.1).

Основную вычислительную сложность в задаче составляет поиск матрицы Клини, требующий не более чем $O(n^4)$ скалярных операций. Вычисление остальных элементов состоим в фиксированном числе операций над матрицами и векторами, а также в поиске индексов, и его вычислительная сложность не превосходит $O(n^3)$. Следовательно поиск решения потребует не более чем $O(n^4)$ скалярных операций.

6.2 Максимизация разброса времен завершения

Представим решение задачи максимизации критерия максимального разброса времени завершения работ проекта (1.9) при ограничениях вида «старт-старт», «старт-финиш» и «крайний срок завершения»

$$\begin{aligned} \max_{x_i,y_i} \quad & \max_{1\le i\le n} y_i + \max_{1\le i\le n} (-y_i)\,, \\ & \max_{1\le j\le n} (a_{ij}+x_j) = y_i, \quad \max_{1\le j\le n} (b_{ij}+x_j) \le x_i, \\ & y_i \le h_i, \quad i=1,\dots,n. \end{aligned} \tag{6.7}$$

Обратим внимание, что задачу (6.7) можно легко представить в терминах полуполя $\mathbb{R}_{\max,+}$, так как она включает только операции определения максимума, сложения и вычитания. Представим уравнения и неравенства ограничений в терминах полуполя $\mathbb{R}_{\max,+}$

$$\bigoplus_{j=1}^{n} a_{ij}x_j \le x_i, \quad \bigoplus_{j=1}^{n} b_{ij}x_j = y_i, \quad y_i \le h_i, \quad i=1,\dots,n. \tag{6.8}$$

Определим следующие обозначения

$$\boldsymbol{A}=(a_{ij}), \quad \boldsymbol{B}=(b_{ij}), \quad \boldsymbol{x}=(x_i), \quad \boldsymbol{y}=(y_i), \quad \boldsymbol{h}=(h_i). \tag{6.9}$$

Запишем целевую функцию в векторной форме

$$\bigoplus_{i=1}^{n} y_i \bigoplus_{j=1}^{n} y_j = \mathbf{1}^T \boldsymbol{y}\boldsymbol{y}^{-}\mathbf{1}. \tag{6.10}$$

Представим ограничения задачи в виде векторных уравнений и неравенств

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y}, \quad \boldsymbol{B}\boldsymbol{x}\le\boldsymbol{x}, \quad \boldsymbol{y}\le\boldsymbol{h}, \tag{6.11}$$

и запишем задачу в векторном виде

$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{y}} \ & \mathbf{1}^T \boldsymbol{y}\boldsymbol{y}^- \mathbf{1} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \quad \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{y} \le \boldsymbol{h}. \end{aligned} \tag{6.12}$$

Представим решение задачи (6.12)

Теорема 14 ([39]). *Предположим, что матрицы* $\boldsymbol{A}$ *и* $\boldsymbol{B}$ *– такие, что матрица* $\boldsymbol{A}$ *является регулярной,* $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$*, а матрица* $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^*$ *имеет регулярные столбцы* $\boldsymbol{d}_i = (d_{ij})$*, где* $\boldsymbol{B}^*$ *– матрица Клини.*

Тогда максимум в задаче (6.12) *равен* $\Delta = \mathbf{1}^T \boldsymbol{D}\boldsymbol{D}^- \mathbf{1}$ *и достигается тогда, когда* $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$ *и* $\boldsymbol{y} = \boldsymbol{D}\boldsymbol{u}$*, где* $\boldsymbol{u} = (u_j)$ *– любой вектор с компонентами*

$$u_k = \alpha d_{sk}^{-1}, \quad u_j \le \alpha d_{sj}^{-1}, \quad j \ne k, \tag{6.13}$$

при условии, что

$$\alpha \le \min_{1 \le j \le n} d_{sj} (\boldsymbol{h}^- \boldsymbol{d}_j)^-, \quad k = \arg \max_{1 \le j \le n} \mathbf{1}^T \boldsymbol{d}_j \boldsymbol{d}_j^- \mathbf{1}, \quad s = \arg \max_{1 \le j \le n} d_{jk}^{-1}. \tag{6.14}$$

Доказательство. Подставляя выражение $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ в целевую функцию и ограничение получим задачу относительно неизвестного вектора $\boldsymbol{x}$ в следующем виде

$$\begin{aligned} \max_{\boldsymbol{x}} \ & \mathbf{1}^T \boldsymbol{A}\boldsymbol{x} (\boldsymbol{A}\boldsymbol{x})^- \mathbf{1}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{h}. \end{aligned} \tag{6.15}$$

Полученная задача может быть сведена к задаче (6.1) при условии, что $\boldsymbol{C} = \boldsymbol{A}$, $\boldsymbol{p} = \mathbf{1}$ и $\boldsymbol{q} = (\mathbf{1}^T \boldsymbol{A})^-$. Проверим выполнение условий теоремы 13. Для матрицы $\boldsymbol{B}$ выполняется условие существования матрицы Клини $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$, все столбцы матрицы $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^*$ являются регулярными и матрица $\boldsymbol{C}$ регулярна. Введем вспомогательный вектор $\boldsymbol{r} = (\boldsymbol{h}^- \boldsymbol{A}\boldsymbol{B}^*)^- = (\boldsymbol{h}^- \boldsymbol{D})^-$. Поскольку векторы $\boldsymbol{p}$ и $\boldsymbol{q}$ являются регулярными, делавем вывод, что указанные условия выполняются.

Воспользовавшись теоремой 13 получим, что максимум в задаче равен $\Delta = \mathbf{1}^T \boldsymbol{D}\boldsymbol{D}^- \mathbf{1}$ и достигается тогда и только тогда, когда $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$, где $\boldsymbol{u}$ – любой вектор с компонентами (6.13), которые удовлетворяют условиям (6.14).

6.3 Оптимизация вредного медицинского вмешательства

К сожалению многие необходимые медицинские процедуры наносят вред здоровью как медицинского персонала, так и пациента. В качестве примеров подобного медицинского вмешательства можно привести, например, снятие рентгенографических снимков, компьютерную и магнитно-резонансную томографию, различные виды операционного вмешательства и многие виды анестезии. При составлении графика подобных процедур крайне желательно максимально разнести их во времени. Учитывая представленную проблему, при составлении графика оказать имеет смысл максимизировать разброс времени завершения процедур.

Введем вспомогательные обозначения.

x_i – время начала процедуры i;

y_i – время завершения процедуры i;

b_{ij} – минимальный период времени между началом процедуры $j = 1, \ldots, n$ и началом процедуры i. Такие ограничения могут возникнуть при составлении графика операций или при повторном контрастном КТ;

c_{ij} – минимальный промежуток времени между началом процедуры $j = 1, \ldots, n$ и завершением процедуры i. Подобные ограничения могут потребоваться при необходимости минимизировать воздействие рентгеновского или радиоактивного излучения на медперсонал;

h_i – наиболее позднее возможное время проведения процедуры.

Представленная задача – частный случай задачи (6.7), которая может быть представлена в терминах полуполя $\mathbb{R}_{\max,+}$ в виде (6.12).

Опишем работу условного медцентра, которому необходимо организовать проведение $n = 3$ вредных медицинских процедур. Ограничения на время при-

ема задаются следующими матрицами и векторами:

$$\boldsymbol{B} = \begin{pmatrix} \mathbb{0} & -2 & 1 \\ 0 & \mathbb{0} & 2 \\ -1 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 4 & 1 & 1 \\ 2 & 2 & 0 \\ 0 & 1 & 3 \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} 5 \\ 4 \\ 4 \end{pmatrix}. \tag{6.16}$$

Обратим внимание, что используется та же матрица $\boldsymbol{B}$ что и в предыдущем примере. таким образом нам известно, что $\mathrm{Tr}(\boldsymbol{B}) = \mathbb{1}$. Вычислим матрицы

$$\boldsymbol{B}^* = \begin{pmatrix} 0 & -2 & 1 \\ 1 & 0 & 2 \\ -1 & -3 & 0 \end{pmatrix}, \quad \boldsymbol{D} = \boldsymbol{C}\boldsymbol{B}^* = \begin{pmatrix} 4 & 2 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix}. \tag{6.17}$$

Затем вычислим матрицы

$$\boldsymbol{D}^- = \begin{pmatrix} -4 & -3 & -2 \\ -2 & -2 & -1 \\ -5 & -4 & -3 \end{pmatrix}, \quad \boldsymbol{D}\boldsymbol{D}^- = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}, \tag{6.18}$$

и найдем максимум в задаче, равный $\Delta = \mathbf{1}^T \boldsymbol{D}\boldsymbol{D}^- \mathbf{1} = 2$.

Определим значение индексов k и s и нижнюю границу для параметра α, которые задаются условиями (6.14). Сперва найдем

$$\mathbf{1}^T \boldsymbol{d}_1 \boldsymbol{d}_1^- \mathbf{1} = 2, \quad \mathbf{1}^T \boldsymbol{d}_2 \boldsymbol{d}_2^- \mathbf{1} = 1, \quad \mathbf{1}^T \boldsymbol{d}_3 \boldsymbol{d}_3^- \mathbf{1} = 2. \tag{6.19}$$

Следовательно индекс k может принимать два значения: 1 и 3.

Рассмотрим значение $k = 1$. Принимая во внимание, что

$$d_{11}^{-1} = -4, \quad d_{21}^{-1} = -3, \quad d_{31}^{-1} = -2, \tag{6.20}$$

выбираем $s = 3$. Также проверяется, что при $k = 3$ следует снова взять $s = 3$.

Осталось найти верхнюю границу параметра α. запишем вспомогательный вектор

$$\boldsymbol{h}^- \boldsymbol{D} = \begin{pmatrix} -1 & -2 & 0 \end{pmatrix}, \tag{6.21}$$

и рассмотрим значения

$$d_{31}(\boldsymbol{h}^- \boldsymbol{d}_1)^- = 3, \quad d_{32}(\boldsymbol{h}^- \boldsymbol{d}_2)^- = 3, \quad d_{33}(\boldsymbol{h}^- \boldsymbol{d}_3)^- = 3. \tag{6.22}$$

Поскольку минимум этих величин равен 3, считаем, что $\alpha \leq 3$.

Для получения решения, соответствующего самому позднему возможному времени начала приема, положим $\alpha = 3$. Найдем компоненты вектора $\boldsymbol{u} = (u_1, u_2, u_3)^T$ при условии, что $k = 1$. Применив формулы (6.13) вычисляем

$$u_1 = 1, \quad u_2 \leq 2, \quad u_3 \leq 0. \tag{6.23}$$

Для случая $k = 3$ компоненты вектора $\boldsymbol{u}$ удовлетворяют условиям

$$u_1 \leq 1, \quad u_2 \leq 2, \quad u_3 = 0. \tag{6.24}$$

Выбирая вектор $\boldsymbol{u} = (1, 2, 0)^T$, получаем решение задачи в следующем виде

$$\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \quad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{u} = \begin{pmatrix} 5 \\ 4 \\ 3 \end{pmatrix}. \tag{6.25}$$

Глава 7

Минимизация разброса времени начала работ

В данной главе сформулирована и решена задача составления оптимального календарного плана выполнения работ проекта, которая заключается в минимизации максимального разброса времени начала всех работ проекта при полном наборе ограничений. Предложенный метод решения заключается в представлении задачи планирования на языке тропической математики и последующее ее решение при помощи методов тропической оптимизации. В разделе 7.1 сформулирована и решена задача оптимального планирования. Как и в предыдущих задачах решение заключается в сведении кизвестной задаче тропической оптимизации. Далее, в разделе 7.2 предложен пример использования полученного результата для решения задачи о двухкомпонентной вакцинации.

7.1 Минимизация разброса времени начала работ

В данном разделе предлагается решение задачи составления оптимального плана работ в соответствии с критерием (1.10) минимума максимального разброса времени начала всех работ при ограничениях вида «старт-старт», «старт-финишь», «финиш-старт», «время выпуска», «время прекращения выпуска» и «крайний срок завершения». Представленный метод решения заключается в представлении задачи на языке идемпотентной алгебры в форме, предложенной в разделе 3.1 и последующем ее решении при помощи методов тропической оптимизации. Запишем исследуемую задачу в обозначениях из главы 3 в следующем виде

$$\begin{aligned} \min_{x_i,y_i} \quad & \max_{1\le i\le n} x_i + \max_{1\le i\le n}(-x_i); \\ & \max_{1\le j\le n}(b_{ij}+x_j) \le x_i, \quad \max_{1\le j\le n}(c_{ij}+x_j) = y_i, \\ & \max_{1\le j\le n}(d_{ij}+y_j) \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1,\dots,n. \end{aligned} \tag{7.1}$$

Нетрудно понять, что задача (7.1) может быть записана в форме задачи линейного программирования и решена в численном виде с помощью известных

вычислительных процедур, таких как симплекс-алгоритм [107–109] и алгоритм Кармаркара [110].

Переформулируем задачу минимизации разброса времени начала выполнения работ на языке тропической математики. Заметим, что в терминах идемпотентного полуполя $\mathbb{R}_{\max,+}$ целевая функция

$$\min_{x_i} \max_{1\le i\le n} x_i + \max_{1\le i\le n} (-x_i) \tag{7.2}$$

можно быть записана следующим образом:

$$\min_{x_i} \bigoplus_{1\le i\le n} x_i \bigoplus_{1\le j\le n} x_j^{-1}. \tag{7.3}$$

Представим ограничения

$$\max_{1\le j\le n} (b_{ij} + x_j) \le x_i,\ \max_{1\le j\le n} (c_{ij} + x_j) = y_i,\ \max_{1\le j\le n} (d_{ij} + y_j) \le x_i,\ i = 1, \dots, n, \tag{7.4}$$

в тропическом виде

$$\bigoplus_{1\le j\le n} b_{ij}x_j \le x_i, \quad \bigoplus_{1\le j\le n} c_{ij}x_j = y_i, \quad \bigoplus_{1\le j\le n} d_{ij}y_j \le x_i, \quad i = 1, \dots, n. \tag{7.5}$$

Задача (7.1) может быть сформулирована в тропическом виде:

$$\begin{aligned} \min_{x_i, y_i} \quad & \bigoplus_{1\le i\le n} x_i \bigoplus_{1\le j\le n} x_j^{-1}, \\ & \bigoplus_{1\le j\le n} b_{ij}x_j \le x_i, \quad \bigoplus_{1\le j\le n} c_{ij}x_j = y_i, \\ & \bigoplus_{1\le j\le n} d_{ij}y_j \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1, \dots, n. \end{aligned} \tag{7.6}$$

Определим следующие векторные обозначения

$$\boldsymbol{B} = (b_{ij}),\ \boldsymbol{C} = (c_{ij}),\ \boldsymbol{D} = (d_{ij}),\ \boldsymbol{x} = (x_i),\ \boldsymbol{f} = (f_i),\ \boldsymbol{g} = (g_i),\ \boldsymbol{h} = (h_i) \tag{7.7}$$

и запишем задачу оптимизации

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\mathbf{1}\mathbf{1}^T\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} = \boldsymbol{y}, \\ & \boldsymbol{D}\boldsymbol{y} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}, \quad \boldsymbol{y} \le \boldsymbol{f}. \end{aligned} \tag{7.8}$$

Сформулируем и докажем следующую теорему

Теорема 15. *Пусть* $\boldsymbol{B}$ *и* $\boldsymbol{D}$ *– матрицы, а* $\boldsymbol{C}$ *– регулярная по столбцам матрица такие, что для матрицы* $\boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC}$ *выполняется условие* $\mathrm{Tr}(\boldsymbol{R}) \leq \mathbb{1}$. *Пусть* $\boldsymbol{g}$ *– вектор, а* $\boldsymbol{f}$ *и* $\boldsymbol{h}$ *– регулярные векторы такие, что для вектора* $\boldsymbol{s}^T = \boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-$ *выполняется условие* $\boldsymbol{s}^T \boldsymbol{R}^* \boldsymbol{g} \leq \mathbb{1}$.

Тогда минимальное значение целевой функции в задаче (7.8) *равно*

$$\theta = \|\boldsymbol{R}^*\| \oplus \bigoplus_{0 \leq i+j \leq n-2} \|\boldsymbol{s}^T \boldsymbol{R}^i\| \|\boldsymbol{R}^j \boldsymbol{g}\|, \tag{7.9}$$

а все регулярные решения имеют вид

$$\boldsymbol{x} = \boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{G} = \boldsymbol{R}^* \oplus \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{R}^i \mathbf{1}\mathbf{1}^T \boldsymbol{R}^j, \tag{7.10}$$

где $\boldsymbol{u}$ *– любой регулярный вектор, который удовлетворяет условиям*

$$\boldsymbol{g} \leq \boldsymbol{u} \leq (\boldsymbol{s}^T \boldsymbol{G})^-. \tag{7.11}$$

Доказательство. Поскольку $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$, задачу можно представить следующим образом

$$\begin{aligned} \min_{\boldsymbol{x}} \ & \boldsymbol{x}^- \mathbf{1}\mathbf{1}^T \boldsymbol{x}, \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{7.12}$$

Объединим неравенства $\boldsymbol{x} \leq \boldsymbol{h}$ и $\boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}$ в одно $\boldsymbol{x} \leq (\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)^- = (\boldsymbol{s}^T)^-$.

Обратим внимание, что исходная задача может быть записана в следующем виде

$$\begin{aligned} \min_{\boldsymbol{x}} \ & \boldsymbol{x}^- \mathbf{1}\mathbf{1}^T \boldsymbol{C}\boldsymbol{x}; \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq (\boldsymbol{s}^T)^-. \end{aligned} \tag{7.13}$$

Полученная задача является частным случаем задачи 3.1, где $\boldsymbol{p} = \boldsymbol{q} = \mathbf{1}$, а в вместо матрицы $\boldsymbol{B}$ и вектора $\boldsymbol{h}$ используются соответственно матрица $\boldsymbol{R}$ и вектор $(\boldsymbol{s}^T)^-$.

Обратим внимание, что так как значение $\boldsymbol{q}^T \boldsymbol{p} = \mathbf{1}^T \mathbf{1} = \mathbb{1} > \mathbb{0}$ выполняются условия теоремы 8.

Применяя теорему 8 и принимая во внимания, что, по определению нормы матрицы,

$$\mathbf{1}^T \boldsymbol{C}\boldsymbol{R}^* \mathbf{1} = \|\boldsymbol{C}\boldsymbol{R}^*\|, \tag{7.14}$$

и, по определению нормы вектора,

$$\boldsymbol{h}^{-}\boldsymbol{R}^{i}\mathbf{1} = \|\boldsymbol{h}^{-}\boldsymbol{R}^{i}\|, \quad \mathbf{1}^{T}\boldsymbol{C}\boldsymbol{R}^{j}\boldsymbol{g} = \|\boldsymbol{C}\boldsymbol{R}^{j}\boldsymbol{g}\|, \tag{7.15}$$

получим рассматриваемое утверждение.

Для иллюстрации полученного результата представим пример решения практической задачи, связанной с проведением вакцинации.

7.2 Оказание помощи раненным в случае чрезвычайной ситуации

Смоделируем мероприятия по оказанию помощи раненным в случае чрезвычайной ситуации (например техногенной или природной катастрофы). В подобных обстоятельствах от структур, занимающихся спасением пострадавших требуется скорейшее оказание первой помощи раненным и последующая их отправка в безопасный медицинский стационар, где им может быть оказана дальнейшая помощь. Учитывая представленную проблему, при составлении графика оказания помощи требуется как можно скорее оказать первую помощь пострадавшим,. Тем не менее тяжесть ранений может быть очень разной и отправка в стационар более тяжелых раненых может превалировать над оказанием первой помощи легко раненым.

Введем следующие вспомогательные обозначения.

x_i – время оказания первой помощи пострадавшему i;

y_i – время отправки пострадавшего i в безопасный стационар;

b_{ij} – минимальный временной интервал между оказанием первой помощи пострадавшему $j = 1, \ldots, n$ и оказанием первой помощи пострадавшему i (возникает, например, из-за дефицита необходимых специалистов или оборудования. Также может быть обусловлено временем, требующимся, чтобы добраться до пострадавшего) ;

c_{ij} – минимальный временной интервал между оказанием помощи пострадавшему $j = 1, \ldots, n$ и отправкой пострадавшего i (обусловлен, например, одной и той же группой крови у пострадавших);

d_{ij} – минимальный период времени между отправкой в стационар пациента $j = 1, \ldots, n$ и оказанием помощи пациенту i (подобные ограничения могут быть связаны со срочностью оказания помощи пострадавшему j и относительно хорошим состоянием пострадавшего i);

g_i – самое раннее возможное время оказания помощи пострадавшему i;

h_i – самое позднее возможное время оказания помощи пострадавшему i;

f_i – самое позднее допустимое время отправки в стационар пострадавшего i.

Исследуемая задача является частным случаем задачи (7.1), которая может быть переформулирована в терминах полуполя $\mathbb{R}_{\max,+}$ в форме задачи (7.8).

Представим задачу оптимизации работы условного пункта оказания помощи раненым, для которого требуется составить расписание помощи для $n = 3$ пострадавших. Временные ограничения задаются следующими матрицами и векторами:

$$\boldsymbol{B} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -2 & -3 & 0 \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} -4 & -5 & -6 \\ \mathbb{0} & -4 & -7 \\ -5 & \mathbb{0} & -4 \end{pmatrix}, \quad (7.16)$$

$$\boldsymbol{g} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix} \quad \boldsymbol{f} = \begin{pmatrix} 6 \\ 6 \\ 6 \end{pmatrix}. \quad (7.17)$$

Проверим выполнение условий теоремы 15. Обратим внимание, что в задаче используются те же матрицы, что и в задаче о наиболее быстром проведении вакцинации. Таким образом мы знаем, что условия $\mathrm{Tr}(\boldsymbol{R}) \leq \mathbb{1} = 0$ и $\boldsymbol{s}^T \boldsymbol{R}^* \boldsymbol{g} = (\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{R}^* \boldsymbol{g} = -1 < \mathbb{1}$ выполняются.

Найдем минимальный разброс θ времени начала работ. Сначала вычислим вспомогательные векторы:

$$\boldsymbol{s}^T = \boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^- = \begin{pmatrix} -2 & -3 & -2 \end{pmatrix}, \ \boldsymbol{s}^T \boldsymbol{R} = \begin{pmatrix} -2 & -3 & -1 \end{pmatrix}, \ \boldsymbol{Rg} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}. \tag{7.18}$$

Найдем следующие нормы матриц и векторов

$$\|\boldsymbol{R}^*\| = 2, \quad \|\boldsymbol{s}^T\| = -2, \quad \|\boldsymbol{s}^T \boldsymbol{R}\| = -1, \quad \|\boldsymbol{g}\| = 0, \quad \|\boldsymbol{Rg}\| = 2. \tag{7.19}$$

Найдем значение параметра θ по формуле

$$\theta = \|\boldsymbol{B}^*\| \oplus \|\boldsymbol{s}^T\|\|\boldsymbol{g}\| \oplus \|\boldsymbol{s}^T \boldsymbol{R}\|\|\boldsymbol{g}\| \oplus \|\boldsymbol{s}^T\|\|\boldsymbol{Rg}\| \oplus \|\boldsymbol{s}^T \boldsymbol{R}\|\|\boldsymbol{Bg}\| = 2. \tag{7.20}$$

Представим все решения $\boldsymbol{x}$ задачи в параметрической форме. Найдем вспомогательные матрицы:

$$\boldsymbol{R}\mathbf{1}\mathbf{1}^T = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{1}\mathbf{1}^T \boldsymbol{R} = \begin{pmatrix} 0 & -1 & 2 \\ 0 & -1 & 2 \\ 0 & -1 & 2 \end{pmatrix}. \tag{7.21}$$

Запишем следующие матрицы и вектор:

$$\boldsymbol{G} = \boldsymbol{R}^* \oplus (-2)(\mathbf{1}\mathbf{1}^T \oplus \boldsymbol{R}\mathbf{1}\mathbf{1}^T \oplus \mathbf{1}\mathbf{1}^T \boldsymbol{R}) = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix},$$

$$\boldsymbol{CG} = \begin{pmatrix} 4 & 3 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix}, \qquad (\boldsymbol{s}^T \boldsymbol{G})^- = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \tag{7.22}$$

Наконец выпишем все решения $\boldsymbol{x} = (x_1, x_2, x_3)^T$ при помощи вектора параметров $\boldsymbol{u} = (u_1, u_2, u_3)^T$ в следующем виде:

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix} \boldsymbol{u}, \qquad \boldsymbol{y} = \begin{pmatrix} 4 & 3 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix} \boldsymbol{u}, \qquad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \le \boldsymbol{u} \le \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \tag{7.23}$$

Обратим внимание, что столбцы матриц $\boldsymbol{G}$ и $\boldsymbol{CG}$ являются коллинеарными и перепишем их следующим образом

$$\boldsymbol{G} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -1 & -2 \end{pmatrix} \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix}, \quad \boldsymbol{CG} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix}. \tag{7.24}$$

определим вспомогательный вектор $\boldsymbol{v}$

$$\boldsymbol{v} = \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix} \boldsymbol{u}, \quad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \leq \boldsymbol{u} \leq \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \boldsymbol{v} \leq \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{7.25}$$

и с его помощью представим решение

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -1 & -2 \end{pmatrix} \boldsymbol{v}, \qquad \boldsymbol{y} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \end{pmatrix} \boldsymbol{v}, \qquad \begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \boldsymbol{v} \leq \begin{pmatrix} 2 \\ 3 \end{pmatrix}. \tag{7.26}$$

Запишем полученный результат в терминах обычной математики

$$\begin{aligned} &x_1 = \max(v_1, v_2 - 1), && x_2 = \max(v_1 + 1, v_2),\ x_3 = \max(v_1 - 1, v_2 - 2), \\ &y_1 = \max(v_1 + 4, v_2 + 3), && y_2 = \max(v_1 + 3, v_2 + 2),\ y_3 = \max(v_1 + 2, v_2 + 1), \\ & && 0 \leq v_1 \leq 2,\ 0 \leq v_2 \leq 2. \end{aligned} \tag{7.27}$$

Глава 8

Максимизация разброса времени начала работ

Изучим задачу составления оптимального календарного плана выполнения работ проекта, которая заключается в максимизации разброса времени начала выполнения работ проекта при заданных ограничениях. В разделе 8.2 предлагается решение рассматриваемой задачи сетевого планирования, которое опирается на результаты, представленные в разделе 8.1. В разделе 8.3 рассматривается практический пример использования полученного результата для оптимизации работы поликлиники во время карантина.

8.1 Решение задачи оптимизации с ограничениями

Предположим, что заданы матрицы $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{X}^{n\times n}$ и векторы $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{g} \in \mathbb{X}^n$. Изучим следующую задачу оптимизации

$$\begin{aligned} \max_{\boldsymbol{x}} \ & \boldsymbol{q}^{-}\boldsymbol{x}\,(\boldsymbol{A}\boldsymbol{x})^{-}\boldsymbol{p}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x}. \end{aligned} \tag{8.1}$$

Представим теорему, предлагающую решение рассматриваемой задачи.

Теорема 16 ([39]). *Пусть матрицы $\boldsymbol{A}$ и $\boldsymbol{B}$ – такие, что $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$ и матрица $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^*$ имеет регулярные столбцы $\boldsymbol{d}_j = (d_{ij})$, где $\boldsymbol{B}^* = (\boldsymbol{b}_j^*)$ – матрица Клини. Предположим, что векторы $\boldsymbol{p} = (p_j)$ и $\boldsymbol{q} = (q_j)$ регулярны.*

Тогда максимум в задаче (8.1) *равен $\Delta = \boldsymbol{q}^{-}\boldsymbol{B}^*\boldsymbol{D}^{-}\boldsymbol{p}$ и достигается тогда и только тогда, когда $\boldsymbol{x} = \boldsymbol{B}^*\boldsymbol{u}$, где $\boldsymbol{u} = (u_j)$ – любой вектор с компонентами*

$$u_k = \alpha d_{sk}^{-1} p_s, \quad g_j \leq u_j \leq \alpha d_{sj}^{-1} p_s, \quad j \neq k, \tag{8.2}$$

при условии, что

$$\alpha \geq \max_{1\leq j\leq n} g_j d_{sj} q_s^{-1}, \quad k = \arg\max_{1\leq j\leq n} \boldsymbol{q}^{-}\boldsymbol{b}_j^*\boldsymbol{d}_j^{-}\boldsymbol{p}, \quad s = \arg\max_{1\leq j\leq n} d_{jk}^{-1} p_j. \tag{8.3}$$

Доказательство. Поскольку для неравенства $\boldsymbol{Bx} \oplus \boldsymbol{g} \leq \boldsymbol{x}$ выполняется условие теоремы 3, мы можем заменить неравенство на его решение $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$, где $\boldsymbol{u}$ – любой регулярный вектор такой, что $\boldsymbol{u} \geq \boldsymbol{g}$.

Подставив решение неравенства и заменив $\boldsymbol{AB}^*$ на $\boldsymbol{D}$ получим новую задачу относительно неизвестного вектора $\boldsymbol{u}$:

$$\begin{gathered} \max_{\boldsymbol{u}} \ \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{u} \left(\boldsymbol{D}\boldsymbol{u}\right)^- \boldsymbol{p}, \\ \boldsymbol{u} \geq \boldsymbol{g}. \end{gathered} \tag{8.4}$$

Рассмотрим данную задачу без ограничения $\boldsymbol{u} \geq \boldsymbol{g}$. Обратим внимание, что матрица $\boldsymbol{D} = \boldsymbol{AB}^*$ имеет регулярные столбцы, а из регулярности вектора $\boldsymbol{q}$ следует, что вектор $(\boldsymbol{q}^- \boldsymbol{B}^*)^-$ регулярный. Воспользовавшись результатом теоремы 7, получим, что максимум в задаче равен $\Delta = \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{D}^- \boldsymbol{p}$ и достигается тогда и только тогда, когда вектор $\boldsymbol{u}$ имеет компоненты

$$u_k = \alpha d_{sk}^{-1} p_s, \quad u_j \leq \alpha d_{sj}^{-1} p_s, \quad j \neq k, \tag{8.5}$$

для всех скаляров $\alpha > \mathbb{0}$ и индексов k и s, которые определяются следующими условиями

$$k = \arg \max_{1 \leq j \leq n} \boldsymbol{q}^- \boldsymbol{b}_j^* \boldsymbol{d}_j^- \boldsymbol{p}, \quad s = \arg \max_{1 \leq j \leq n} d_{jk}^{-1} p_j. \tag{8.6}$$

Найдем, при каких ограничениях на параметр α вектор $\boldsymbol{u}$ будет удовлетворять неравенству $\boldsymbol{u} \geq \boldsymbol{g}$. Воспользовавшись формулой (8.5), выпишем условия существования подобного вектора $\boldsymbol{u}$ в виде

$$\alpha d_{sj}^{-1} p_s \geq g_j, \quad j = 1, \ldots, n. \tag{8.7}$$

Поскольку все столбцы матрицы $\boldsymbol{D}$ являются регулярными, а вектор $\boldsymbol{p}$ регулярный, скаляр $d_{sj} p_s^{-1}$ не равен нулю. Домножив на $d_{sj} p_s^{-1}$ обе части последнего неравенства и приняв во внимание монотонность умножения, получим неравенство $\alpha \geq g_j d_{sj} p_s^{-1}$.

Тогда при выполнения условия

$$\alpha \geq \max_{1 \leq j \leq n} g_j d_{sj} p_s^{-1}, \tag{8.8}$$

вектор $\boldsymbol{u}$, компоненты которого вычисляется по формулам (8.5)-(8.6), удовлетворяет неравенству $\boldsymbol{u} \geq \boldsymbol{g}$, а вектор $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$ является решением задачи (8.1).

Обратим внимание, что для неразложимой матрицы $\boldsymbol{B}$ матрица Клини $\boldsymbol{B}^*$ не имеет нулевых элементов. Таким образом, если матрица $\boldsymbol{A}$ является регулярной по строкам, то матрица $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^*$ будет иметь регулярные столбцы. Следовательно для выполнения условий теоремы 16 достаточно неразложимости матрицы $\boldsymbol{B}$ и регулярности по строкам матрицы $\boldsymbol{A}$. Кроме того, в случае, если матрица $\boldsymbol{A}$ имеет только регулярные столбцы, столбцы матрицы $\boldsymbol{D}$ будут регулярными при любой матрице $\boldsymbol{B}$.

Найдем вычислительную сложность поиска решения задачи. Основную трудоемкость составляет поиск матрицы Клини, который потребует не более чем $O(n^4)$ скалярных операций. Вычислительная сложность поиска остальных элементов решения не превосходит $O(n^2)$, следовательно общая вычислительная сложность задачи не превосходит $O(n^4)$.

8.2 Максимизация разброса времени начала работ

Представим решение задачи составления оптимального графика выполнения работ в соответствии с критерием максимума разброса времени начала работ (1.10) с ограничениями вида «старт-старт», а также ограничениях на самое раннее время начала выполнения работ

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \max_{1\le i\le n} x_i + \max_{1\le i\le n} (-x_i)\,, \\ & \max\left(\max_{1\le j\le n} (b_{ij} + x_j)\,, g_i\right) \le x_i, \quad i = 1,\dots,n. \end{aligned} \tag{8.9}$$

Обратим внимание, что задача (8.9) включает только операции определения максимума, сложения и вычитания, а значит может быть естественным образом представлена в терминах полуполя $\mathbb{R}_{\max,+}$.

Представим целевую функцию в тропическом виде

$$\bigoplus_{i=1}^{n} x_i \bigoplus_{j=1}^{n} x_j. \tag{8.10}$$

Запишем неравенство-ограничение в терминах полуполя $\mathbb{R}_{\max,+}$

$$\bigoplus_{j=1}^{n} b_{ij}x_j \oplus g_i \le x_i, \qquad i = 1,\dots,n. \tag{8.11}$$

Введем следующие матрично-векторные обозначения

$$\boldsymbol{B} = (b_{ij}), \quad \boldsymbol{g} = (g_i), \quad \boldsymbol{x} = (x_i) \tag{8.12}$$

и представим целевую функцию в виде выражения

$$\bigoplus_{i=1}^{n} x_i \bigoplus_{j=1}^{n} x_j = \mathbf{1}^T \boldsymbol{x} \boldsymbol{x}^{-} \mathbf{1}, \tag{8.13}$$

а ограничения в виде неравенства

$$\boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x}. \tag{8.14}$$

Запишем задачу в векторной форме

$$\begin{aligned} \max_{\boldsymbol{x}} \ & \mathbf{1}^T \boldsymbol{x} \boldsymbol{x}^{-} \mathbf{1}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x}. \end{aligned} \tag{8.15}$$

Сформулируем и докажем следующую теорему

Теорема 17 ([39]). *Пусть матрица $\boldsymbol{B} = (\boldsymbol{b}_j)$ со столбцами $\boldsymbol{b}_j = (b_{ij})$ является неразложимой и $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$. Тогда максимум в задаче* (8.15) *равен $\Delta = \mathbf{1}^T \boldsymbol{B}^* (\boldsymbol{B}^*)^{-} \mathbf{1}$, где $\boldsymbol{B}^*$ – матрица Клини со столбцами $\boldsymbol{b}_j^* = (b_{ij}^*)$, и достигается тогда и только тогда, когда $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$, где $\boldsymbol{u} = (u_j)$ – любой вектор с компонентами*

$$u_k = \alpha (b_{sk}^*)^{-1}, \quad g_j \leq u_j \leq \alpha (b_{sj}^*)^{-1}, \quad j \neq k, \tag{8.16}$$

при условии, что

$$\alpha \geq \max_{1 \leq j \leq n} g_j b_{sj}^*, \quad k = \arg \max_{1 \leq j \leq n} \mathbf{1}^T \boldsymbol{b}_j^* (\boldsymbol{b}_j^*)^{-} \mathbf{1}, \quad s = \arg \max_{1 \leq j \leq n} (b_{jk}^*)^{-1}. \tag{8.17}$$

Доказательство. Рассмотрим задачу (8.1) и положим $\boldsymbol{p} = \mathbf{1}$, $\boldsymbol{q} = \mathbf{1}$ и $\boldsymbol{A} = \boldsymbol{I}$. В таком случае задача (8.15) имеет вид задачи (8.1). Убедимся, что выполняются условия теоремы 16. Легко видеть, что векторы $\boldsymbol{p}$ и $\boldsymbol{q}$ являются регулярными. Так как матрица $\boldsymbol{B}$ неразложима и $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$, матрица $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^* = \boldsymbol{B}^*$ имеет регулярные столбцы. Следовательно условия теоремы выполнены.

Воспользовавшись теоремой, получим, что максимум в задаче равен $\Delta = \mathbf{1}^T \boldsymbol{B}^* (\boldsymbol{B}^*)^{-} \mathbf{1}$ и достигается тогда и только тогда, когда $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$, где $\boldsymbol{u}$ – вектор с компонентами (8.16), удовлетворяющими условию (8.17).

8.3 Оптимизация работы поликлиники во время карантина

Рассмотрим работу поликлиники в период карантина. Даже в условиях пандемии многим людям необходимо осуществление плановых медицинских мероприятий, таких как посещение зубного врача, терапевта, эндокринолога. Одним из способов снижения количества социальных контактов является максимальное разнесение периодов посещения поликлиники гражданами во времени, то есть, максимизация разброса времени начала процедур.

Введем вспомогательные обозначения.

x_i – время начала приема пациента;

b_{ij} – минимальный период времени между началом приема пациента $j = 1, \ldots, n$ и началом приема пациента i;

g_i – наиболее раннее возможное время начала приема.

Представленная задача является частным случаем задачи (8.9), которая может быть естественным образом представлена в терминах идемпотентного полуполя $\mathbb{R}_{\max,+}$ в виде (8.15).

Опишем работу условной поликлиники, для которой необходимо составить график приема $n = 3$ пациентов с ограничениями на время приема, заданными при помощи матрицы и вектора:

$$\boldsymbol{B} = \begin{pmatrix} \mathbb{0} & -2 & 1 \\ 0 & \mathbb{0} & 2 \\ -1 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}. \tag{8.18}$$

Проверим, что выполняется условие $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$ и найдем матрицу Клини $\boldsymbol{B}^*$. Сначала вычислим матрицы

$$\boldsymbol{B}^2 = \begin{pmatrix} 0 & \mathbb{0} & 0 \\ 1 & -2 & 1 \\ \mathbb{0} & -3 & 0 \end{pmatrix}, \quad \boldsymbol{B}^3 = \begin{pmatrix} -1 & -2 & 1 \\ 0 & -1 & 2 \\ -1 & \mathbb{0} & -1 \end{pmatrix}. \tag{8.19}$$

Затем найдем значение идемпотеного аналога определителя $\mathrm{Tr}(\boldsymbol{B}) = \mathrm{tr}\,\boldsymbol{B} \oplus \mathrm{tr}\,\boldsymbol{B}^2 \oplus \mathrm{tr}\,\boldsymbol{B}^3 = 0 = \mathbb{1}$. Следовательно условия теоремы 17 выполнены. Запишем матрицу Клини

$$\boldsymbol{B}^* = \boldsymbol{I} \oplus \boldsymbol{B} \oplus \boldsymbol{B}^2 = \begin{pmatrix} 0 & -2 & 1 \\ 1 & 0 & 2 \\ -1 & -3 & 0 \end{pmatrix}. \tag{8.20}$$

Зная матрицу Клини $\boldsymbol{B}^*$, найдем матрицы

$$(\boldsymbol{B}^*)^- = \begin{pmatrix} 0 & -1 & 1 \\ 2 & 0 & 3 \\ -1 & -2 & 0 \end{pmatrix}, \quad \boldsymbol{B}^*(\boldsymbol{B}^*)^- = \begin{pmatrix} 0 & -1 & 1 \\ 2 & 0 & 3 \\ -1 & -2 & 0 \end{pmatrix}, \tag{8.21}$$

и вычислим максимум $\Delta = \mathbf{1}^T \boldsymbol{B}^*(\boldsymbol{B}^*)^- \mathbf{1} = 3$ целевой функции задачи.

Затем найдем индексы k, s и нижнюю границу параметра α, которые определяются при помощи условиями (8.17). Сперва вычислим

$$\mathbf{1}^T \boldsymbol{b}_1^*(\boldsymbol{b}_1^*)^- \mathbf{1} = 2, \quad \mathbf{1}^T \boldsymbol{b}_2^*(\boldsymbol{b}_2^*)^- \mathbf{1} = 3, \quad \mathbf{1}^T \boldsymbol{b}_3^*(\boldsymbol{b}_3^*)^- \mathbf{1} = 2, \tag{8.22}$$

откуда следует, что значение индекса k равно 2. Далее найдем

$$(b_{12}^*)^{-1} = 2, \quad (b_{22}^*)^{-1} = 0, \quad (b_{32}^*)^{-1} = 3, \tag{8.23}$$

и выберем значение $s = 3$.

После вычисления

$$g_1 b_{31}^* = 1, \quad g_2 b_{32}^* = -3, \quad g_1 b_{31}^* = 0, \tag{8.24}$$

находим максимум, равный 1, из чего следует, что $\alpha \geq 1$.

Определим вектор $\boldsymbol{x}$ самых ранних сроков начала работ, соответствующих значению параметра $\alpha = 1$. Сперва найдем границы (8.16) для всех компонентов вектора $\boldsymbol{u} = (u_1, u_2, u_3)^T$ при условии, что $\alpha = 1$. Проведя соответствующие вычисления получим

$$u_1 = 2, \quad u_2 = 4, \quad 0 \leq u_3 \leq 1. \tag{8.25}$$

Предположим, что $u_3 = 0$ и по вектору $\boldsymbol{u} = (2, 4, 0)^T$ найдем вектор

$$\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u} = \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix}. \tag{8.26}$$

Принимая во внимание, что при $u_3 = 1$ решение останется прежним, будем считать, что наиболее раннее время начала работ задается при помощи равенств

$$x_1 = 2, \quad x_2 = 4, \quad x_3 = 1. \tag{8.27}$$

Заключение

Таким образом в работе были получены следующие результаты.

- Изучена задача максимизации максимальной продолжительности проекта при ограничениях вида «старт-старт», «старт-финиш», «финиш-старт», «время выпуска», «время прекращения выпуска» и «крайний срок завершения».

- Рассмотрена задача минимизации выхода максимального отклонения от директивных сроков выполнения работ при ограничениях вида «старт-старт», «старт-финиш», «финиш-старт», «время выпуска», «время прекращения выпуска»и «крайний срок завершения».

- Рассмотрена задача минимизации максимального разброса времени завершения работ проекта при ограничениях вида «старт-старт» и «старт-финиш».

- Исследована задача максимизации максимального разброса времени начала выполнения работ проекта при ограничениях вида «старт-старт» и «время выпуска».

- Исследована задача минимизации максимального разброса времени начала работ проекта при ограничениях типа «старт-старт», «время выпуска» и «крайний срок завершения».

- Изучена задача максимизации максимального разброса времени завершения работ проекта при ограничениях типа «старт-старт», «старт-финиш» и «крайний срок завершения».

- Представленные задачи переформулированы на языке тропической оптимизации.

- Предложены прямые аналитические решения рассматриваемых задач оптимизации, которые могут быть использованы как в практических задачах так и для формального анализа.

- Для матрично-векторных операции идемпотентной алгебры и решений рассматриваемых задач оптимизации представлена их программная реализация.

- Для всех рассматриваемых задач представлены практические промеры их использования для решения реальных практических задач и предложены поясняющие численные примеры.

Список литературы

1. Kelley J., Walker M. Critical-path planning and scheduling // Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM Computer Conference. ACM. 1959. P. 160–173.

2. Saplosky H. The Polaris system development. Cambridge: Harvard University Press, 1972. 272 p.

3. Pinedo M. Scheduling. 3rd ed. New York: Springer, 2008.

4. Vanhoucke M. Project Management with Dynamic Scheduling. 2nd ed. Berlin: Springer, 2012.

5. Blazewicz J., Ecker K., Pesch E. et al. Handbookon Scheduling. In: International Handbooks on Information Systems. Cham, Springer, 2019.

6. Маслов В.П., Колокольцов В.Н. Идемпотентный анализ и его применение в оптимальном управлении. М.: Физматлит, 1994. 144 с.

7. Кривулин Н.К. Методы идемпотентной алгебры в задачах моделирования и анализа сложных систем. Издательство Санкт-Петербургского государственного университета, 2009. 256 с.

8. Cuninghame-Green R. Minimax algebra and applications // Advances in Imaging and Electron Physics. 1994. Vol. 90. P. 1–121.

9. Zimmermann U. Linear and combinatorial optimization in ordered algebraic structures. Amsterdam: Elsevier, 2011. 390 p.

10. Baccelli F.L., Cohen G., Olsder G.J., Quadrat J.P. Synchronization and linearity: an algebra for discrete event systems. Chichester: Wiley, 1992. 514 p.

11. Grigoriev D., Shpilrain V. Tropical cryptography // Communications in Algebra. 2014. Vol. 42, no. 6. P. 2624–2632.

12. Zimmermann K. Disjunctive optimization, max-separable problems and extremal algebras // Theoretical Computer Science. 2003. Vol. 293, no. 1. P. 45–54.

13. Tharwat A., Zimmermann K. One class of separable optimization problems: solution method, application // Optimization. 2010. Vol. 59, no. 5. P. 619–625.

14. Кривулин Н.К., Плотников П.В. Использование тропической оптимизации для решения минимаксных задач размещения с прямоугольной метрикой на прямой // Вестник Санкт-Петербургского университета. Серия 1. Математика. Механика. Астрономия. 2016. Т. 3, № 4. С. 602–614.

15. Cuninghame-Green R. Describing industrial processes with interference and approximating their steady-state behaviour // Journal of the Operational Research Society. 1962. Vol. 13, no. 1. P. 95–100.

16. Николаев Д.А. Моделирование и управление движением агента в неопределенной внешней среде методами идемпотентной алгебры // Управление большими системами: сборник трудов. 2012. № 40. С. 311–328.

17. Матвеенко В.Д. Оптимальные траектории схемы динамического программирования и экстремальные степени неотрицательных частиц // Дискретная математика. 1990. Т. 2, № 1. С. 59–71.

18. Кривулин Н.К., Романовский И.В. Решение задач математического программирования с использованием методов тропической оптимизации // Вестник Санкт-Петербургского университета. Серия 1. Математика. Механика. Астрономия. 2017. Т. 4, № 3. С. 448–458.

19. Cuninghame-Green, R.A. Projections in minimax algebra // Mathematical Programming. 1976. Vol. 10, no. 1. P. 111–123.

20. Krivulin N. A multidimensional tropical optimization problem with a non-linear objective function and linear constraints // Optimization. 2015. Vol. 64, no. 5. P. 1107–1129.

21. Butkovic P., Aminu A. Introduction to max-linear programming // IMA Journal of Management Mathematics. 2009. Vol. 20, no. 3. P. 233–249.

22. Heidergott B., Olsder G. Max-plus at Work: Modeling and Analysis of Synchronized Systems. Princeton Series in Applied Mathematics. Princeton: Princeton University Press, 2006. 226 p.

23. Krivulin N. A constrained tropical optimization problem: Complete solution and application example // Tropical and Idempotent Mathematics and Applications. 2014. Vol. 616. P. 163–177.

24. Krivulin N. Complete solution of a constrained tropical optimization problem with application to location analysis // International Conference on Relational and Algebraic Methods in Computer Science / Springer. 2014. P. 362–378.

25. Кривулин Н.К. Экстремальное свойство собственного значения неразложимых матриц в идемпотентной алгебре и решение задачи размещения Ролса // Вестник Санкт-Петербургского университета. Серия 1. Математика. Механика. Астрономия. 2011. Т. 44, № 4. С. 272–284.

26. Butkovič P. Max-linear systems: theory and algorithms. Springer Science & Business Media, 2010.

27. Grigoriev D. On a tropical dual Nullstellensatz // Advances in Applied Mathematics. 2012. Vol. 48, no. 2. P. 457–464.

28. Grigoriev D., Koshevoy G. Complexity of tropical Schur polynomials // Journal of Symbolic Computation. 2016. Vol. 74. P. 46–54.

29. Krivulin N. Tropical optimization problems // Advances in Economics and Optimization: Collected Scientific Studies Dedicated to the Memory of L. V. Kantorovich. 2014. P. 195–214.

30. Krivulin N. Extremal properties of tropical eigenvalues and solutions to tropical optimization problems // Linear Algebra and its Applications. 2015. Vol. 468. P. 211–232.

31. Zimmermann K. Interval linear systems and optimization problems over max-algebras // Linear Optimization Problems with Inexact Data. 2009. P. 165–193.

32. Krivulin N. Explicit solution of a tropical optimization problem with application to project scheduling // Mathematical Methods and Optimization Techniques in Engineering. 2013. Vol. 20, no. 3. P. 39–45.

33. Krivulin N. A maximization problem in tropical mathematics: A complete solution and application examples // Informatica. 2016. Vol. 27, no. 3. P. 587–606.

34. Krivulin N. Tropical optimization problems in project scheduling // MISTA 2015:Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications. 2015. P. 492–506.

35. Krivulin N. Direct solution to constrained tropical optimization problems with application to project scheduling // Computational Management Science. 2017. Vol. 14, no. 1. P. 91–113.

36. Krivulin N. Tropical optimization problems with application to project scheduling with minimum makespan // Annals of Operations Research. 2017. Vol. 256, no 1. P. 75–92.

37. Krivulin N. Tropical optimization problems in time-constrained project scheduling // Optimization. 2017. Vol. 66, no. 2. P. 205–224.

38. Кривулин Н.К., Губанов С.А. Решение задачи сетевого планирования на основе методов тропической оптимизации // Вестник Санкт-Петербургского университета. Серия 10. Прикладная математика. Информатика. Процессы управления. 2016. № 3. С. 62–72.

39. Кривулин Н.К., Губанов С.А. Использование методов тропической оптимизации в задачах сетевого планирования // Вестник Санкт-Петербургского университета. Серия 10. Прикладная математика. Информатика. Процессы управления. 2017. № 4. С. 384–397.

40. Кривулин Н.К., Губанов С.А. Алгебраическое решение задачи оптимального планирования сроков проекта в управлении проектами // Вестник Санкт-Петербургского университета. Математика. Механика. Астрономия. 2021. Т. 8(66), № 1. С. 73–87.

41. Губанов С.А. Алгебраическое решение задач оптимального планирования с учетом директивных сроков начала работ проекта // Вестник Санкт-Петербургского университета. Математика. Механика. Астрономия. 2022. Т. 9(67), № 4. С. 602–611.

42. Кривулин Н.К., Губанов С.А. Решение задачи тропической оптимизации с приложением к сетевому планированию // Междисциплинарные исследования в области математического моделирования и информатики. Материалы 6-й научно-практической internet-конференции. Тольятти 14-15 мая 2015г. № 4. С. 224–227.

43. Кривулин Н.К., Губанов С.А. Использование методов тропической оптимизации для оптимального составления графиков работ // Материалы 7-Й всероссийской научной конференции по проблемам информатики 26-28 апреля 2017г. С. 515–522.

44. Кривулин Н.К., Губанов С.А. Составление календарных графиков выполнения работ с помощью методов тропической оптимизации // Материалы всероссийской научной конференции по проблемам информатики 23-26 апреля 2019г. С. 319–325.

45. Кривулин Н.К., Губанов С.А. Решение задач сетевого планирования на основе методов тропической математики // Модели и методы тропической математики в прикладных задачах экономики и управления. Сборник научных статей. 2014. № 2. С. 99–121.

46. Губанов С.А. Алгебраическое решение задачи планирования в управлении проектами // Полиномиальная компьютерная алгебра. Материалы международной конференции. 2022. С. 28–32.

47. Губанов С.А. Решение задачи оптимального управления проектом с использованием методов тропической оптимизации // Тезисы XXIII Всероссийской конференции молодых учёных по математическому моделированию и информационным технологиям. 2022. С. 50.

48. Кривулин Н.К., Губанов С.А. Решение задачи оптимального планирования проекта методами тропической математики // Математические методы и модели в высокотехнологичном производстве. Сборник тезисов докладов II Международного форума. 2022. С. 366–369.

49. Губанов С.А. Решение задачи составления оптимального календарного графика выполнения работ проекта при помощи методов тропической оптимизации // Современные проблемы машиностроения. Сборник трудов XV Международной научно-технической конференции. 2022. С. 193–194.

50. Demeulemeester E., Herroelen W. Project scheduling: a research handbook. International Series in Operations Research and Management Science. Springer, 2002.

51. T'kindt V., Billaut J.-C. Multicriteria scheduling: theory, models and algorithms. Springer Science & Business Media, 2006.

52. Cuninghame-Green R. Projections in minimax algebra // Mathematical Programming. 1976. Vol. 10, no. 1. P. 111–123.

53. Zimmermann U. Linear and combinatorial optimization in ordered algebraic structures. 1981. Vol. 10.

54. Golan J. Semirings and affine equations over them: theory and applications. Springer Science & Business Media, 2013. Vol. 556.

55. Gondran M., Minoux M. Graphs, dioids and semirings: new models and algorithms. Springer Science & Business Media, 2008. Vol. 41.

56. Pandit S. A new matrix calculus // Journal of the Society for Industrial and Applied Mathematics. 1961. Vol. 9, no. 4. P. 632–639.

57. Ciffler B. Scheduling general production systems using schedule algebra // Naval Research Logistics Quarterly. 1963. Vol. 10, no. 1. P. 237–255.

58. Dedekind R. Über die Theorie der ganzen algebraischen Zahlen // Vorlesungen Ëuber Zahlentheorie. Druck und Verlag Braunschweig, 1894. P. 434–657.

59. Golan, J.S. Semirings and their Applications. Springer Science & Business Media, 1999.

60. Glazek K. A guide to the literature on semirings and their applications in mathematics and information sciences: with complete bibliography. Springer Science & Business Media, 2002.

61. Vandiver H. Note on a simple type of algebra in which the cancellation law of addition does not hold // Bulletin of the American Mathematical Society. 1934. Vol. 40, no. 12. P. 914–920.

62. Stephen C. Kleene. Representation of events in nerve nets and finite automata // Automata Studies. 1956. P. 3–43.

63. Cuninghame-Green R. Describing industrial processes with interference and approximating their steady-state behaviour // Journal of the Operational Research Society. 1962. Vol. 13, no. 1. P. 95–100.

64. Воробьев Н.Н. Экстремальная алгебра матриц // Доклады академии наук СССР. 1963. Т. 152, № 1. С. 24–27.

65. Воробьев Н.Н. Экстремальная алгебра положительных матриц // Elektronische Informationsverarbeitung und Kybernetik. 1967. Т. 3, № 1. С. 39.

66. Воробьев Н.Н. Экстремальная алгебра неотрицательных матриц // Elektronische Informationsverarbeitung und Kybernetik. 1970. Т. 6, № 4-5. С. 303–312.

67. Романовский И.В. Несколько замечаний о функциональном преобразовании Беллмана-Каруша // Вестник ЛГУ. 1962. С. 148–150.

68. Романовский И.В. Асимптотическое поведение процессов динамического программирования с непрерывным множеством состояний // Доклады академии наук СССР / Российская академия наук. Т. 159. 1964. С. 1224–1227.

69. Birkhoff G. Lattice theory // American Mathematical Society Colloquium Publications. Vol. 25. 1948. 311 p.

70. Канторович Л.В. Функциональный анализ в полуупорядоченных пространствах. Гос. изд-во технико-теорет лит-ры, 1950. 548 с.

71. Лунц А.Г. Алгебраические методы анализа и синтеза контактных схем // Известия Российской академии наук. Серия математическая. 1952. Т. 16, № 5. С. 405–426.

72. Поваров Н.Г. Матричные методы анализа релейно-контактных схем по условиям несрабатывания // Автоматика и телемеханика. 1954. Т. 15, № 4. С. 332–335.

73. Shimbel A. Structure in communication nets // Proceedings of the symposium on information networks / Polytechnic Institute of Brooklyn. 1954. P. 119–203.

74. Bellman R., Karush W. On a new functional transform in analysis: the maximum transform: Tech. rep.: System Development Corp Santa Monica CA, 1961.

75. Канторович Л.В. О перемещении масс // Доклады академии наук СССР. 1942. Т. 37, № 7-8. С. 227—-229.

76. Корбут А.А. Экстремальные пространства // Доклады академии наук СССР / Российская академия наук. Т. 164. 1965. С. 1229–1231.

77. Корбут А.А. Экстремальные векторные пространства и их свойства // Elektronische Informationsverarbeitung und Kybernetik. Т. 8/9. 1972. С. 525–536.

78. Дудников П.И., Самборский С.Н. Эндоморфизмы полумодулей над полукольцами с идемпотентной операцией // Известия Российской академии наук. Серия математическая. 1991. Т. 55, № 1. С. 93–109.

79. Maslov V. On a new superposition principle for optimization problem // Séminaire Équations aux dérivées partielles (Polytechnique). 1985. P. 1–14.

80. Маслов В.П. О новом принципе суперпозиции для задач оптимизации // Успехи математических наук. 1987. Т. 42, № 3 (255). С. 39–48.

81. Dobrokhotov S.Y., Kolokoltsov V.N., Maslov V.P. Quantization of the Bellman equation, exponential asymptotics and tunneling // Advances in Soviet Mathematics. 1992. Vol. 13. P. 1–46.

82. Maslov V.P., Samborskii S.N. Stationary Hamilton–Jacobi and Bellman equations (existence and uniqueness of solutions) // Idempotent Analysis. 1992. Vol. 41. P. 87–101.

83. Литвинов Г.Л., Маслов В.П., Шпиз Г.Б. Идемпотентный функциональный анализ. Алгебраический подход // Математические заметки. 2001. Т. 69, № 5. С. 758–797.

84. Litvinov G., Maslov V. The correspondence principle for idempotent calculus and some computer applications // Idempotency. 1998. Vol. 11. P. 420–443.

85. Литвинов Г.Л. Деквантование Маслова, идемпотентная и тропическая математика: краткое введение // Записки научных семинаров ПОМИ. 2005. Т. 326, № 0. С. 145–182.

86. Sergeev S. Minimal elements and cellular closures over the max-plus semiring // Idempotent and Tropical Mathematics and Problems of Mathematical Physics. 2007. Vol. 2. P. 49–52.

87. Gaubert, S., Katz, R.D., Sergeev, S. Tropical linear-fractional programming and parametric mean payoff games // Journal of Symbolic Computation. 2012. Vol. 47, no. 12. P. 1447–1478.

88. Sergeev S. Max-algebraic attraction cones of nonnegative irreducible matrices // Linear Algebra and its Applications. 2011. P. 1736–1757.

89. Mikhalkin G. Amoebas of algebraic varieties and tropical geometry // Different Faces of Geometry. 2004. P. 257–300.

90. Itenberg I., Mikhalkin G., Shustin E. Tropical algebraic geometry. Springer Science & Business Media, 2009. Vol. 35.

91. Brugallé E., Itenberg I., Mikhalkin G., Shaw K. Brief introduction to tropical geometry. 2015. P. 1–75.

92. Казарян М.Э. Тропическая геометрия. Московский центр непрерывного математического образования (МЦНМО), 2012. 43 с.

93. Zimmermann K. Some optimization problems with extremal operations // Mathematical Programming at Berwolfach. 1984. Vol. 22. P. 237–251.

94. Butkovič P, Tam K.P. On some properties of the image set of a max-linear mapping // Tropical and Idempotent Mathematics. 2009. Vol. 495. P. 233–249.

95. Butkovič P., Aminu A. Non-linear programs with max-linear constraints: a heuristic approach // IMA Journal of Management Mathematics. 2012. no. 1. P. 41–66.

96. Hoffman A. J. On abstract dual linear programs // Naval Research Logistics Quarterly. 1963. no. 1. P. 369–373.

97. Zimmermann, K., Gavalec, M. Duality for max-separable problems // Central European Journal of Operations Research. 2012. P. 409–419.

98. Superville L. Various aspects of max-algebra // PhD thesis, The City University of New York. 1978. P. 409–419.

99. Zimmermann K. On max-separable optimization problems // Algebraic and Combinatorial Methods in Operations Research. 1984. Vol. 95. P. 357–362.

100. Zimmermann K. Optimization problems with unimodal functions in max-separabal constraints // Optimization. 1992. no. 1-2. P. 31–41.

101. Cuninghame-Green R. A., Butkovič P. The equation $a \oplus x = b \oplus y$ over (max,+) // Theoretical Computer Science. 2003. no. 1. P. 3–12.

102. Butkovič P. On properties of solution sets of extremal linear programs // Algebraic and Combinatorial Methods in Operations Research. 1984. P. 41–54.

103. Akian M., Gaubert S., A. Guterman A. Tropical polyhedra are equivalent to mean payoff games // International Journal of Algebra and Computation. 2012. Vol. 22, no. 1.

104. Krivulin N. Eigenvalues and eigenvectors of matrices in idempotent algebra // Vestnik St. Petersburg University, Mathematics. 2006. no. 2. P. 72–83.

105. Krivulin N. Solution of generalized linear vector equations in idempotent algebra // Vestnik St. Petersburg University, Mathematics. 2006. no. 1. P. 16–26.

106. Krivulin N. A new algebraic solution to multidimensional minimax location problems with Chebyshev distance // WSEAS Transactions on Mathematics. 2012. no. 7. P. 146–151.

107. Kantorovich L.V. Mathematical methods of organizing and planning production // Management Sci. 1960. Vol. 6, no. 4. P. 366–422.

108. Романовский И.В. Алгоритмы решения экстремальных задач. М.: Наука, 1977. 352 с.

109. Васильев Ф.П., Иваницкий А.Ю. Линейное программирование. 3-е изд. М.: Факториал Пресс, 2008.

110. Karmarkar N. A new polynomial-time algorithm for linear programming // Combinatorica. 1984. Vol. 4, no. 4. P. 373–395.

Приложение А **Программная реализация скалярных операций идемпотентной алгебры**

Сперва рассмотрим программную реализацию необходимых для решения задач тропической оптимизации скалярных операций идемпотентной алгебры. Представленный вариант программной реализации реализован на языке высокого уровня C++.

А.1 Структура программного обеспечения

- Абстрактный класс `Algebra`, реализующий задание идемпотентной алгебры. В классе присутствуют следующие открытые методы:
  - Функция `double oplus(const double& a,const double& b)` определяет операцию идемпотентного сложения. На вход функция принимает скаляры a и b и возвращает значение суммы $a \oplus b$;
  - Функция `double otimes(const double& a,const double& b)` определяет идемпотентное умножение. На вход функция принимает скаляры a и b и возвращает значение произведения $a \otimes b$;
  - Функция `bool order(const double& a, const double& b)` задает отношение частичного порядка. В случае, если $a \leq b$ функция возвращает true, и false иначе;
  - Функция `pseudo_inverse(const double& a)` реализует вычисление псевдообратного элемент На вход функция принимает скаляр a и возвращается a^{-};
  - Функция `double get_zero()` возвращает идемпотентный ноль;
  - Функция `get_unit()` возвращает идемпотентную единицу;
- Класс `R_max_plus` является наследником класса `Algebra` и используется для реализации алгебры $\mathbb{R}_{\max,+}$.

A.2 Листинг программы

```
class Algebra
{
public:
    virtual const double oplus(const double& a,const double& b)= 0;
    virtual const double otimes(const double& a,const double& b)= 0;
    virtual const bool order(const double& a,const double& b)= 0;
    virtual const double pseudo_inverse(const double& a)= 0;
    virtual const double power(const double& x,const double& y)= 0;
    virtual const double get_zero() {return idemp_zero;}
    virtual const double get_unit() {return idemp_unit;}

protected:
    double idemp_zero;
    double idemp_unit;
};

class R_max_plus : public Algebra
{
public:
    R_max_plus()
    {
        idemp_zero = -1.0e+10;
        idemp_unit = 0.0;
    }
    const double oplus(const double& a, const double& b)
    {
        return std::max(a,b);
    }
    const double otimes(const double& a, const double& b)
    {
        return (a + b);
    }
    const bool order(const double& a, const double& b)
    {
        return a <= b;
    }
    const double pseudo_inverse(const double& a)
    {
        return -a;
    }
    const double power(const double& x, const double& y)
    {
        return x * y;
    }
};
```

Приложение Б **Программная реализация матрично-векторных операций в идемпотентной алгебре**

Программное представление рассмотренных выше задач оптимизации также потребует реализации матрично векторных операций. Предложим вариант подобной реализации на языке высокого уровня C++.

Б.1 Структура программного обеспечения

- Функция `std::vector<std::vector<size_t>> get_all_pairs (size_t n_min_k)` возвращает все векторы, состоящие из двух индексов таких, что их сумма не превосходит `n_min_k`;
- Шаблонный класс `Matrix<T>`, использующийся для представления матричных операций над идемпотентной алгеброй. В качестве класса-шаблона используется класс-наследник описанного в приложении А класса `Algebra`. В классе присутствуют следующие открытые методы:
  - `Matrix()` – конструктор по умолчанию;
  - `Matrix(size_t _row,size_t _col)` – конструктор с параметрами. На вход принимается число строк `_row` и число столбцов `_col`.
  - `Matrix& operator = (const Matrix& matrix)` – оператор присваивания
  - Функция `double get_zero()` – возвращает нулевой элемент матрицы;
  - Функция `double get_unit()` – возвращает единичный элемент матрицы;
  - Функция `double get_rows()` – возвращает количество строк матрицы;
  - Функция `double get_cols()` – возвращает количество столбцов матрицы;

– Функция `bool is_quadr()` – возвращает true, если матрица квадратная;

– Функция `void Zero()` – заполняет матрицу нулями (из класса-шаблона);

– Функция `void Identity()` – делает матрицу единичной;

– Функция `void Identity_vec()` – делает вектор (матрицу–столбец) единичным ;

– Оператор `double& operator()(int _row, int _col)` – дает доступ к элементу с индексами `_row` и `_col`;

– Оператор `Matrix operator*(const Matrix& matrix)` – осуществляет перемножение матриц

– Оператор `Matrix operator*(double alpha)` – осуществляет домножение матрицы на скаляр;

– Оператор `Matrix operator+(const Matrix& matrix)` – осуществляет сложение матриц;

– Оператор `Matrix operator^(size_t power)` – обеспечивает возведение матрицы в степень power;

– Функция `Matrix ast()` – вычисляет матрицу Клини;

– Функция `double tr()` – вычисляет оператор tr;

– Функция `double Tr()` – вычисляет оператор Tr;

– Функция `Matrix transpose()` – возвращает транспонированную матрицу;

– Функция `Matrix pseudo_inverse()` – возвращает псевдообратную матрицу;

– Функция `double norm()` – вычисляет норму матрицы

– Функция `bool is_row_regular()` – проверяет, что матрица регулярна по строкам;

- Функция `bool is_col_regular()` – проверяет, что матрица регулярна по столбцам;
- Функция `bool is_regular()` – проверяет, регулярна ли матрица;
- Функция `bool regular_cols()` – проверяет, имеет ли матрица регулярные столбцы;
- Функция `void to_file(std::ofstream& outf)` – осуществляет вывод матрицы в файл;
- Функция `Matrix get_col(size_t j)` – возвращает столбец;
- Функция `bool is_razl()` – проверяет, разложима ли матрица;

Б.2 Листинг программы

```
typedef std::unique_ptr<double[]> DBPT;
std::vector<std::vector<size_t>> get_all_pairs(size_t n_min_k)
{
    std::vector<std::vector<size_t>> res;
    if(n_min_k < 0)
    {
        return res;
    }
    for(size_t i = 0; i <= n_min_k; i++)
        for(size_t j = 0; j <= n_min_k; j++)
            if(i+j <= n_min_k)
            {
                std::vector<size_t> new_res;
                new_res.push_back(i);
                new_res.push_back(j);
                res.push_back(new_res);
            }
    return res;
}

template<class T>
class Matrix
{
private:
    T alg;
    size_t row;
    size_t col;
public:
    double get_zero() {return alg.get_zero();}
    double get_unit() {return alg.get_unit();}


    std::unique_ptr<DBPT[]> elems;
```

```
string errbuf;
//char* errbuf;

Matrix()
{
    row = 0;
    col = 0;
}

~Matrix()
{
}

Matrix(size_t _row,size_t _col)
{
    row = _row;
    col = _col;
    elems.reset(new DBPT[row]);
    for(int i = 0; i < row; i++)
       elems.get()[i].reset(new double[col]);
    Zero();
}

Matrix(Matrix& matrix)   //конструктор копирования
{
    *this = matrix;
}

Matrix& operator = (const Matrix& matrix)
{
    row = matrix.row;
    col = matrix.col;
    elems.reset(new DBPT[row]);
    for(int i = 0; i < row; i++)
        elems.get()[i].reset(new double[col]);
    for(int i = 0; i < row; i++)
        for(int j = 0; j < col; j++)
            (*this)(i,j) = matrix(i, j);
    return *this;
}

double get_rows()
{
    return row;
}

double get_cols()
{
    return col;
}

bool is_quadr()
{
    return row == col;
}
```

```
void Zero()
{
    for(int i = 0; i < row; i++)
        for(int j = 0 ; j < col; j++)
            (*this)(i,j) = alg.get_zero();
}

void Identity()
{
    if (row != col)
        throw "Error identity, matrix is not squared";
    Zero();
    for(int i = 0; i < row; i++)
            (*this)(i,i) = alg.get_unit();
}

void Identity_vec()
{
    if (col != 1)
        throw "Error identity, object is not a vector";
    for(int i = 0; i < row; i++)
            (*this)(i,0) = alg.get_unit();
}

double& operator()(int _row, int _col)
{
    return elems[_row].get()[_col];
}

double operator()(int _row, int _col) const
{
    return elems[_row].get()[_col];
}


Matrix operator*(const Matrix& matrix)
{
    if (col != matrix.row)
        throw "Error * , different matrix dimentions ";

    Matrix res(row, matrix.col);
    res.Zero();

    for(int i = 0 ; i < row ; i++){
        for(int j = 0 ; j < matrix.col; j++)
            for(int k = 0 ; k < col; k++){
                res(i,j)
                = alg.oplus(
                    res(i,j),
                    alg.otimes(
                        this->operator ()(i,k)
                        , matrix(k,j)));
            }
```

```
    }
    return res;
}

Matrix operator*(double alpha)
{
    Matrix res(row, col);
    res.Zero();

    for(int i = 0 ; i < row ; i++){
        for(int j = 0 ; j < col; j++){
                res(i,j) =
                 alg.otimes(this->operator ()(i,j), alpha);
            }
    }
    return res;
}

Matrix operator+(const Matrix& matrix)
{
    if (row != matrix.row || col != matrix.col)
    {
        throw "Error + , different matrix dimentions ";
    }
    Matrix res(row, col);
    for(int i = 0; i < row; i++)
        for(int j=0; j < col; j++)
            res(i,j) =
             alg.oplus(this->operator()(i, j), matrix(i, j));
    return res;
}

Matrix operator^(size_t power)
{
    if (row != col)
        throw "Error in power, matrix is not squared";
    Matrix res(row, col);
    res.Identity();
    for(size_t i = 0; i < power; i++)
    {
        res = res * (*this);
    }
    return res;
}

Matrix ast()
{
    if (row != col)
        throw "Error ast, matrix is not squared";
    if(!alg.order(Tr(), alg.get_unit()))
        throw "Error ast, Tr(A) >= 1";
    Matrix res(row, col);
    for(size_t i = 0; i < row; i++)
    {
        res = res + (*this)^i;
```

```
    }
    return res;
}

double tr()
{
    if (row != col)
        throw "Error identity, matrix is not squared";
    double res = alg.get_zero();
    for(size_t i = 0; i < row; i++)
    {
        res = alg.oplus(res, (*this)(i,i));
    }
    return res;
}

double Tr()
{
    if (row != col)
        throw "Error identity, matrix is not squared";
    double res = alg.get_zero();
    for(size_t i = 0; i < row; i++)
    {
        res = alg.oplus(res, ((*this)^(i+1)).tr());
    }
    return res;
}

Matrix transpose()
{
    Matrix res(col, row);
    for(size_t i = 0; i < row; i++)
    {
        for(size_t j = 0; j < col; j++)
        {
            res(j,i) = (*this)(i,j);
        }
    }
    return res;
}

Matrix pseudo_inverse()
{
    Matrix res(col, row);
    res = transpose();
    for(size_t i = 0; i < res.row; i++)
        for(size_t j = 0; j < res.col; j++)
            res(i,j) = alg.pseudo_inverse(res(i,j));
    return res;
}

double norm()
{
    double res = get_zero();
    for(size_t i = 0; i < row; i++){
```

```
        for(size_t j = 0; j < col; j++)
            res = alg.oplus(this->operator ()(i,j), res);
    }
    return res;
}

void out() const
{
    for(size_t i = 0; i < row; i++)
    {
        for(size_t j = 0; j < col; j++)
        {
            std::cout << (*this)(i,j) << "    ";
        }
        std::cout << std::endl;
    }
}

bool is_row_regular()
{
    for(size_t i = 0; i < row; i++){
        size_t fail_counter = 0;
        for(size_t j = 0; j < col; j++)
        {
            if((*this)(i,j) != alg.get_zero())
                break;
            else
                fail_counter++;
        }
        if(fail_counter == col)
            return false;
    }
    return true;
}

bool is_col_regular()
{
    for(size_t i = 0; i < col; i++){
        size_t fail_counter = 0;
        for(size_t j = 0; j < row; j++)
        {
            if((*this)(i,j) != alg.get_zero())
                break;
            else
                fail_counter++;
        }
        if(fail_counter == row)
            return false;
    }
    return true;
}

bool is_regular()
{
    return is_row_regular() && is_col_regular();
```

```
}

bool regular_cols()
{
    bool res = true;
    Matrix<T> b_j;
    for(size_t j = 0; j < col; j++)
    {
        b_j = this->get_col(j);
        if(!b_j.is_row_regular())
            res = false;
    }
    return res;
}

void to_file(std::ofstream& outf)
{
    for(int i = 0; i < row; i++)
    {
        outf << "\n";
        for(int j=0; j < col; j++)
        {
            if ((*this)(i,j) == get_zero())
            {
                outf  << " z " ;
            }
            else
                if ((*this)(i,j) < 0.0)
                {
                    outf   << (*this)(i,j) << " ";}
                else
                {
                    outf  << " " << (*this)(i,j) << " ";
                }
        }

    }
    outf << "\n";
}

Matrix<T> get_col(size_t j)
{
    Matrix<T> res(row, 1);
    res.Zero();
    if(j >= row)
        return res;
    for(size_t i = 0; i < row; i++)
    {
        res(i,0) = this->operator ()(i,j);
    }
    return res;
}

bool is_razl()
{
```

```
        for(size_t i = 0; i < row; i++)
        {
            for(size_t j = 0; j < col; j++)
            {
                bool visited[row];
                if(!dfs(i,j,row, visited))
                {
                    return false;
                }
            }
        }
        return true;
    }


private:
    bool dfs(size_t u, size_t t, size_t n, bool* visited)
    {
        if(row != col || n != row)
            return false;
        visited[u] = true;
        for(size_t v = 0; v < n; v++)
        {
            if(!visited[v])
                if(dfs(v,t,n,visited))
                    return true;
        }
        return false;
    }
};
```

Приложение В **Программная реализация решения рассмотренных выше задач управления проектами**

Данная программа также реализована на языке C++. Программа написана в объектно-ориентированном стиле. Классы, использующиеся для реализации задач сетевого планирования наследуются от базового класса `Task`. Сами задачи формулируются в виде специально сгенерированного текстового файла. Результаты также выводятся в текстовый файл.

В.1 Абстрактный класс описания задач

Абстрактный класс `Task<T>` предназначен для обеспечения считывания и записи задач. В качестве шаблонного элемента используется наследник класса `Task<T>`, описанного в предыдущем разделе. В классе реализованы следующие открытые методы:

- `Task(const std::string& _order, const std::string& _solution)` – конструктор класса. На вход подаются строка адреса входного текстового файла `_order` и строка класса выходного тестового файла `_solution`, в который записывается решение;
- Функция `solve()` осуществляет решение задачи и запись решения в выходной файл

В.1.1 Формат ввода-вывода задач

- Любая задача начинается с ключевого слова `task`;
- Далее записываются матрицы и векторы, составляющие условия задачи. Начало задания матрицы выделяется ключевым словом `rowcol`. Отдельно на каждой строке выводятся число строк и столбцов матрицы. Затем задаются элементы матрицы. Вывод каждой матрицы должен заканчиваться словом `_rowcol`. Идемпотентный ноль задается символом z.

B.1.2 Листинг реализации класса

```
template<class T>
class Task
{
public:
    Task(const std::string& _order, const std::string& _solution)
    {
        order = _order;
        solution = _solution;
    }
    virtual void solve() = 0;
    double get_zero()
    {
        return alg.get_zero();
    }

    double get_unit()
    {
        return alg.get_unit();
    }

protected:
    virtual void read() = 0;
    std::vector<string> read_start(size_t& position)
    {
        std::vector<string> buf_tmp = Task<T>::read_task_file();
        for (int k = 0; k < buf_tmp.size(); k++)
        {
            if (buf_tmp[k] == "z")
                buf_tmp[k] = std::to_string(get_zero());
        }

        // разберём буфер и подготовим данные для класса задач

        for (position = 0; position < buf_tmp.size(); position++)
            if (buf_tmp[position] == "task") break;

        // проверим нашли ли слово "task"
        if (position == buf_tmp.size())
        {
            cout << "Invalid task file format ";
            exit(4);
        }
        return buf_tmp;
    }
    virtual void write() = 0;
    std::vector<string> read_task_file()
    {
        std::vector<string> buf;
        string buf_string;
        std::ifstream tskfile;  //поток ввода задачи
        int i=0;
        tskfile.open(order);
```

```
        if (!tskfile.is_open())
        {
            cout << "File does not exist ";
            exit(3);
        }
//  Файл задачи существует. Займёмся его разбором
        while (!tskfile.eof())
        {
            tskfile >> buf_string;      // считываем строку из файла
            buf.push_back(buf_string); // записываем её в буфер
        }
        tskfile.close();
        return buf;
    }
    Matrix<T> read_next_matrix_from_position(
    const std::vector<string>& buf_tmp, size_t& position)
    {
        while
        (
            buf_tmp[position] != "rowcol"
            && position < buf_tmp.size()
        )
            position++;
        if (buf_tmp[position] == "rowcol")
        {
            position++;
            int row, col;
            row = std::stoi(buf_tmp[position]);
            position++;
            col = std::stoi(buf_tmp[position]);
            Matrix<T> res(row, col);
            position++;
            while (buf_tmp[position] != "_rowcol")
            {
                for(size_t k = 0; k < row; k++)
                    for(size_t l = 0; l < col; l++)
                    {
                        res(k,l) = std::stod(buf_tmp[position]);
                        position++;
                        if(buf_tmp[position] == "_rowcol")
                            return res;
                    }
            }
            return res;
        }
        else
        {
            Matrix<T> res(0, 0);
            return res;
        }
    }
protected:
    std::string order;
    std::string solution;
```

```
    T alg;
};
```

B.2 Класс минимизации продолжительности проекта

Класс `Min_duration_time<T>`, который используется для решения задачи (3.36) является наследником класса `Task<T>`. В классе присутствуют следующие открытые методы:

- `Min_duration_time(const std::string& _order, const std::string& _solution)` – конструктор класса. На вход подаются строка адреса входного текстового файла `_order` и строка класса выходного тестового файла `_solution`, в который записывается решение;
- Функция `solve()` осуществляет решение задачи и запись решения в выходной файл

B.2.1 Формат ввода-вывода

В начале файла записывается ключевое слово `task`. Затем, в соответствии с описанными выше правилами записи матриц задаются матрицы $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ и векторы $\boldsymbol{g}$, $\boldsymbol{f}$ и $\boldsymbol{h}$. Приведем пример входного файла для рассмотренной в примере выше задачи сетевого планирования.

```
task
rowcol
3
3
0 -1 1
0 -1 2
-2 -3 0
_rowcol
rowcol
3
3
4 2 3
3 1 2
2 1 3
_rowcol
rowcol
3
3
-4 -5 -6
z -4 -7
-5 z -4
```

```
_rowcol
rowcol
3
1
0
0
0
_rowcol
rowcol
3
1
4
3
2
_rowcol
rowcol
3
1
6
6
6
_rowcol
```

В выходном файле сначала выводится минимум задачи (3.36), который равен θ, затем приводится диапазон значений для вектора $\boldsymbol{u}$. В конце приведены векторы $\boldsymbol{x}$ и $\boldsymbol{y}$. Единичные векторы обозначаются как `id_vec`. Приведем пример выходного файла:

```
theta = 5
 0
 0
 3
<= u <=
 2
 3
 1
x = G * u
G =
 0  0 -2
 1  0  2
 1  1  0
y = C * G * u
 C * G =
 4  3 5
 3  2 4
 2  1 3
```

B.2.2 Листинг реализации класса

```
template<class T>
class Min_duration_time: protected Task<T>
{
public:
    Min_duration_time
```

```
(
    const std::string& _order
    , const std::string& _solution
):Task<T>(_order, _solution){}

void read()
{
    size_t position;
    auto buf_tmp = Task<T>::read_start(position);
    B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    C=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    D=Task<T>::read_next_matrix_from_position(buf_tmp,position);

    g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    h=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    f=Task<T>::read_next_matrix_from_position(buf_tmp,position);
}
void write()
{
    std::ofstream solvefile;
    solvefile.open(this->solution);
    solvefile << "theta = " << _theta << std::endl;
    u_left.to_file(solvefile);
    solvefile << "<= u <= " << std::endl;
    u_right.to_file(solvefile);
    solvefile
    << "x = G * u"<< std::endl;
    << "G = "<< std::endl;
    G.to_file(solvefile);
            solvefile
    << "y = C * G * u" << std::endl;
    << "CG = "<< std::endl;
    CG.to_file(solvefile);
    solvefile.close();
}
void solve()
{
    read();
    Matrix<T> DC(B.get_rows(),B.get_cols());
    DC = D * C;
    R = B + DC;
    Matrix<T> sT(B.get_rows(),1);
    sT = f.pseudo_inverse() * C + h.pseudo_inverse();
    Matrix<T> _sT_R_ast_g;
    _sT_R_ast_g = sT * R.ast() * g;
    if
    (
        this->alg.order
        (
            R.ast().Tr()
            , this->get_unit())
        )
        &&
        this->alg.order
```

```
    (
        _sT_R_ast_g(0,0)
        , this->get_unit())
    )
)
{

    _theta = this->alg.oplus
             (
                 (C * R).norm()
                 , this->alg.otimes
                       (
                           (sT * R).norm()
                           , (C * g).norm()
                       )
             );
    _theta = this->alg.oplus
             (
                 _theta
                 , this->alg.otimes
                       (
                           sT.norm()
                           , (C * g).norm()
                       )
             );

     _theta = this->alg.oplus
             (
                 _theta
                 , this->alg.otimes
                       (
                           sT.norm()
                           , (C * R * g).norm()
                       )
             );

    _theta = this->alg.oplus
             (
                 _theta
                 , this->alg.otimes
                       (
                           (sT * R).norm()
                           , (C * R * g).norm()
                       )
             );
    Matrix<T> id_vec(B.get_rows(),1);
    id_vec.Identity_vec();
    G
    =
    R.ast
    +
    _theta *
    (
        (id_vec * id_vec.transpose()) * C
        + R * (id_vec * id_vec.transpose()) * C
```

```
                + (id_vec * id_vec.transpose()) * C * R
            );
            CG = C * G;
            u_left = g;
            u_right =
             (sT * G).pseudo_inverse();
            write();
        }
    }
 protected:
    Matrix<T> B;
    Matrix<T> C;
    Matrix<T> D;
    Matrix<T> R;

    Matrix<T> g;
    Matrix<T> f;
    Matrix<T> h;

    Matrix<T> u;
    Matrix<T> u_left, u_right;
    Matrix<T> G;
    Matrix<T> CG;

    double _theta;
};
```

B.3 Класс минимизации максимального отклонения

В классе `Min_max_deviation<T>` реализован алгоритм решения задачи (4.14). Класс является наследником класса `Task<T>`. Опишем его открытые методы:

- `Min_max_deviation(const std::string& _order, const std::string& _solution)` – конструктор класса. На вход подаются строка адреса входного текстового файла `_order` и строка класса выходного тестового файла `_solution`, в который записывается решение;
- Функция `solve()` осуществляет решение задачи и запись решения в выходной файл

B.3.1 Формат ввода-вывода

В начале файла записывается ключевое слово `task`. Далее, в соответствии с введенными ранее правилами записи матриц задаются матрицы $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ и векторы $\boldsymbol{g}$, $\boldsymbol{f}$, $\boldsymbol{h}$ и $\boldsymbol{p}$. Приведем пример входного файла для представленной в примере выше задачи сетевого планирования.

```
task
rowcol
3
3
0 -1 1
0 -1 2
-2 -3 0
_rowcol
rowcol
3
3
4 2 3
3 1 2
2 1 3
_rowcol
rowcol
3
3
-4 -5 -6
z -4 -7
-5 z -4
_rowcol
rowcol
3
1
0
0
0
_rowcol
rowcol
3
1
4
3
2
_rowcol
rowcol
3
1
6
6
6
_rowcol
rowcol
3
1
2
2
1
_rowcol
```

В выходном файле сперва выводится минимум задачи (4.14), равный θ, затем приводится диапазон значений для вектора $\boldsymbol{u}$. В конце приведены векторы $\boldsymbol{x}$ и $\boldsymbol{y}$. Единичные векторы обозначаются как `id_vec`. Приведем пример выходного файла:

```
theta = 1
 1
 1
 0
<= u <=
 2
 3
 1
x = G * u
G =
 0  -1 1
 1  0  2
 -1  -2  0
y = C * G * u
 C * G =
 4  2 3
 3  1 2
 2  1 3
```

B.3.2 Листинг реализации класса

```
template<class T>
class Min_max_deviation: protected Task<T>
{
public:
    Min_max_deviation
    (
        const std::string& _order
        , const std::string& _solution
    ):Task<T>(_order, _solution){}

    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        C=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        D=Task<T>::read_next_matrix_from_position(buf_tmp,position);

        g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        h=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        f=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        p=Task<T>::read_next_matrix_from_position(buf_tmp,position);

    }
    void write()
    {
        std::ofstream solvefile;
        solvefile.open(this->solution);
        solvefile << "theta = " << _theta << std::endl;
        u_left.to_file(solvefile);
        solvefile << "<= u <= " << std::endl;
        u_right.to_file(solvefile);
        solvefile
        << "x = R^(ast) * u"<< std::endl;
```

```
    << "R^(ast) = "<< std::endl;
    R_ast.to_file(solvefile);
            solvefile
    << "y = C * R^(ast) * u" << std::endl;
    << "CG = "<< std::endl;
    CR_ast.to_file(solvefile);
    solvefile.close();
}
void solve()
{
    read();
    Matrix<T> DC(B.get_rows(),B.get_cols());
    DC = D * C;
    R = B + DC;
    Matrix<T> sT(B.get_rows(),1);
    sT = f.pseudo_inverse() * C + h.pseudo_inverse();
    Matrix<T> _sT_R_ast_g;
    _sT_R_ast_g = sT * R.ast() * g;
    if
    (
        this->alg.order
        (
            R.ast().Tr()
            , this->get_unit())
        )
        &&
        this->alg.order
        (
            _sT_R_ast_g(0,0)
            , this->get_unit())
        )
    )
    {
       _theta
       =
       (
           (
               p.pseudo_inverse() * R.ast() * p)^(1/2)
           )
           +
           sT * R.ast() * p
           +
           p.pseudo_inverse() * R.ast() * g
       )(0,0);

       Matrix<T> id_vec(B.get_rows(),1);
       id_vec.Identity_vec();
       G
       =
       R.ast
       +
       _theta *
       (
           (id_vec * id_vec.transpose()) * C
           + R * (id_vec * id_vec.transpose()) * C
```

```
                + (id_vec * id_vec.transpose()) * C * R
            );
            R_ast = R.ast();
            CR_ast = C * R_ast;
            u_left = g + this->alg.pseudo_inverse(_theta) * p;
            u_right =
            (
                (
                    sT + this->alg.pseudo_inverse(_theta)
                        * p.pseudo_inverse()
                ) * R_ast
            ).pseudo_inverse();
            write();
        }
    }
 protected:
    Matrix<T> B;
    Matrix<T> C;
    Matrix<T> D;
    Matrix<T> R;

    Matrix<T> g;
    Matrix<T> f;
    Matrix<T> h;

    Matrix<T> u;
    Matrix<T> u_left, u_right;
    Matrix<T> R_ast;
    Matrix<T> CR_ast;

    double _theta;
};
```

B.4 Класс минимизации разброса времени завершения работ

Класс `Min_fin_time<T>`, использующийся для решения задачи (5.13), наследуется от класса `Task<T>`, описанного в предыдущем разделе. В классе реализованы следующие открытые методы:

- `Min_fin_time(const std::string&_order, const std::string&_solution)` – конструктор класса. На вход подаются строка адреса входного текстового файла `_order` и строка класса выходного тестового файла `_solution`, в который записывается решение;

- Функция `solve()` осуществляет решение задачи и запись решения в выходной файл

B.4.1 Формат ввода-вывода

В начале файла записывается ключевое слово `task`. Затем, в соответствии с описанными выше правилами записи матриц задаются матрицы $\boldsymbol{A}$, $\boldsymbol{B}$ и векторы $\boldsymbol{g}$, $\boldsymbol{h}$, $\boldsymbol{f}$. Приведем пример входного файла для рассмотренной в примере выше задачи сетевого планирования.

```
task
rowcol
3
3
4 0 z
2 3 1
1 1 3
_rowcol
rowcol
3
3
z -2 1
0 z 2
-1 z z
_rowcol
rowcol
3
1
1
0
0
_rowcol
rowcol
3
1
5
4
4
_rowcol
rowcol
3
1
8
8
10
_rowcol
```

В выходном файле сначала выводится решение задачи (5.13), равное Δ, затем записывается ограничение на α. В конце приведены векторы $\boldsymbol{x}$ и $\boldsymbol{y}$, на которых достигается решение. Приведем пример выходного файла:

```
Delta = 2
5 <= alpha <= 7
x = alpha *
-4
-3
```

```
-5
y = alpha *
 0
 0
-2
```

B.4.2 Листинг реализации класса

```
template<class T>
class Min_fin_time: protected Task<T>
{
public:
    Min_fin_time
    (
        const std::string& _order, const std::string& _solution
    ):Task<T>(_order, _solution){}

    void solve()
    {
        read();
        Matrix<T> id_vec(A.get_rows(),1);
        id_vec.Identity_vec();
        Matrix<T> alpha_left;
        Matrix<T> alpha_right;
        Matrix<T> delta;

        alpha_left = (id_vec.transpose() * A) * (B.ast()) * g;
        alpha_left.out();
        alpha_right =
          (((f.pseudo_inverse()* A) + h.pseudo_inverse())
          * B.ast()
          * (id_vec.transpose() * A * B.ast()).pseudo_inverse()
          ).pseudo_inverse();
        _alpha_left = alpha_left(0,0);
        _alpha_right = alpha_right(0,0);
        if(_alpha_left <= _alpha_right)
        {
            delta =
            (
             A * B.ast() *
              (
               id_vec.transpose() * A * B.ast()
              ).pseudo_inverse()
            ).pseudo_inverse() * id_vec;
            _delta = delta(0,0);

            x =
             B.ast() *
             (
              id_vec.transpose() * A * B.ast()
             ).pseudo_inverse();
            y = A * x;
            write();
        }
```

```
    }

protected:
    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
        A=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        h=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        f=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    }

    void write()
    {
        std::ofstream solvefile;
        solvefile.open(this->solution);
        solvefile << "Delta = " << _delta << std::endl;
        solvefile << _alpha_left
                  << "<= alpha <= "
                  << _alpha_right << std::endl;
        solvefile << "x = alpha * ";
        x.to_file(solvefile);
        solvefile << "y = alpha * ";
        y.to_file(solvefile);
        solvefile.close();
    }
 protected:
    Matrix<T> A;
    Matrix<T> B;
    Matrix<T> f;
    Matrix<T> g;
    Matrix<T> h;
    Matrix<T> x;
    Matrix<T> y;
    double _alpha_left, _alpha_right, _delta;
};
```

B.5 Класс максимизации разброса времени завершения работ

Класс `Max_fin_time<T>`, который используется для решения задачи(6.12), наследуется от класса `Task<T>`, описанного в предыдущем разделе. В классе реализованы следующие открытые методы:

- `Max_fin_time(const std::string& _order, const std::string& _solution)` – конструктор класса. На вход подаются строка адреса входного текстового файла `_order` и строка класса выходного тестового файла

`_solution`, в который записывается решение;

- Функция `solve()` осуществляет решение задачи и запись решения в выходной файл

B.5.1 Формат ввода-вывода

В начале файла записывается ключевое слово `task`. Затем, в соответствии с описанными выше правилами записи матриц задаются матрицы $\boldsymbol{A}$ и $\boldsymbol{B}$ и вектор $\boldsymbol{h}$. Приведем пример входного файла для рассмотренной в примере выше задачи сетевого планирования.

```
task
rowcol
3
3
4 1 1
2 2 0
0 1 3
_rowcol
rowcol
3
3
z -2 1
0 z 2
-1 z z
_rowcol
rowcol
3
1
5
4
4
_rowcol
```

В выходном файле сначала выводится решение задачи (6.12), которое равно Δ. Затем приводится матрица Клини $\boldsymbol{B}^*$ и матрица $\boldsymbol{D}$. Далее задается диапазон значений скаляра $\boldsymbol{alpha}$, значения индексов k, s и диапазон значений для вектора $\boldsymbol{u}$ В конце приведены векторы $\boldsymbol{x}$ и $\boldsymbol{y}$. Единичные векторы обозначаются словом `id_vec`. Выпишем пример выходного файла:

```
Delta = 2

B_ast =
 0 -2  1
 1  0  2
-1 -3  0

D =
```

```
 4  2  5
 3  2  4
 2  1  3
alpha <= 3, k = 3, s =  3
u_3= alpha *-3
u_1 <= alpha *-2
u_2 <= alpha *-1
x = B_ast * u
y = D * u
```

B.5.2 Листинг реализации класса

```
template<class T>
class Max_fin_time: protected Task<T>
{
public:
    Max_fin_time
    (
        const std::string& _order, const std::string& _solution
    ):Task<T>(_order, _solution){}
    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
        A=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        h=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    }
    void write()
    {
        std::ofstream solvefile;
        solvefile.open(this->solution);
        solvefile << "Delta = " << delta << std::endl;
        solvefile << "B_ast =" << std::endl;
        B_ast.to_file(solvefile);
        solvefile << "D =" << std::endl;
        D.to_file(solvefile);
        solvefile << "alpha <= "
                  << alpha
                  << ", k = "
                  << k + 1 <<", s =  " << s + 1 << std::endl;
        solvefile << "u_"
                  << k + 1
                  << "= alpha *"
                  << this->alg.power(D(s,k), -1) << std::endl;
        for(size_t i = 0; i < D.get_rows(); i++)
            if(i != k)
        {
                solvefile << "u_"
                          << i + 1
                          << " <= "
                          << "alpha *"
                          << this->alg.power(D(s,i), -1)
                          << std::endl;
```

```
    }
    solvefile << "x = " << "B_ast * u" << std::endl;
    solvefile << "y = " << "D * u" << std::endl;
    solvefile.close();
}
void solve()
{
    read();
    B_ast = B.ast();
    D = A * B_ast;
    if(A.is_regular() &&
            this->alg.order(B.Tr(), this->get_unit()) &&
            D.regular_cols())
    {
       Matrix<T> _delta;
       Matrix<T> id_vec(B.get_rows(),1);
       id_vec.Identity_vec();
       _delta = id_vec.transpose()
                * D * D.pseudo_inverse() * id_vec;
       delta = _delta(0,0);
       k = 0;
       s = 0;
       double val = this->alg.get_zero();
       for(size_t j = 0; j < D.get_cols(); j++)
       {
           Matrix<T> d_ast_j;
           d_ast_j = D.get_col(j);
           Matrix<T> id_d_di_id;
           id_d_di_id =
               id_vec.transpose()
               * d_ast_j * d_ast_j.pseudo_inverse() * id_vec;
           if(this->alg.order(val, id_d_di_id(0,0)))
           {
               k = j;
               val = id_d_di_id(0,0);
           }
       }
       val = this->alg.get_zero();
       for(size_t j = 0; j < D.get_cols(); j++)
       {
           if(this->alg.order(val, this->alg.power(D(j,k), -1)))
           {
               s = j;
               val = this->alg.power(D(j,k), -1);
           }
       }
       val = this->alg.get_zero();
       Matrix<T> d_0;
       d_0 = D.get_col(0);
       Matrix<T> buff;
       buff = (h.pseudo_inverse() * d_0).pseudo_inverse();
       buff = buff * D(s,0);
       alpha = buff(0,0);
```

```
            for(size_t j = 1; j < D.get_cols(); j++)
            {
                Matrix<T> d_j;
                d_j = D.get_col(j);
                buff = (h.pseudo_inverse() * d_j).pseudo_inverse();
                buff = buff * D(s,j);
                if(this->alg.order(buff(0,0), alpha))
                {
                    alpha = buff(0,0);
                }
            }
            write();
        }
    }
 protected:
    Matrix<T> A;
    Matrix<T> B;
    Matrix<T> h;
    Matrix<T> x;
    Matrix<T> y;
    double delta;
    double alpha;
    size_t k,s;
    Matrix<T> B_ast;
    Matrix<T> D;
};
```

B.6 Класс минимизации разброса времени начала работ

Класс `Min_start_time<T>`, который используется для решения задачи (7.8), наследуется от класса `Task<T>`, описанного в предыдущем разделе. В классе реализованы следующие открытые методы:

- `Min_start_time(const std::string&_order, const std::string&_solution)` – конструктор класса. На вход подаются строка адреса входного текстового файла `_order` и строка класса выходного тестового файла `_solution`, в который записывается решение;
- Функция `solve()` осуществляет решение задачи и запись решения в выходной файл

B.6.1 Формат ввода-вывода

В начале файла записывается ключевое слово `task`. Затем, в соответствии с описанными выше правилами записи матриц задаются матрица $\boldsymbol{B}$ и векторы

$\boldsymbol{g}$ и $\boldsymbol{h}$. Приведем пример входного файла для рассмотренной в примере выше задачи сетевого планирования.

```
task
rowcol
3
3
z 0 -2
-1 z -2
1 1 z
_rowcol
rowcol
3
1
0
0
3
_rowcol
rowcol
3
1
2
1
5
_rowcol
```

В выходном файле сначала выводится минимум задачи (7.8), который равен θ, затем приводится диапазон значений для вектора $\boldsymbol{u}$. В конце приведен вектор $\boldsymbol{x}$. Единичные векторы обозначаются как `id_vec`. Приведем пример выходного файла:

```
theta = 2
 0
 0
 3
<= u <=
 2
 1
 3
x = (theta^{-1} sum(B_i * id_vec * id_vec^{T} B_j) oplus B^{*} * u
 0  0 -2
-1  0 -2
 1  1  0
```

B.6.2 Листинг реализации класса

```
template<class T>
class Min_start_time: protected Task<T>
{
public:
    Min_start_time
    (
        const std::string& _order, const std::string& _solution
```

```
):Task<T>(_order, _solution){}
void read()
{
    size_t position;
    auto buf_tmp = Task<T>::read_start(position);
    B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    l=Task<T>::read_next_matrix_from_position(buf_tmp,position);
}
void write()
{
    std::ofstream solvefile;
    solvefile.open(this->solution);
    solvefile << "theta = " << _theta << std::endl;
    u_left.to_file(solvefile);
    solvefile << "<= u <= " << std::endl;
    u_right.to_file(solvefile);
    solvefile <<
    "x = (theta^{-1} sum(B_i * id_vec * id_vec^{T} B_j)
     oplus B^{*} * u";
    theta_min_one_B_B_Bi_ast.to_file(solvefile);
    solvefile.close();
}
void solve()
{
    read();
    Matrix<T> id_vec(B.get_rows(),1);
    id_vec.Identity_vec();
    Matrix<T> _l_B_g_ast;
    _l_B_g_ast = l.pseudo_inverse() * B.ast() * g;
    if(
    this->alg.order(
    this->alg.oplus(B.Tr(), _l_B_g_ast(0,0)), this->get_unit()))
    {
       size_t n = B.get_rows();
       auto powers = get_all_pairs(n-2);
       double elem = this->get_zero();
       for(size_t i = 0; i < powers.size(); i++)
       {
          double lB =
           (l.pseudo_inverse() * (B^(powers[i][0]))).norm();
          double Bg = ((B^(powers[i][1])) * g).norm();
          elem=this->alg.oplus(elem, this->alg.otimes(lB, Bg));
       }
       _theta = this->alg.oplus(elem, B.ast().norm());
       for(size_t i = 0; i < powers.size(); i++)
       {
          theta_min_one_B_B_Bi_ast =
            (B^powers[i][0]) *
              (id_vec * id_vec.transpose()) * (B^powers[i][1]);
       }
       theta_min_one_B_B_Bi_ast =
          theta_min_one_B_B_Bi_ast * this->alg.power(_theta,-1);
```

```
            theta_min_one_B_B_Bi_ast =
               theta_min_one_B_B_Bi_ast + B.ast();

            u_left = g;
            u_right =
            (
                l.pseudo_inverse() * theta_min_one_B_B_Bi_ast
            ).pseudo_inverse();
            write();
        }
    }
 protected:
    Matrix<T> B;
    Matrix<T> g;
    Matrix<T> l;
    Matrix<T> u;
    Matrix<T> u_left, u_right;
    Matrix<T> theta_min_one_B_B_Bi_ast;
    double _theta;
};
```

B.7 Класс максимизации разброса времени начала работ

Класс `Max_start_time<T>`, который используется для решения задачи (8.15), наследуется от класса `Task<T>`, описанного в предыдущем разделе. В классе реализованы следующие открытые методы:

- `Max_start_time(const std::string&_order, const std::string& solution)` – конструктор класса. На вход подаются строка адреса входного текстового файла `_order` и строка класса выходного тестового файла `_solution`, в который записывается решение;
- Функция `solve()` осуществляет решение задачи и запись решения в выходной файл

B.7.1 Формат ввода-вывода

В начале файла записывается ключевое слово `task`. Затем, в соответствии с описанными выше правилами записи матриц задаются матрица $\boldsymbol{B}$ и вектор $\boldsymbol{g}$. Приведем пример входного файла для рассмотренной в примере выше задачи сетевого планирования.

```
task
rowcol
3
```

```
3
z -2 1
0 z 2
-1 z z
_rowcol
rowcol
3
1
2
0
0
_rowcol
```

В выходном файле сначала выводится решение задачи (8.15), равное Δ. Затем приводится матрица Клини $\boldsymbol{B}^*$. Далее задается диапазон значений для α, значения индексов k, s и диапазон значений для вектора $\boldsymbol{u}$ В конце приведен вектор $\boldsymbol{x}$. Единичные векторы обозначаются как `id_vec`. Выпишем пример выходного файла:

```
Delta = 3
B_ast =
 0 -2  1
 1  0  2
-1 -3  0
alpha >= 1, k = 2, s =  3
u_2= alpha *3
2 <= u_1 <= alpha *1
0 <= u_3 <= alpha *-0
x = B_ast * u
```

B.7.2 Листинг реализации класса

```
template<class T>
class Max_start_time: protected Task<T>
{
public:
    Max_start_time
    (
        const std::string& _order, const std::string& _solution
    ):Task<T>(_order, _solution){}
    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    }
    void write()
    {
        std::ofstream solvefile;
        solvefile.open(this->solution);
        solvefile << "Delta = " << delta << std::endl;
```

```
    solvefile << "B_ast =" << std::endl;
    B_ast.to_file(solvefile);
    solvefile << "alpha >= "
              << alpha_bound
              << ", k = "
              << k + 1
              <<", s =  "
              << s + 1
              << std::endl;
    solvefile << "u_"
              << k + 1
              << "= alpha *"
              << this->alg.power(B_ast(s,k), -1)
              << std::endl;
    for(size_t i = 0; i < B_ast.get_rows(); i++)
        if(i != k)
    {
            solvefile << g(i,0)
                      << " <= "
                      << "u_"
                      << i + 1
                      << " <= "
                      << "alpha *"
                      << this->alg.power(B_ast(s,i), -1)
                      << std::endl;
    }
    solvefile << "x = "
              << "B_ast * u"
              << std::endl;
    solvefile.close();
}
void solve()
{
    read();
    if(
       this->alg.order
        (B.Tr(), this->get_unit())
       &&
       !B.is_razl() && B.is_quadr()
      )
    {
       Matrix<T> _delta;
       Matrix<T> id_vec(B.get_rows(),1);
       id_vec.Identity_vec();
       B_ast = B.ast();
       _delta =
        id_vec.transpose()
          * B_ast * B_ast.pseudo_inverse() * id_vec;
       delta = _delta(0,0);
       k = 0;
       s = 0;
       double val = this->alg.get_zero();
       for(size_t j = 0; j < B_ast.get_cols(); j++)
       {
           Matrix<T> b_ast_j;
```

```
                b_ast_j = B_ast.get_col(j);
                Matrix<T> id_b_bi_id;
                id_b_bi_id
                 = id_vec.transpose()
                 * b_ast_j * b_ast_j.pseudo_inverse() * id_vec;
                if(this->alg.order(val, id_b_bi_id(0,0)))
                {
                    k = j;
                    val = id_b_bi_id(0,0);
                }
            }
            val = this->alg.get_zero();
            for(size_t j = 0; j < B_ast.get_cols(); j++)
            {
                if
                (
                    this->alg.order
                    (
                        val, this->alg.power(B_ast(j,k), -1)
                    )
                )
                {
                    s = j;
                    val = this->alg.power(B_ast(j,k), -1);
                }
            }
            val = this->alg.get_zero();

            for(size_t j = 0; j < B_ast.get_cols(); j++)
            {
                if
                (
                    this->alg.order
                    (
                        val, this->alg.otimes(g(j,0),B_ast(s,j))
                    )
                )
                {
                    val = this->alg.otimes(g(j,0),B_ast(s,j));
                }
            }
            alpha_bound = val;
            write();
        }
    }
 protected:
    Matrix<T> B;
    Matrix<T> g;
    double delta;
    size_t k,s;
    Matrix<T> B_ast;
    double alpha_bound;

};
```

Saint-Petersburg University

On the rights of the manuscript

Gubanov Sergei Alexandrovich

Solution of minimax problems of optimal project sheduling using idempotent algebra methods

Scientific specialty 1.2.2.
Mathematical modeling, numerical methods and software packages

Thesis is submitted for the degree of
Candidate of Physical and Mathematical Sciences
Translation from Russian

Scientific Supervisor:
Doctor of Physical and Mathematical Sciences
Krivulin Nikolai Kimovich

Saint Petersburg
2023

Contents

Intro

The relevance of the dissertation topic

The dissertation paper is devoted to the development of models, algorithms and software tools for solving project management problems using tropical optimization methods. The paper investigates a number of topical problems of optimal scheduling of projects, which are formulated as new problems of tropical optimization, and their direct analytical solution is proposed.

In the process of development of enterprises, their activities are continuously becoming more complex. In addition to simple repetitive actions, there is a need to implement non-repeated interrelated activities, called complex activities or projects. To manage complex non-repetitive actions, including the temporary scheduling of such actions, apply various approaches that are combined under the term «project management». Examples of such approaches include network planning methods, which rely on project analysis using graph and network models, such as the critical path method (CPM - Critical Path Method) [1] and the method of evaluation and revision of plans (PERT - Program Evaluation and Review Technique) [2]), methods of linear and mixed integer linear programming [3–5]

Although in the general case, project management problems are quite complex and NP-hard, time planning problems (due to the lack of imposed cost and resource constraints) can usually be formulated and solved as linear programming problems. As a consequence, they can be solved numerically in polynomial time using appropriate computational procedures such as the Karmarkar and Floyd-Warshall algorithms.

The use of tropical mathematics methods is one of the effective ways to solve practical optimization problems that arise in such problems.

Tropical mathematics studies semifields and semirings with idempotent addition [6–11], and various related computational tasks. Often functions that are non-linear and non-smooth in ordinary mathematics become linear after they are translated into a tropical form. Such problems can be solved by computing tropical eigenvalues and matrix vectors and solving tropical equations and inequalities. Moreover, many linear algebra algorithms (for example, the Jacobi method and the

Gauss-Seidel algorithm) have idempotent counterparts, allowing to create efficient computational algorithms. As a result, new possibilities appear for the analysis of such problems, which often leads to a simplification of both the procedures for their numerical solution and the interpretation of the results obtained.

In addition, with the help of linear equations formulated in terms of idempotent algebra, the many dynamical systems behavior can be described, which are used, for example, in solving planning and control problems, making it possible to propose new methods for simulating such systems.

Methods and algorithms of tropical mathematics are successfully used to solve many practical problems, including placement problems, decision making and network planning tasks. To solve such problems, one usually uses finite-step computational methods, including methods of linear and mixed integer linear programming, methods of discrete and combinatorial optimization, in which iterative procedures are used to numerically obtain one of the solutions, if the solution exists, or to verify that it does not exist.

In contrast to the indicated numerical methods, tropical optimization methods make it possible to find the entire set of solutions analytically in explicit form in a compact matrix-vector form. The analytical form of the results obtained is more informative and convenient. It provides more opportunities for formal analysis of solutions by mathematical methods, can tell more about the fundamental structure of the solution set, and better explain the influence of input parameters on the solution. The analytical representation of the set of solutions in matrix-vector form can serve as a basis for refining solutions by finding among them solutions that satisfy additional constraints and optimality criteria.

In addition, the analytical solution is usually simpler for algorithmic and software implementation. The solution procedure consists of a finite number of matrix-vector operations and has low polynomial computational complexity. Such a procedure enables efficient software implementation for execution on both serial and parallel computing systems. Finally, the solution obtained in an analytical form is more accurate, since it avoids the accumulation of rounding errors that usually occur in numerical algorithms.

Tropical optimization methods have been successfully used to solve problems of

time planning for projects with various types of constraints and optimality criteria. Time constraints include precedence relationships for project start and finish times («start-finish», «start-start», «finish-start», and «finish-finish» constraints), as well as upper and lower bounds for start times («release time», «release deadline») and completion («completion deadline») of each job. The optimality criteria are the temporal characteristics of the project that need to be minimized (maximized), such as the maximum duration of the project, the maximum deviation from the due dates for completing the project, the maximum spread in the start time of the project, and others. For a number of formulations of temporal planning problems, which are determined by the choice of the optimality criterion and the set of constraints, analytical solutions obtained using tropical optimization methods are known. Such solutions, however, are not known for all optimality criteria that may be of interest for practical planning problems. In addition, existing solutions usually take into account only a part of possible restrictions, while for practical problems it is usually essential to take into account all known types of restrictions. Therefore, the formulation and solution of new previously unexplored problems of temporary project planning based on the application and further development of tropical optimization models and methods, including the development of computational procedures, algorithms and software, seems to be very relevant.

The degree of development of the topic in the literature

In many optimization problems, such as placement problems [8, 12–14] and network planning [15], the objective function and constraints can be described using the maximum and minimum operations, and as well as arithmetic operations. Finding solutions to such problems is usually associated with some difficulties, which can be associated, in particular, with the non-linearity and non-smoothness of the objective function and restrictions. An effective way to simplify such tasks is to reformulation in the language of tropical mathematics and subsequent use of its results [16–18] to solve the problem. Tropical optimization problems are usually understood as optimization problems formulated in terms of tropical mathematics. Such problems are an important part of modern research on tropical mathematics. The development of this direction was initiated by the works [9, 15, 19], recent research results can be found, for example, in [13, 20–28].

Separately, it is necessary to note the class of optimization problems that can be formulated in terms of tropical mathematics and consist in minimizing non-linear functionals, with restrictions in the form of linear vector inequalities [29]. The solution of such problems is usually based on the extremal property of the spectral matrix radius and is related to its calculation [20, 23, 25, 30].

Often, when representing temporal planning problems in the language of tropical mathematics, new tropical optimization problems arise that require a special approach to their solution [8, 22–24, 29–37].

This work is devoted to solving a number of problems of time planning and arising from their presentation on language of tropical mathematics the new problems of tropical optimization. The studied time planning problems with a smaller set of constraints have already been solved in the works [24, 33, 35, 37]. In the presented dissertation, it is proposed to extend these results to problems with a large set of constraints.

The object of research is the development of models and methods of tropical optimization for solving applied problems. **The subject of the study** is the construction of analytical solutions, computational procedures and software tools based on tropical optimization methods for the problems of time scheduling of project implementation dates.

Purpose of the work

The purpose of the dissertation work is to expand the existing apparatus for solving the problems of compiling optimal work schedules using tropical optimization methods to new planning problems that have not been studied before. Problems with a more general set of constraints and optimality criteria of the plan are considered, for which it is required to find analytical solutions, construct computational procedures, and develop software tools.

Main tasks

To achieve this goal, it is necessary to formulate and solve the following tasks:

1. To introduce a number of new tasks for developing optimal jobs schedules project in the form of minimax optimization problems with different types of constraints. The following are used as optimality criteria:

- project duration;
- deviation from the specified due dates for the execution of project jobs;
- scatter of project completion time;
- scatter in the start time of the project.

Constraints are set using:

- minimum allowable time intervals between the start of job;
- minimum time intervals between the start and completion of job;
- minimum time intervals between completion and start of job;
- earliest and latest allowable start times;
- the latest allowable completion time.

2. Formulate these new planning problems in the language of tropical mathematics (in the form of tropical optimization problems).

3. Develop methods for finding a complete solution to optimization problems with linear restrictions on the set of admissible values in an explicit and simple analytical form, based on the application of the apparatus of tropical mathematics and its presentation in the form of theorems.

4. Develop computational procedures and software tools for solving tropical optimization problems.

5. Develop applications of the results obtained for solving urgent problems of planning the work of medical institutions.

Compliance of the dissertation with the passport of the specialty

The content of the dissertation research corresponds to the following paragraphs of the passport specialty 1.2.2 - « Mathematical modeling, numerical methods and software packages»: development of new mathematical methods for modeling objects and phenomena (item 1); development of qualitative and approximate analytical methods research of mathematical models (item 2); development, justification

and testing of efficient computational methods using modern computer technologies (point 3); implementation of efficient numerical methods and algorithms in the form complexes of problem-oriented programs for computational experiment (item 4);

Scientific novelty

Scientific novelty is as follows:

- A number of new problems of planning the timing of projects with different optimality criteria and types of constraints are considered.
- For the considered planning problems, their representation is constructed in the form of tropical optimization problems with different objective functions and constraints.
- For the problems under study, new general analytical solutions are found, their representation in a compact matrix-vector form is obtained.
- Based on the results obtained, new efficient computational procedures and developed software tools for finding solutions to these problems.
- The results obtained were applied for the first time to solve urgent problems of planning medical events.

Research methods

The work uses the tools of linear algebra, general number theory, mathematical modeling. In addition, they are used methods of optimization, computer modeling, construction of mathematical models of complex systems and idempotent mathematics. Programming was carried out in the high-level language C++ with extensive use of object-oriented programming approaches, which were naturally used to describe classes that implement matrix-vector operations of idempotent algebra and solve tropical optimization problems.

Degree of confidence Reliability of the theoretical results presented in the paper provided by their rigorous mathematical proof. The paper gives complete proofs for the theorems proven by the applicant. References are provided for other used results. for evidence.

Theoretical and practical value of the work

The results of the dissertation work are of theoretical value and consist in obtaining a complete solution for a number of tropical optimization problems. The solutions have a matrix-vector form, and, therefore, the calculations can be naturally parallelized. In addition, solutions are presented in a convenient form in a simple analytical form, which greatly simplifies further analytical studies. using mathematical methods. The obtained theoretical results are used to find solutions to actual practical problems of scheduling the timing of projects and it is proposed to use them for improving the efficiency of medical institutions, and for the elimination of emergencies, which is especially important in our time. For the presented tasks of drawing up optimal calendar schedules for the implementation of project jobs, software tools have also been developed that implement the proposed solutions.

The results of the dissertation work were obtained with the support of grants Russian Humanitarian Science Foundation RGNF No. 13-02-00338a – «Models and methods of tropical mathematics in applied problems of economics and management» and RGNF №16-02-00059 – «Development of models and methods of tropical mathematics in applied problems of economics and management», as well as grants from the Russian Foundation for Fundamental Research RFBR No. 18-010-00723A – «Development models and methods of tropical mathematics for applied problems of economics and management» and RFBR №20-010-00145 – «Models and methods of tropical optimization in applied problems economics and management».

Personal contribution

The author took an active part in the mathematical proof, software implementation and presentation of the results of the work on seminars and conferences. The published results include the following: personally received by the author:

- Direct complete solutions formulated as theorems for the following constrained optimal scheduling problems: project duration minimization problem, the task of minimizing the deviation from the given due dates for the execution of project jobs, the task of minimizing and maximizing the scatter in the completion time of the project, the tasks of minimizing and maximizing the spread of the start time of the project jobs.

- Solutions of auxiliary problems of tropical optimization formulated as theo-

rems and their strict mathematical proofs.

- Computational procedures for solving and analysis of their computational complexity.

- A class library written in the C++ high-level language that implements the obtained solution search algorithms.

- Development of applications of the obtained results for solving urgent problems of planning the work of medical institutions.

Brief description of the work

The dissertation work consists of an introduction, eight chapters, a conclusion and an appendix. The full volume of the dissertation is 130 pages of typewritten text. The list of references contains 110 titles. The introduction substantiates the relevance of the topic of the dissertation, it seems a review of the relevant literature, the goals and objectives of the work are set, and their scientific value is argued.

The chapter 1 contains the main definitions related to the tasks of time planning of project execution dates. In 1.1 the concept of project management is defined, a brief overview of the development of this topic is given, and a presentation of project management problems in the form of a problem of maximizing or minimizing a certain optimality criterion under constraints in the form of equalities and inequalities is proposed. Further, in the section 1.2, the conditions for the order of the project jobs are formally set. Finally, in section 1.3 the optimality criteria used in the problems under study are given.

Then, in chapter 2 the results of idempotent algebra and tropical optimization are presented, which will be required to solve the studied problems of time planning of calendar dates for the execution of jobs. Section 2.1 presents an overview of the main definitions of tropical mathematics, necessary for solving project management problems using tropical optimization methods. Further, in the section 2.2 the concept of a tropical optimization problem is defined, examples of various problems are given and the solution of optimization problems is presented, which will be further used to solve tropical optimization problems with more stringent constraints.

Chapter 3 presents the problem of minimizing the duration of the project with restrictions on the time of its jobs. In section 3.1, a tropical optimization problem is formulated and solved, which will be used to solve the optimal planning problem considered in this chapter. Then, in the section 3.2, the task of minimizing the duration of the project execution with constraints is formally set and presented its solution. Finally, in section 3.3 the result obtained is used for the task of organizing the fastest vaccination.

Then, in chapter 4, a solution to the problem of minimizing the maximum deviation in the timing of jobs is proposed project from the given deadlines with known restrictions on the time of job. In the section 4.1 the solution of the problem of tropical optimization is proposed, to which the network planning problem under study is subsequently reduced in section 4.2. Finally, in 4.3 the found result is used to solve the problem of organizing the evacuation of the population in case of an emergency.

Chapter 5 deals with the problem of minimizing the scatter of project completion time under given constraints. In 5.1 a tropical optimization is formulated and solved, to which in 5.2 the problem of optimal control studied in the chapter is reduced. Further, in section 5.3, an example of a practical task is given, which can be solved using the presented results and consists in optimizing the conduct of medical examinations.

Then, in chapter 6, the problem of maximizing the spread of project completion time with given constraints is considered. In section 6.1, a solution to the problem of tropical optimization is proposed, to which further, in section 6.2, the network planning problem under study is reduced. In the section 6.3, the result obtained is used to optimize the work of a medical center that conducts medical procedures associated with harmful effects on the body.

Chapter 7 deals with the solution of the problem of minimizing the scatter of the start time of the project with restrictions on time and the sequence of their execution using the methods of idempotent algebra. Further, in section 7.1 the problem is formally set network planning and reduces to the well-known problem of tropical optimization. Then, in 7.2 the result is used to find solving the problem of organizing assistance to the wounded in case of an emergency.

Finally, in chapter 8, we study the problem of maximizing the scatter of jobs start times project with time constraints. In section 8.1 the problem of tropical optimization is formulated and solved, to which, in section 8.2, the optimal control problem under consideration is reduced. In the section 8.3, it is proposed to use the obtained result to optimize the work of the polyclinic during the quarantine period.

Provisions for defense

- A mathematical model of the problem of minimizing the duration of the project has been developed, tasks of minimizing the maximum deviation from the due dates, tasks of minimizing and maximizing the maximum spread in the completion time of the project, as well as the tasks of minimizing and maximizing the maximum spread of the start time of the project jobs with restrictions on the time of their execution.

- A number of tropical optimization problems were formulated and completely solved with different objective functions and linear constraints.

- Complete analytical solutions of planning problems formulated as tropical optimization problems are obtained. The results are presented in the form of theorems.

- Computational procedures for solving tropical optimization problems are developed and their computational complexity is studied.

- Software tools for solving problems of tropical optimization have been developed.

- Applications of the obtained results are developed for solving actual tasks of planning the work of medical institutions.

Approbation of results

The results of the dissertation were reported on 6th Scientific and Practical Internet Conference «Interdisciplinary research in the field mathematical modeling and informatics» (Tolyatti, 2015); 7th All-Russian Scientific Conference on Informatics Problems SPISOK-2017 (St. Petersburg, 2017); All-Russian Scientific Conference on

Problems of Informatics SPISOK-2019 (St. Petersburg, 2019); All-Russian Scientific Conference on Problems of Informatics SPISOK-2022 (St. Petersburg, 2022); International conference Polinomial Computer Algebra (PCA - 2022); 23rd All-Russian Conference of Young Scientists on mathematical modeling and information technologies (Novosibirsk, 2022); 2nd International Forum "Mathematical Methods and Models in high-tech production" (St. Petersburg, 2022); 15th International Scientific and Technical Conference "Modern mechanical engineering problems" (Tomsk, 2022); seminar on stochastic programming at the Department of System Programming, as well as at seminars of the Department of Statistical Modeling of the Faculty of Mathematics and Mechanics of St. Petersburg University in the process of studying the applicant in graduate school.

Publications The main results of the dissertation work are presented in printed papers [38–41], published in peer-reviewed scientific publications recommended by the Higher Attestation Commission under the Russian Ministry of Education and Science. In total, the author published 12 publications [38–49], devoted to solving various problems of tropical optimization and solving problems of network planning and control, by reducing them to known optimization problems and applying methods of tropical optimization. All solutions are obtained analytically in a convenient matrix-vector form.

Chapter 1

Project scheduling problems

This chapter provides the main definitions related to the tasks of time scheduling of the timing of the project. Section 1.1 defines the concept of project management, provides a brief overview of the development of this topic and proposes a representation of project management problems in the form of a problem of maximizing or minimizing some optimality criterion under constraints in the form of equalities and inequalities. Further, in the section 1.2, the conditions for the order of the project jobs are formally set. Finally, in section 1.3 are given the optimality criteria used in the problems under study.

1.1 Project management problems

Project management is the coordination of activities that are carried out to achieve the objectives of the project with a reasonable distribution of available resources. One of the most common project management tasks is network planning [50, 51].

The first methods for solving such problems were proposed in the late fifties. Critical Path Method (CPM) [1] and the method of evaluation and revision of plans (PERT – Program Evaluation and Review Technique) [2].

Similar problems can be represented as optimization problems, for the solution of which there exists various methods and algorithms. One of the promising ways to solve network planning problems is proposed in [21, 52, 53], method that consists in solving a problem using models and methods of tropical optimization.

Tropical (idempotent) mathematics is a branch of mathematics that studies semirings with idempotent addition [7, 22, 26, 54, 55]. In [15, 56, 57] it is proposed as a natural tool used to describe and solve real practical problems of network planning.

Typically, such problems are formulated as problems of minimization (maximization) of objective function over finite-dimensional semimodules over an idem-

potent semifield under various constraints expressed using linear equations and inequalities.

Next, we describe the optimization problems that arise when planning jobs schedules for various projects and are reduced to maximizing or minimizing some objective function under constraints in the form of equalities and inequalities. The objective function is determined by the optimality criterion of the plan, and the constraints are set by the conditions for the start and completion of jobs of the project. In these problems, the objective function and constraints are expressed using operations of maximum, addition and subtraction.

1.2 Constraints for the execution of jobs

Let us consider a project that consists of performing n jobs under given constraints on their execution time. We will assume that no job can be completed until some given time has passed after start of other jobs (restriction «start-finish»). There is also a minimum time interval between the start times of any two jobs («start-start» constraint). The earliest start time for each job is given by the constraint «release time», and the latest allowed start time using the constraint «release deadline». The latest possible completion time is determined by the «completion deadline» constraint. We assume that each job is completed immediately after the fulfillment of all specified restrictions on the time of its completion

Let us represent the restrictions introduced above in a more formal form in the form of linear equations and inequalities. To do this, for each job $i = 1, \dots, n$ we define the following notation:

x_i – start time of the job i;

y_i – completion time of the job i;

b_{ij} – minimum time interval between the start of $j = 1, \dots, n$ and the start of i;

c_{ij} – minimum time interval between the start of $j = 1, \dots, n$ and the end of i;

d_{ij} – minimum time interval between completion of $j = 1, \dots, n$ and start of i;

If the value of the interval b_{ij}, c_{ij} or d_{ij} is not set, then we consider it equal to $-\infty$;

g_i – earliest possible start time of i;

h_i – latest allowed start time;

f_i is the latest allowed completion time.

The «release time», «release deadline», and «completion deadline» constraints are denoted as g_i, h_i and f_i and set lower and upper bounds for x_i and y_i:

$$g_i \le x_i \le h_i, \qquad y_i \le f_i. \tag{1.1}$$

Constraints «start-start» for i operation can be defined for all $j = 1, \ldots, n$ in the form of inequalities

$$b_{ij} + x_j \le x_i. \tag{1.2}$$

The union of all inequalities in j gives the equivalent inequality

$$\max_{1 \le j \le n} (b_{ij} + x_j) \le x_i. \tag{1.3}$$

Constraints «start-finish» for each job i can be represented using the inequality

$$\max_{1 \le j \le n} (c_{ij} + x_j) \le y_i. \tag{1.4}$$

We will assume that the job is completed immediately after the «start-finish» constraints set for it are met, and, therefore, at least one of the inequalities is satisfied as an equality. Consequently, the inequality can be replaced by the equivalent equality

$$\max_{1 \le j \le n} (c_{ij} + x_j) = y_i. \tag{1.5}$$

For each job i, we represent all precedence relations of the form «finish-start» in the form of inequality

$$\max_{1 \le j \le n} (d_{ij} + y_j) \le x_i. \tag{1.6}$$

1.3 Criteria for the optimality of time planning

Often in the problems of optimal project sheduling there is a need to minimize overall duration of the project. For such problems, the following optimality criterion can be used, which is required to be minimized

$$\max_{1\le i\le n} y_i - \min_{1\le i\le n} x_i = \max_{1\le i\le n} y_i + \max_{1\le i\le n} (-x_i)\,. \tag{1.7}$$

In project management tasks, it may be necessary, if possible, to minimize the maximum temporary deviation from some given interval. Let p_i and q_i denote, respectively, the lower and upper limits of the intervals of directive start dates for each job $i = 1, \ldots, n$. We believe that in the problem it is required, if possible, to minimize the maximum overflow beyond the lower boundary of the interval $p_i - x_i$ and the maximum overflow $x_i - q_i$. Let us define the optimality criterion to be minimized in the following form

$$\max\left(\max_{1\le i\le n}(p_i - x_i), \max_{1\le i\le n}(x_i - q_i)\right). \tag{1.8}$$

When drawing up production plans, it often becomes necessary to complete the execution of all jobs at the same time. This approach is called "just-in-time" planning [50, 51]. As a criterion for the optimality of the plan, the maximum spread between the completion time of all jobs can be used

$$\max_{1\le i\le n} y_i - \min_{1\le i\le n} y_i = \max_{1\le i\le n} y_i + \max_{1\le i\le n} (-y_i)\,, \tag{1.9}$$

to be minimized. Note that if it becomes necessary to avoid the simultaneous completion of all jobs, the same criterion can be applied, but it will need to be maximized In optimal scheduling problems, it may be necessary to avoid avoiding the simultaneous start of jobs. For such tasks, you we suggest the following optimality criterion

$$\max_{1\le i\le n} x_i - \min_{1\le i\le n} x_i = \max_{1\le i\le n} x_i + \max_{1\le i\le n} (-x_i)\,, \tag{1.10}$$

to be maximized. If there is a need to ensure a minimum spread of time between the start of jobs, the same criterion can be used, but it will need to be minimized.

Network planning problems to be solved are various combinations of the presented optimality criteria and constraints. Similar problems can be applied in a

number of real practical problems, for example, when optimizing the work of a polyclinic or a medical laboratory.

Chapter 2

Idempotent algebra and tropical optimization

Let us present the main definitions and results of idempotent algebra and tropical optimization, which will be required to solve the studied problems of time scheduling of calendar dates for the execution of jobs. Section 2.1 provides an overview of the main definitions of tropical mathematics, necessary for solving project management problems using tropical optimization methods. Then, in the section 2.2, the concept of a tropical optimization problem is defined, examples of various problems are given, and the solution of optimization problems is presented, which will be further used to solve tropical optimization problems with more stringent constraints.

2.1 Elements of tropical mathematics

To formulate and solve the tropical optimization problems considered in this paper, we need algebraic definitions, notation, and preliminary results of idempotent algebra [7, 22, 26, 54, 55].

2.1.1 Introduction to tropical mathematics

The concept of a semiring, which is the central concept of tropical mathematics was first implicitly used in R. Dedekind's book [58] of 1894 and some other papers. In the papers [59, 60] it is pointed out that the notion of a semiring was explicitly introduced by G. Vandiver in 1934 in his paper [61]. Due to the large number of related theoretical and applied problems, semirings and their various applications have been a popular research topic for a long time. In the 1950-60s, the active development of tropical mathematics began. The papers of S. K. Kleene [62] in 1956 and R. A. Cunningheim-Green [63] in 1962, N. N. Vorobiev [64–66] in 1963, 1967 and 1970 and I.V. Romanovsky [67, 68] in 1962 and 1964.

One of the first Russian articles devoted to tropical mathematics is the paper of N.N. Vorobyov [65], which was preceded by various theoretical works (for example, on the theory of structures [69] and $\mathcal{K}$-spaces [70]) and numerous practical problems

of computational mathematics. As N. N. Vorobyov writes: «... the need to generalize a number of specific facts encountered in various applied mathematical theories. It is enough to refer to the articles by A. G. Lunts [71] and N. G. Povarov [72] on the theory of relay-contact circuits, as well as A. Shimbel [73], R. Bellman and W. Karush [74] et al. ([65], p. 42), explored questions of paths in a graph.» Similar computational problems arose not only in the design and production of contact circuits, but also in many other areas of the national economy. (see, for example, the paper of L. V. Kantorovich [75] of 1942). Vorobyov notes that "the ideas of extreme harmonic analysis... in the spirit of dynamic programming are analyzed by IV Romanovsky [67, 68]" ([65], p. 42). A. A. Korbut, in his papers [76, 77], conducts further research on the «extreme vector spaces» introduced in these papers. S. N. Samborsky in his paper [78] explores “the question of the existence nontrivial endomorphism spectrum over an idempotent semimodule”, and also describes the asymptotic behavior during iterations and the convergence of the «Neumann series» that appear when solving the equations $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{y} \oplus \boldsymbol{f}$. In its modern form, idempotent analysis was developed by a research team led by Academician V.P. Maslov [79–84] in the 1980s in Moscow [85]. For now there are two main directions in the development of tropical mathematics: the use of purely algebraic methods used, for example, in the works [7, 8, 10, 20, 26, 29, 55, 86–88], and a geometric approach based on the use of tropical geometry methods [27, 28, 89–92].

2.1.2 Idempotent semiefield

Let the set $\mathbb{X}$ be closed under the associative and commutative operations of addition $\oplus$ and multiplication $\otimes$. We assume that $\mathbb{X}$ contains their neutral elements zero $\mathbb{0}$ and unit $\mathbb{1}$. Addition satisfies the idempotency property (for any $x \in \mathbb{X}$ the equality $x \oplus x = x$ holds), and multiplication is distributive with respect to addition and reversible (for any nonzero x there is an element x^{-1} such that $x \otimes x^{-1} = \mathbb{1}$). Since $\mathbb{X}_+ = \mathbb{X} \setminus \{\mathbb{0}\}$ forms a multiplication group, the algebraic structure $\langle \mathbb{X}, \mathbb{0}, \mathbb{1}, \oplus, \otimes \rangle$ will be called an idempotent semifield.X

Using idempotent addition, we define the following partial order: $x \leq y$ if and only if $x \oplus y = y$. In terms of the proposed partial order for addition and

multiplication the monotonicity property is satisfied, under which, if the condition $x \leq y$ is satisfied, for any $z \in \mathbb{X}$ the inequalities $x \oplus z \leq y \oplus z$ and $xz \leq yz$ for any $x, y \in \mathbb{X}$. For inversion, the property of antitonicity is satisfied, which means that for all $x, y \neq \mathbb{0}$ the inequality $x \leq y$ implies the inequality $x^{-1} \geq y^{-1}$. Addition has the extreme property that inequalities $x \leq x \oplus y$ and $y \leq x \oplus y$ hold for all $x, y \in \mathbb{X}$. Moreover, the inequality $x \oplus y \leq z$ is true if and only if when the inequalities $x \leq z$ and $y \leq z$ are satisfied. We assume that the presented partial order is supplemented to a linear one.

For each $x \in \mathbb{X} \setminus \{\mathbb{0}\}$ define a positive integer power p as follows $x^0 = \mathbb{1}$, $x^p = x^{p-1}x$, $x^{-p} = (x^{-1})^p$, $\mathbb{0}^p = \mathbb{0}$. We consider that the operation of raising to an integer degree can be naturally spread in the case of a rational exponent.

As examples of an idempotent semifield $\langle \mathbb{X}, \mathbb{0}, \mathbb{1}, \oplus, \otimes \rangle$ one can consider real semifields

$$\begin{aligned} &\mathbb{R}_{\max,+} = \langle \mathbb{R} \cup \{-\infty\}, -\infty, 0, \max, + \rangle, && \mathbb{R}_{\min,+} = \langle \mathbb{R} \cup \{+\infty\}, +\infty, 0, \min, + \rangle, \\ &\mathbb{R}_{\max,\times} = \langle \mathbb{R}_+ \cup \{0\}, 0, 1, \max, \times \rangle, && \mathbb{R}_{\min,\times} = \langle \mathbb{R}_+ \cup \{+\infty\}, +\infty, 1, \min, \times \rangle, \end{aligned} \tag{2.1}$$

where $\mathbb{R}$ is the set of real numbers, $\mathbb{R}_+ = \{x \in \mathbb{R} | x > 0\}$.

For the semifield $\mathbb{R}_{\max,+}$ we define the zero of $\mathbb{0}$ as $-\infty$, and the unit $\mathbb{1}$ as 0. For each element $x \in \mathbb{R}$ there is an inverse element x^{-1}, which corresponds to $-x$ in standard notation. For each $x, y \in \mathbb{R}$, the degree of x^y is the same as the arithmetic product xy. The partial order relation given by idempotent addition coincides with the usual linear order relation on $\mathbb{R}$.

We define the binomial identity as

$$(x \oplus y)^\alpha = x^\alpha \oplus y^\alpha \tag{2.2}$$

for all $x, y \in \mathbb{X}$ and rational number $\alpha \geq 0$.

For every $x_1 \dots x_k \in \mathbb{X}$ holds tropical analog of the inequality between the geometric mean and the arithmetic mean

$$(x_1 \dots x_k)^{1/k} \leq x_1 \oplus \dots \oplus x_k. \tag{2.3}$$

2.1.3 Matrices and vectors

Denote by $\mathbb{X}^{m\times n}$ the set of matrices that consists of m rows and n columns with elements from $\mathbb{X}$. We define addition, multiplication, and multiplication by a scalar for size-matched matrices $\boldsymbol{A} = (a_{ij})$, $\boldsymbol{B} = (b_{ij})$, $\boldsymbol{C} = (c_{ij})$ using the following formulas:

$$\{\boldsymbol{A} \oplus \boldsymbol{B}\}_{ij} = a_{ij} \oplus b_{ij}, \quad \{\boldsymbol{B} \otimes \boldsymbol{C}\}_{ij} = \bigoplus_k b_{ik}c_{kj}, \quad \{x\boldsymbol{A}\}_{ij} = xa_{ij}. \tag{2.4}$$

We assume that the order relation and related properties are generalized to matrices and are understood component by component.

A matrix with all elements equal to $\mathbb{0}$ we consider as zero in $\mathbb{X}^{m\times n}$.

A matrix that does not have zero rows (columns) is regular in rows (columns). A regular matrix is a matrix that is regular in both rows and columns.

As usual, for any matrix $\boldsymbol{A}$ its transposed matrix is denoted as $\boldsymbol{A}^T$.

For any matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{X}^{m\times n}$, we define the multiplicatively conjugate matrix $\boldsymbol{A}^- = (a_{ij}^-) \in \mathbb{X}^{n\times m}$, where $a_{ij}^- = a_{ji}^{-1}$ if $a_{ji} \neq \mathbb{0}$ and $a_{ij}^- = \mathbb{0}$ otherwise.

The set of column vectors of dimension n is denoted by $\mathbb{X}^n$.

A vector with all components equal to $\mathbb{0}$ is called null and is denoted by $\mathbf{0} = (\mathbb{0}, \ldots, \mathbb{0})^T$. A vector without zero components is regular.

We denote by $\mathbf{1} = (\mathbb{1}, \ldots, \mathbb{1})^T$ the vector consisting of ones.

Let us define the multiplicatively conjugate vector for each nonzero vector $\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \mathbb{X}^n$ as a row vector $bmx^- = (x_1^-, \ldots, x_n^-)$, where $x_i^- = x_i^{-1}$ if $x_i \neq \mathbb{0}$, and $x_i^- = \mathbb{0}$ otherwise.

The vector $\boldsymbol{y} \in \mathbb{X}^n$ is collinear to the vector $\boldsymbol{x} \in \mathbb{X}^n$ if there is a scalar $c \in \mathbb{X}$ such that $\boldsymbol{y} = c\boldsymbol{x}$.

We define idempotent analogues of vector and matrix norms for any vector $\boldsymbol{x} \in \mathbb{X}^n$ and matrix $\boldsymbol{A} \in \mathbb{X}^{m\times n}$ by

$$||\boldsymbol{x}|| = \bigoplus_{i=1}^{n} x_i, \quad ||\boldsymbol{A}|| = \bigoplus_{i=1}^{m}\bigoplus_{j=1}^{n} a_{ij}. \tag{2.5}$$

Let us describe square matrices in $\mathbb{X}^{n\times n}$. We will call as identity matrix $\boldsymbol{I}$ with elements equal to $\mathbb{1}$ on the main diagonal and $\mathbb{0}$ outside it.

Let us define a non-negative integer power p for the square matrix $\boldsymbol{A} = (a_{ij})$ in the following way:

$$\boldsymbol{A}^0 = \boldsymbol{I}, \quad \boldsymbol{A}^p = \boldsymbol{A}^{p-1}\boldsymbol{A}. \tag{2.6}$$

The power of the sum of matrices can be represented by the expression

$$(\boldsymbol{A} \oplus \boldsymbol{B})^p = \bigoplus_{k=1}^{p} \bigoplus_{i_0+\cdots+i_k=p-k} \boldsymbol{B}^{i_0}(\boldsymbol{A}\boldsymbol{B}^{i_1} \cdots \boldsymbol{A}\boldsymbol{B}^{i_k}) \oplus \boldsymbol{B}^p. \tag{2.7}$$

For any matrix $\boldsymbol{A} \in \mathbb{X}^{n\times n}$, we define tropical analogues of the determinant and spectral radius of the matrix, which are calculated, respectively, by the formulas

$$\operatorname{tr}\boldsymbol{A} = a_{11} \oplus \cdots \oplus a_{nn}, \quad \operatorname{Tr}(\boldsymbol{A}) = \operatorname{tr}\boldsymbol{A} \oplus \cdots \oplus \operatorname{tr}\boldsymbol{A}^n, \quad \lambda = \operatorname{tr}\boldsymbol{A} \oplus \cdots \oplus \operatorname{tr}^{1/n}(\boldsymbol{A}^n). \tag{2.8}$$

The quantity $\operatorname{tr}\boldsymbol{A}$, which is called the trace of a matrix, has the following properties:

$$\operatorname{tr}(\boldsymbol{A} \oplus \boldsymbol{B}) = \operatorname{tr}\boldsymbol{A} \oplus \operatorname{tr}\boldsymbol{B}, \qquad \operatorname{tr}(\boldsymbol{A}\boldsymbol{B}) = \operatorname{tr}(\boldsymbol{B}\boldsymbol{A}), \qquad \operatorname{tr}(x\boldsymbol{A}) = x \operatorname{tr}\boldsymbol{A}. \tag{2.9}$$

If the condition $\operatorname{Tr}(\boldsymbol{A}) \le \mathbb{1}$ is satisfied, the Kleene matrix is defined as

$$\boldsymbol{A}^* = \boldsymbol{I} \oplus \boldsymbol{A} \oplus \cdots \oplus \boldsymbol{A}^{n-1}. \tag{2.10}$$

It is easy to see that from the expression (2.7) follows the equality for the Kleene matrix from the sum of matrices

$$(\boldsymbol{A} \oplus \boldsymbol{B})^* = \bigoplus_{k=1}^{n-1} \bigoplus_{m=0}^{n-k-1} \bigoplus_{i_0+i_1+\cdots+i_k=m} \boldsymbol{B}^{i_0}(\boldsymbol{A}\boldsymbol{B}^{i_1} \cdots \boldsymbol{A}\boldsymbol{B}^{i_k}) \oplus \boldsymbol{B}^*. \tag{2.11}$$

A matrix is called decomposable if it can be reduced to a block-triangular form by permuting the rows together with the same permutation of the columns. Otherwise, the matrix is indecomposable.

Note that a matrix with regular columns (rows) has only non-zero elements and is therefore indecomposable. For every indecomposable matrix, its Kleene matrix has no zero entries.

Any matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{X}^{n\times n}$ can be mapped to a directed graph $\langle V, E\rangle$ with set of peaks $V = \{1, \ldots, n\}$ and set of arcs $E = \{(i, j)|i, j \in V\}$. For each pair of vertices $i, j \in V$ the set E includes an arc (i, j) if $a_{ij} > \mathbb{0}$, and does not include such an arc otherwise.

A matrix is indecomposable if and only if its graph is strongly connected.

2.1.4 Solution of vector inequalities

The inequalities described in this section will be used in what follows. as constraint inequalities in tropical optimization problems, and the presented theorems will be used to solve tropical optimization problems.

Let us given the matrix $\boldsymbol{A} \in \mathbb{X}^{m\times n}$ and the vector $\boldsymbol{b} \in \mathbb{X}^m$. Let us write down the problem of solving the inequality with respect to the unknown vector $\boldsymbol{x} \in \mathbb{X}^n$

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \tag{2.12}$$

This problem can be solved using the following theorem:

Theorem 1 ([7]). *For a regular by columns matrix $\boldsymbol{A}$ and a regular vector $\boldsymbol{b}$, the vector $\boldsymbol{x}$ is a solution to inequality* (2.12) *if and only if*

$$\boldsymbol{x} \leq (\boldsymbol{b}^{-}\boldsymbol{A})^{-}. \tag{2.13}$$

For any matrix $\boldsymbol{A} \in \mathbb{X}^{n\times n}$ the inequality with respect to the unknown vector $\boldsymbol{x} \in \mathbb{X}^n$

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{x} \tag{2.14}$$

will be called a linear homogeneous inequality. Its solution can be found using the following result

Theorem 2 ([36]). *Let $\boldsymbol{x}$ be the general solution of a homogeneous inequality.*

Then if $\mathrm{Tr}(\boldsymbol{A}) \leq \mathbb{1}$ *then $\boldsymbol{x} = \boldsymbol{A}^{*}\boldsymbol{u}$ for all regular vectors $\boldsymbol{u}$. Otherwise, there are no regular solutions.*

The previous problem can be made more difficult by adding $\boldsymbol{b} \in \mathbb{X}^n$ vectors. The resulting inequality with respect to the unknown vector $\boldsymbol{x} \in \mathbb{X}^n$

$$\boldsymbol{A}\boldsymbol{x} \oplus \boldsymbol{b} \leq \boldsymbol{x} \tag{2.15}$$

will be called a linear inhomogeneous inequality. Let us formulate a theorem proposing a solution to such an inequality.

Theorem 3 ([20]). *Let $\boldsymbol{x} \in \mathbb{X}^n$ be a general regular solution of inequalities* (2.15). *Then the following statements are true:*

1. *If* $\mathrm{Tr}(\boldsymbol{A}) \leq \mathbb{1}$, *then* $\boldsymbol{x} = \boldsymbol{A}^* \boldsymbol{u}$ *for any regular* $\boldsymbol{u}$ *such as* $\boldsymbol{u} \geq \boldsymbol{b}$.

2. *If* $\mathrm{Tr}(\boldsymbol{A}) > \mathbb{1}$, *then there is no regular solutions.*

Let us note that the restrictions on the time frame for the execution of project jobs presented in section 2.1 are the inequalities discussed above, written in terms of the semifield $\mathbb{R}_{\max,+}$, which can be used in solving optimal planning problems.

2.2 Tropical optimization problems

Tropical optimization problems (optimization problems formulated in the language of idempotent mathematics) are usually defined in terms of minimizing (maximizing) functions on finite-dimensional semimodules over idempotent semifields under constraints in the form of linear equations and inequalities.

Such problems are widely used in solving many practical problems. For example, in the papers [20, 21, 23, 31, 33, 36, 37, 52, 53, 93–95] the solution of network planning problems is given by reducing them to tropical optimization problems. The main advantage of this approach is that, in comparison with the methods of mathematical programming, which often provide only an algorithmic solution, tropical optimization methods make it possible to explicitly obtain complete solution.

Problems with a non-linear objective function are considered below, since a significant part of network planning problems belong to this class of problems. Information about optimization problems with linear objective functions can be found in the papers [12, 21, 26, 31, 53, 95–102]. The 2.2.1 section provides a brief overview of various optimization problems, starting with [52, 53, 93], where optimization problems were considered as searching for a vector providing the best estimate in the Chebyshev norm ($\|\boldsymbol{x}\| = \max_{1 \leq i \leq n}(x_i)$). In paper [94] introduces the concept of an interval semi-norm, which is a special case of objective functions considered in

[36, 37] papers, which are used to solve the problems of managing project time schedules. Further, in the paragraph 2.2.2, tropical optimization problems are considered in more detail, which form the basis for finding solutions to constrained tropical optimization problems.

2.2.1 Examples of tropical optimization problems

Let us first consider the following problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & (\boldsymbol{A}\boldsymbol{x})^{-} \boldsymbol{d}, \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{d}. \end{aligned} \tag{2.16}$$

This problem can be considered as a search for a vector $\boldsymbol{x}$ that provides the best lower bound for the vector $\boldsymbol{d}$ using $\boldsymbol{A}\boldsymbol{x}$ in the Chebyshev norm. In the paper [52] presents a complete explicit solution based on the application of the abstract theory of linear operators over the semifield $\mathbb{R}_{\max,+}$. A similar solution is also considered in [53].

The following two problems, originally formulated in the usual form, can be represented in the language of idempotent algebra as tropical optimization problems with a nonlinear objective function and linear constraints. The article [93] investigates the problem of minimizing the Chebyshev-type distance function in $\mathbb{R}_{\max,+}$, which is solved using a threshold-type polynomial algorithm. This problem can be written in the form of an optimization problem with a non-linear objective function and constraints given by the two-sided inequality

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & (\boldsymbol{A}\boldsymbol{x})^{-} \boldsymbol{p} \oplus \boldsymbol{q}^{-} \boldsymbol{A}\boldsymbol{x}, \\ & \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{2.17}$$

The interval seminorm is the maximum spread between the vector components. The problem of minimizing the interval seminorm is considered, for example, in [94], where an explicit solution is proposed in mixed analytic form written in terms of the semifield $\mathbb{R}_{\max,+}$ and its dual semifield $\mathbb{R}_{\min,+}$. Note that the interval seminorm can be easily represented in terms of $\mathbb{R}_{\max,+}$ using multiplicatively conjugate vectors:

$$\min \quad \mathbb{1}^T \boldsymbol{A}\boldsymbol{x} \, (\boldsymbol{A}\boldsymbol{x})^{-} \mathbb{1}. \tag{2.18}$$

With the help of the connection between the solutions of two-sided equations (in which the unknown vector is present in both parts of the equation) described in paper [103], in [87] for the following problem with a nonlinear objective function

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{p}^T\boldsymbol{x}\left(\boldsymbol{q}^T\boldsymbol{x}\right)^{-1}, \\ & \boldsymbol{A}\boldsymbol{x} \oplus \boldsymbol{b} \leq \boldsymbol{C}\boldsymbol{x} \oplus \boldsymbol{d}, \end{aligned} \tag{2.19}$$

an iterative computational scheme for finding a solution in $\mathbb{R}_{\max,+}$ is presented.

Let us present a brief overview of problems presented in terms of a general linearly ordered semifield that admit direct explicit solutions in vector form. An analytical method for solving such problems based on new results of tropical spectral theory and solving linear tropical inequalities was proposed in [7, 104, 105].

In the article [36] was developed a complete solution to the problem without restrictions

$$\min_{\boldsymbol{x}} \quad \boldsymbol{d}^{-}\boldsymbol{A}\boldsymbol{x} \oplus (\boldsymbol{A}\boldsymbol{x})^{-}\boldsymbol{d}. \tag{2.20}$$

In [106], in turn, presents an explicit solution of another problem without restrictions in the form

$$\min_{\boldsymbol{x}} \quad (\boldsymbol{A}\boldsymbol{x})^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{A}\boldsymbol{x}. \tag{2.21}$$

In addition, it is proved that two problems with constraints in the form of linear inequalities

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{x}, \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{x}; \end{aligned} \tag{2.22}$$

and equalities

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{x}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}; \end{aligned} \tag{2.23}$$

can be reduced to the previous problem without restrictions, which makes it possible to obtain their direct explicit solutions.

Finally, [37] proposes a complete explicit solution to the problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{A}\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x} \end{aligned} \tag{2.24}$$

provided that at least one of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is indecomposable.

The problems described in the current paragraph form the basis for the results of tropical optimization presented below, which are in many respects the development and complication of these problems.

2.2.2 Examples of solving some problems of tropical optimization

Suppose we are given a matrix $\boldsymbol{A} \in \mathbb{X}^{m\times n}$ and vectors $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{X}^m$. It is necessary to calculate the regular vectors $\boldsymbol{x} \in \mathbb{X}^n$ for which the minimum in the problem is achieved

$$\min_{\boldsymbol{x}} \ \boldsymbol{q}^{-}\boldsymbol{x}(\boldsymbol{A}\boldsymbol{x})^{-}\boldsymbol{p}. \tag{2.25}$$

The article [37] found the following solution to problem (2.25).

Theorem 4. *Suppose that the matrix $\boldsymbol{A}$ is row-regular, the vector $\boldsymbol{p}$ is nonzero, and $\boldsymbol{q}$ is regular. Then in problem* (2.25) *the minimum is equal to*

$$\Delta = (\boldsymbol{A}\boldsymbol{q})^{-}\boldsymbol{p} \tag{2.26}$$

and is achieved on any vector

$$\boldsymbol{x} = \alpha\boldsymbol{q}, \qquad \alpha > \mathbb{0}. \tag{2.27}$$

Note that the problem of minimizing the interval seminorm (2.18), presented in the previous paragraph is a special case of the (2.25) problem.

$$\begin{aligned} \min_{\boldsymbol{x}} \ & \boldsymbol{x}^{-}\boldsymbol{A}\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \\ & \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}. \end{aligned} \tag{2.28}$$

We define auxiliary notation

$$\boldsymbol{S}_k = \bigoplus_{0\le i_1+\cdots+i_k\le n-k} \boldsymbol{A}\boldsymbol{B}^{i_1}\cdots\boldsymbol{A}\boldsymbol{B}^{i_k}, \qquad k = 1,\dots,n; \tag{2.29}$$

$$\boldsymbol{T}_k = \bigoplus_{0\le i_0+i_1+\cdots+i_k\le n-k-1} \boldsymbol{B}^{i_0}(\boldsymbol{A}\boldsymbol{B}^{i_1}\cdots\boldsymbol{A}\boldsymbol{B}^{i_k}), \qquad k = 1,\dots,n-1. \tag{2.30}$$

The article [35] proposes the following solution to the problem (2.28).

Theorem 5. *Let $\boldsymbol{A}$ be a matrix with non-negative spectral radius $\lambda > \mathbb{0}$, and $\boldsymbol{B}$ is a matrix such that $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$. Let $\boldsymbol{g}$ be a vector and $\boldsymbol{h}$ a regular vector such that $\boldsymbol{h}^{-}\boldsymbol{B}^{*}\boldsymbol{g} \leq \mathbb{1}$.*

Then the minimum value in problem (2.28) is equal to

$$\theta = \bigoplus_{k=1}^{n} \mathrm{tr}^{1/k}(\boldsymbol{S}_k) \oplus \bigoplus_{k=1}^{n-1} (\boldsymbol{h}^{-}\boldsymbol{T}_k\boldsymbol{g})^{1/k}, \tag{2.31}$$

and all regular solutions have the form

$$\boldsymbol{x} = (\theta^{-1}\boldsymbol{A} \oplus \boldsymbol{B})^{*}\boldsymbol{u}, \tag{2.32}$$

where $\boldsymbol{u}$ is any regular vector that satisfies the condition

$$\boldsymbol{g} \leq \boldsymbol{u} \leq (\boldsymbol{h}^{-}(\theta^{-1}\boldsymbol{A} \oplus \boldsymbol{B})^{*})^{-}. \tag{2.33}$$

Note that the (2.24) problem is a special case of the (2.28) problem with relaxed constraints.

Let the matrix $\boldsymbol{B}$ and the vectors $\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{p}$ and $\boldsymbol{q}$ be given. Consider the following problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{2.34}$$

Let us formulate a theorem proposing a solution to this problem

Theorem 6 ([24])**.** *Let $\boldsymbol{B}$ be a matrix satisfying the condition $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$. Let $\boldsymbol{g}$ be a vector, and $\boldsymbol{q}$ and $\boldsymbol{h}$ be regular vectors such that $\boldsymbol{h}^{-}\boldsymbol{B}^{*}\boldsymbol{g} \leq \mathbb{1}$.*

Then the minimum value of the objective function in problem (2.34) is equal to

$$\theta = (\boldsymbol{q}^{-}\boldsymbol{B}^{*}\boldsymbol{p})^{1/2} \oplus \boldsymbol{h}^{-}\boldsymbol{B}^{*}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{B}^{*}\boldsymbol{g}, \tag{2.35}$$

and all regular solutions have the form

$$\boldsymbol{x} = \boldsymbol{B}^{*}\boldsymbol{u} \tag{2.36}$$

where $\boldsymbol{u}$ is any regular vector that satisfies the conditions

$$\boldsymbol{g} \oplus \theta^{-1}\boldsymbol{p} \leq \boldsymbol{u} \leq ((\boldsymbol{h}^{-} \oplus \theta^{-1}\boldsymbol{q}^{-})\boldsymbol{B}^{*})^{-}. \tag{2.37}$$

Let us present the result of the article [33], suggesting a solution for the following optimization problem. Let us given matrices $\boldsymbol{A} \in \mathbb{X}^{n\times n}$ and vectors $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{X}^n$. It is necessary to find a regular solution $\boldsymbol{x} \in \mathbb{X}^n$ of the problem

$$\max_{\boldsymbol{x}} \ \boldsymbol{q}^- \boldsymbol{x} \, (\boldsymbol{A}\boldsymbol{x})^- \boldsymbol{p}. \tag{2.38}$$

Let us formulate a theorem that offers a direct complete solution to the (2.38) problem.

Theorem 7. *Let the matrix $\boldsymbol{A} = (\boldsymbol{a_j})$ has only regular columns $\boldsymbol{a}_j = (a_{ij})$, and the vectors $\boldsymbol{p} = (p_j)$ and $\boldsymbol{q} = (q_j)$ are regular.*

Then the maximum in the (2.38) *problem is equal to*

$$\Delta = \boldsymbol{q}^- \boldsymbol{A}^- \boldsymbol{p} \tag{2.39}$$

and is reached if and only if the vector $\boldsymbol{x} = (x_j)$ has components

$$x_k = \alpha a_{sk}^{-1} p_s, \quad x_j \le \alpha a_{sj}^{-1} p_s, \quad j \ne k, \tag{2.40}$$

for all $\alpha > \mathbb{0}$ and indices k and s defined by the conditions

$$k = \arg\max_{1\le j\le n} q_j^{-1} \boldsymbol{a}_j^- \boldsymbol{p}, \quad s = \arg\max_{1\le j\le n} a_{jk}^{-1} p_j. \tag{2.41}$$

Note that the maximization of the interval seminorm is a special case of the problem (2.38).

Chapter 3

Minimizing of project duration

In this chapter, the problem of drawing up an optimal calendar schedule for the implementation of project jobs is formulated and solved, which consists in minimizing the total duration of the project with given restrictions on the sequence of jobs. The proposed solution method consists in presenting the planning problem in the terms of tropical mathematics and then solving it using tropical optimization methods. In section 3.2 the optimal control problem is formulated and its solution is given, based on the representation of the problem in the form of a tropical optimization problem. Finally, in section 3.3 an example of the practical use of the obtained result for solving the problem of the fastest possible vaccination is presented.

3.1 Solving a tropical optimization problem with constraints

Suppose we are given a matrix $\boldsymbol{B} \in \mathbb{X}^{n\times n}$ and the vectors $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{g}, \boldsymbol{h} \in \mathbb{X}^n$. Let us study the following tropical optimization problem:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^- \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{x}; \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}. \end{aligned} \tag{3.1}$$

To find a solution to the problem we will use result of the theorem 5. The resulting solution, then, taking into account the special form of the objective function of the problem, will be reduced to a more compact form. Let us formulate and prove the following theorem.

Theorem 8. *Let $\boldsymbol{B}$ be a matrix such that $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$, $\boldsymbol{g}$ is a vector, and $\boldsymbol{h}$ is a regular vector such that the inequality $\boldsymbol{h}^- \boldsymbol{B}^* \boldsymbol{g} \le \mathbb{1}$ holds. We assume that the vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ are nonzero, such that $\boldsymbol{q}^T \boldsymbol{p} > \mathbb{0}$.*

Then the minimum value of the objective function in problem (3.1) *is equal to*

$$\theta = \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p} \oplus \bigoplus_{0\le i+j\le n-2} (\boldsymbol{h}^- \boldsymbol{B}^i \boldsymbol{p})(\boldsymbol{q}^T \boldsymbol{B}^j \boldsymbol{g}), \tag{3.2}$$

and all regular solutions have the form

$$\boldsymbol{x} = \boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{G} = \boldsymbol{B}^* \oplus \bigoplus_{0 \le i+j \le n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^j, \tag{3.3}$$

where $\boldsymbol{u}$ *is any regular vector that satisfies the conditions*

$$\boldsymbol{g} \le \boldsymbol{u} \le (\boldsymbol{h}^- \boldsymbol{G})^-. \tag{3.4}$$

Proof. Since $\boldsymbol{q}^T \boldsymbol{p} > \mathbb{0}$ the spectral radius of the matrix $\boldsymbol{p}\boldsymbol{q}^T$ is nonzero. Because problem (3.1) has the form (2.28), we can use the theorem 5. First, by the formula (2.31), we calculate the minimum of θ. Substituting the equality $\boldsymbol{A} = \boldsymbol{p}\boldsymbol{q}^T$ into the expression $\boldsymbol{A}\boldsymbol{B}^{i_1} \cdots \boldsymbol{A}\boldsymbol{B}^{i_k}$ and writing out the scalar factors, we obtain the following equality

$$\boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_1} \cdots \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k} = (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k}. \tag{3.5}$$

Thus the equality (2.29) takes the form

$$\boldsymbol{S}_k = \bigoplus_{0 \le i_1 + \cdots + i_k \le n-k} (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k}. \tag{3.6}$$

We calculate the trace of the matrix under the sum sign. Taking into account the trace properties, we have

$$\operatorname{tr}((\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k}) = \boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_k} \boldsymbol{p}. \tag{3.7}$$

Using the obtained result, we write the following sum

$$\bigoplus_{k=1}^{n} \operatorname{tr}^{1/k}(\boldsymbol{S}_k) = \bigoplus_{k=1}^{n} \bigoplus_{0 \le i_1 + \cdots + i_k \le n-k} (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_k} \boldsymbol{p})^{1/k}. \tag{3.8}$$

Consider the first term corresponding to $k = 1$ in the sums on the left and right, in more detail

$$\operatorname{tr}(\boldsymbol{S}_1) = \bigoplus_{0 \le i \le n-1} \boldsymbol{q}^T \boldsymbol{B}^i \boldsymbol{p} = \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p}. \tag{3.9}$$

Let us prove that the other terms do not exceed the first one. Indeed, using the inequality between geometric and arithmetic means, we get

$$(\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_k} \boldsymbol{p})^{1/k} \le \boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \oplus \cdots \oplus \boldsymbol{q}^T \boldsymbol{B}^{i_k} \boldsymbol{p}, \tag{3.10}$$

hence for all $k = 2, \dots, n$ we have

$$\bigoplus_{0\le i_1+\dots+i_k\le n-k} (\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{p})^{1/k} \le \bigoplus_{i=0}^{n-k} \boldsymbol{q}^T\boldsymbol{B}^i\boldsymbol{p} \le \bigoplus_{i=0}^{n-1} \boldsymbol{q}^T\boldsymbol{B}^i\boldsymbol{p} = \operatorname{tr}(\boldsymbol{S}_1). \tag{3.11}$$

Since the first term dominates over the rest, we have

$$\bigoplus_{k=1}^{n} \operatorname{tr}^{1/k}(\boldsymbol{S}_k) = \operatorname{tr}(\boldsymbol{S}_1) = \boldsymbol{q}^T\boldsymbol{B}^*\boldsymbol{p}. \tag{3.12}$$

Then, using the equality (3.5), we represent (2.30) in the following form

$$\boldsymbol{T}_k = \bigoplus_{0\le i_0+i_1+\dots+i_k\le n-k-1} (\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})\boldsymbol{B}^{i_0}\boldsymbol{p}\boldsymbol{q}^T\boldsymbol{B}^{i_k}. \tag{3.13}$$

Multiplying the matrices under the summation sign on the left by $\boldsymbol{h}^-$ and on the right by $\boldsymbol{g}$ we obtain the equality

$$\boldsymbol{h}^-(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})\boldsymbol{B}^{i_0}\boldsymbol{p}\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g} = \\ = (\boldsymbol{h}^-\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}). \tag{3.14}$$

Taking into account the obtained equality, we write out the sum

$$\bigoplus_{k=1}^{n-1}(\boldsymbol{h}^-\boldsymbol{T}_k\boldsymbol{g})^{1/k} = \\ = \bigoplus_{k=1}^{n-1} \bigoplus_{0\le i_0+i_1+\dots+i_k\le n-k-1} ((\boldsymbol{h}^-\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}))^{1/k} \tag{3.15}$$

and consider the first term, which corresponds to $k = 1$ and has the form

$$\boldsymbol{h}^-\boldsymbol{T}_1\boldsymbol{g} = \bigoplus_{0\le i_0+i_1\le n-2} (\boldsymbol{h}^-\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{g}). \tag{3.16}$$

Let us make sure that the value of other terms does not exceed the sum of $\operatorname{tr}(\boldsymbol{S}_1) \oplus \boldsymbol{h}^-\boldsymbol{T}_1\boldsymbol{g}$. Using the inequality between geometric and arithmetic means, we obtain

$$((\boldsymbol{h}^-\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\cdots\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}))^{1/k} \le \\ \le (\boldsymbol{q}^T\boldsymbol{B}^{i_1}\boldsymbol{p}\oplus\cdots\oplus\boldsymbol{q}^T\boldsymbol{B}^{i_{k-1}}\boldsymbol{p}) \oplus (\boldsymbol{h}^-\boldsymbol{B}^{i_0}\boldsymbol{p})(\boldsymbol{q}^T\boldsymbol{B}^{i_k}\boldsymbol{g}). \tag{3.17}$$

Thus, for all $k = 2, \ldots, n$ holds the inequality

$$\bigoplus_{0 \le i_0 + i_1 + \cdots + i_k \le n-k-1} ((\boldsymbol{h}^- \boldsymbol{B}^{i_0} \boldsymbol{p})(\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p})(\boldsymbol{q}^T \boldsymbol{B}^{i_k} \boldsymbol{g}))^{1/k} \le$$

$$\le \bigoplus_{i=0}^{n-1} \boldsymbol{q}^T \boldsymbol{B}^i \boldsymbol{p} \oplus \bigoplus_{0 \le i+j \le n-2} (\boldsymbol{h}^- \boldsymbol{B}^i \boldsymbol{p})(\boldsymbol{q}^T \boldsymbol{B}^j \boldsymbol{g}) = \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p} \oplus \boldsymbol{h}^- \boldsymbol{T}_1 \boldsymbol{g}. \quad (3.18)$$

Taking into account the last relation, we obtain the double inequality

$$\boldsymbol{h}^- \boldsymbol{T}_1 \boldsymbol{g} \le \bigoplus_{k=1}^{n-1} (\boldsymbol{h}^- \boldsymbol{T}_k \boldsymbol{g})^{1/k} \le \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p} \oplus \boldsymbol{h}^- \boldsymbol{T}_1 \boldsymbol{g}. \quad (3.19)$$

To get (3.2), it remains to combine the results:

$$\theta = \bigoplus_{k=1}^{n} \mathrm{tr}^{1/k}(\boldsymbol{S}_k) \oplus \bigoplus_{k=1}^{n-1} (\boldsymbol{h}^- \boldsymbol{T}_k \boldsymbol{g})^{1/k} = \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p} \oplus \bigoplus_{0 \le i+j \le n-2} (\boldsymbol{h}^- \boldsymbol{B}^i \boldsymbol{p})(\boldsymbol{q}^T \boldsymbol{B}^j \boldsymbol{g}). \quad (3.20)$$

Based on the formulas (3.3) and (3.4) Let us move on to a parametric representation of the entire set of solutions of the problem. Let us find the Kleene matrix $(\theta^{-1} \boldsymbol{A} \oplus \boldsymbol{B})^* = (\theta^{-1} \boldsymbol{p}\boldsymbol{q}^T \oplus \boldsymbol{B})^*$ and apply the identity (2.11). As before, we will use the equality (3.5) to transform the expression under the sum sign and get

$$(\theta^{-1} \boldsymbol{p}\boldsymbol{q}^T \oplus \boldsymbol{B})^* = \boldsymbol{B}^* \oplus \bigoplus_{k=1}^{n-1} \bigoplus_{m=0}^{n-k-1} \bigoplus_{i_0 + \ldots + i_k = m} \theta^{-m} \boldsymbol{B}^{i_0} \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k} (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}). \quad (3.21)$$

Consider the sum on the right side and write the term for $k = 1$ as

$$\bigoplus_{m=0}^{n-2} \bigoplus_{i_0 + i_1 = m} \theta^{-1} \boldsymbol{B}^{i_0} \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_1} = \bigoplus_{0 \le i+j \le n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^j. \quad (3.22)$$

Note that from the resulting expression (3.2) for the minimum of the objective function of the problem, it follows that for all $i = 0, \ldots, n-1$ holds the inequalities $\theta \ge \boldsymbol{q}^T \boldsymbol{B}^i \boldsymbol{p}$ and, moreover, since the vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ are non-zero, we have $\theta \ge \boldsymbol{q}^T \boldsymbol{I} \boldsymbol{p} > \mathbb{0}$. Considering that then for all i the inequalities $\theta^{-1} \boldsymbol{q}^T \boldsymbol{B}^i \boldsymbol{p} \le \mathbb{1}$, we get the inequality

$$\theta^{-1} \boldsymbol{B}^{i_0} \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k} (\theta^{-1} \boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \theta^{-1} \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \le \theta^{-1} \boldsymbol{B}^{i_0} \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k}. \quad (3.23)$$

In view of the last inequality, for all $k = 2, \ldots, n-1$ we have

$$\bigoplus_{m=0}^{n-k-1} \bigoplus_{i_0+\ldots+i_k=m} \theta^{-m} \boldsymbol{B}^{i_0} \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^{i_k} (\boldsymbol{q}^T \boldsymbol{B}^{i_1} \boldsymbol{p} \cdots \boldsymbol{q}^T \boldsymbol{B}^{i_{k-1}} \boldsymbol{p}) \leq \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^j, \tag{3.24}$$

whence it follows that the term of the sum for $k = 1$ is greater than the rest, which can be discarded. Thus, the Kleene matrix takes the form

$$(\theta^{-1} \boldsymbol{p}\boldsymbol{q}^T \oplus \boldsymbol{B})^* = \boldsymbol{B}^* \oplus \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^j = \boldsymbol{G}. \tag{3.25}$$

After substituting the resulting expression into the formulas (2.32) and (2.33) we obtain a parametric description of the problem solutions in the form (3.3) and (3.4).

Let us find the computational complexity of solving the problem. First, we estimate the complexity of finding the matrix

$$\boldsymbol{G} = \boldsymbol{B}^* \oplus \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{B}^i \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^j, \tag{3.26}$$

which consists in calculating the Kleene matrix, requiring at most $O(n^4)$ scalar operations, and sum of the product of the following matrices

$$\bigoplus_{l=0}^{l=n-2} \boldsymbol{M_l}, \quad \boldsymbol{M_l} = \theta^{-1} \boldsymbol{B}^i \boldsymbol{p}\boldsymbol{q}^T \boldsymbol{B}^j, \quad l = i + j. \tag{3.27}$$

whose calculation is to calculate $1 + \cdots + n - 2 = (n-2)(n-1)/2$ matrices $\boldsymbol{M_l}$.

For finding all powers of the matrix $\boldsymbol{B}$ requires $O(n^4)$ operations. Calculation of the vectors $\boldsymbol{B}^i \boldsymbol{p}$ and $\boldsymbol{q}^T \boldsymbol{B}^j$, after finding the powers of the matrix $\boldsymbol{B}$, would require $O(n^2)$ scalar operations. The subsequent search for each matrix $\boldsymbol{M_l}$ also requires $O(n^2)$ operations. Therefore, $O(n^4)$ operations are required to find auxiliary matrices and vectors and $O(n^4)$ operations to calculate the sum $\bigoplus_{l=0}^{l=n-2} \boldsymbol{M_l}$.

Minimum calculation

$$\theta = \boldsymbol{q}^T \boldsymbol{B}^* \boldsymbol{p} \oplus \bigoplus_{l=0}^{l=n-2} \psi_l, \quad \psi_l = (\boldsymbol{h}^- \boldsymbol{B}^i \boldsymbol{p})(\boldsymbol{q}^T \boldsymbol{B}^j \boldsymbol{g}), \quad i + j = l. \tag{3.28}$$

consists in calculating n powers of the matrix $\boldsymbol{B}$, finding the Kleene matrix, the scalar $\boldsymbol{q}^T\boldsymbol{B}^*\boldsymbol{p}$ and $1+\cdots+n-2=(n-2)(n-1)/2$ scalar ψ_l. Computing $n-1$ powers of the matrix and the Kleene matrix will require no more than $O(n^4)$ operations.

3.2 Minimizing the maximum project duration

We study the problem of drawing up an optimal jobexecution plan in accordance with the criterion for the minimum maximum duration of the project (1.7) under all the above restrictions (with a full set of restrictions). Let us formulate the problem under study more formally.

$$\begin{aligned}\min_{x_i,y_i}\quad & \max_{1\le i\le n} y_i+\max_{1\le i\le n}(-x_i);\\ & \max_{1\le j\le n}(b_{ij}+x_j)\le x_i,\quad \max_{1\le j\le n}(c_{ij}+x_j)=y_i,\\ & \max_{1\le j\le n}(d_{ij}+y_j)\le x_i,\quad g_i\le x_i\le h_i,\quad y_i\le f_i,\quad i=1,\dots,n.\end{aligned}\tag{3.29}$$

It is easy to see that the problem (3.29) can be represented as a linear programming problem and solved numerically using known computational procedures. Further, using the methods of tropical mathematics, a direct analytical solution of the problem under consideration will be constructed.

Note that in terms of the idempotent semifield $\mathbb{R}_{\max,+}$, the objective function

$$\max_{1\le i\le n} y_i+\max_{1\le i\le n}(-x_i)\tag{3.30}$$

can be represented as follows:

$$\bigoplus_{1\le i\le n} y_i\bigoplus_{1\le j\le n} x_j^{-1}.\tag{3.31}$$

Let us rewrite constraints

$$\max_{1\le j\le n}(b_{ij}+x_j)\le x_i,\ \max_{1\le j\le n}(c_{ij}+x_j)=y_i,\ \max_{1\le j\le n}(d_{ij}+y_j)\le x_i,\ i=1,\dots,n,\tag{3.32}$$

in tropical form

$$\bigoplus_{1\le j\le n} b_{ij}x_j\le x_i,\quad \bigoplus_{1\le j\le n} c_{ij}x_j=y_i,\quad \bigoplus_{1\le j\le n} d_{ij}y_j\le x_i,\quad i=1,\dots,n.\tag{3.33}$$

Taking into account that the usual order relation holds in the algebra $\mathbb{R}_{\max,+}$, we represent problem (3.29) in tropical form:

$$\begin{aligned} \min_{x_i,y_i} \quad & \bigoplus_{1\le i\le n} y_i \bigoplus_{1\le j\le n} x_j^{-1}, \\ & \bigoplus_{1\le j\le n} b_{ij}x_j \le x_i, \quad \bigoplus_{1\le j\le n} c_{ij}x_j = y_i, \\ & \bigoplus_{1\le j\le n} d_{ij}y_j \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1,\dots,n. \end{aligned} \tag{3.34}$$

We define vector notation

$$\begin{gathered} \boldsymbol{B} = (b_{ij}), \quad \boldsymbol{C} = (c_{ij}), \quad \boldsymbol{D} = (d_{ij}), \\ \boldsymbol{x} = (x_i), \quad \boldsymbol{y} = (y_i), \quad \boldsymbol{f} = (f_i), \quad \boldsymbol{g} = (g_i), \quad \boldsymbol{h} = (h_i) \end{gathered} \tag{3.35}$$

and write the problem in matrix-vector form

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{x}^{-}\mathbf{1}\mathbf{1}^T\boldsymbol{y}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} = \boldsymbol{y}, \\ & \boldsymbol{D}\boldsymbol{y} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}, \quad \boldsymbol{y} \le \boldsymbol{f}. \end{aligned} \tag{3.36}$$

Let us solve the problem using the result of the theorem 8.

Theorem 9. *Let $\boldsymbol{B}$ and $\boldsymbol{D}$ be matrices, $\boldsymbol{C}$ be a column-regular matrix such that the matrix $\boldsymbol{R} = \boldsymbol{B}$ oplus$\boldsymbol{DC}$ satisfies* $\mathrm{Tr}(\boldsymbol{R}) \le \mathbb{1}$. *Let $\boldsymbol{g}$ be a vector, and $\boldsymbol{f}$ and $\boldsymbol{h}$ be regular vectors such that the vector $\boldsymbol{s}^T = \boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-}$ satisfies $\boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{g} \le \mathbb{1}$.*

Then the minimum value of the objective function in problem (3.36) *is equal to*

$$\theta = \|\boldsymbol{C}\boldsymbol{R}^*\| \oplus \bigoplus_{0\le i+j\le n-2} \|\boldsymbol{s}^T\boldsymbol{R}^i\|\|\boldsymbol{C}\boldsymbol{R}^j\boldsymbol{g}\|, \tag{3.37}$$

and all regular solutions have the form

$$\boldsymbol{x} = \boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{G} = \boldsymbol{R}^* \oplus \bigoplus_{0\le i+j\le n-2} \theta^{-1}\boldsymbol{R}^i\mathbf{1}\mathbf{1}^T\boldsymbol{C}\boldsymbol{R}^j, \tag{3.38}$$

where $\boldsymbol{u}$ is any regular vector that satisfies the conditions

$$\boldsymbol{g} \le \boldsymbol{u} \le (\boldsymbol{s}^T\boldsymbol{G})^{-}. \tag{3.39}$$

Proof. Substituting $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ into the problem, we will get

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\mathbf{1}\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{x}, \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{3.40}$$

By using theorem 1, we get that $\boldsymbol{x} \leq (\boldsymbol{f}^{-}\boldsymbol{C})^{-}$. Considering that the inversion has the antitonicity property, we rewrite the inequalities that bound the vector $\boldsymbol{x}$ from above

$$\boldsymbol{x}^{-} \geq \boldsymbol{f}^{-}\boldsymbol{C}, \quad \boldsymbol{x}^{-} \geq \boldsymbol{h}^{-}. \tag{3.41}$$

Combining the two inequalities into one, we will get $\boldsymbol{x}^{-} \geq \boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-}$, или $\boldsymbol{x} \leq (\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})^{-} = (\boldsymbol{s}^{T})^{-}$.

Note that the original problem can be written in the form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\mathbf{1}\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{x}; \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq (\boldsymbol{s}^{T})^{-}. \end{aligned} \tag{3.42}$$

Thus, the problem comes down to a problem 8, where $\boldsymbol{p} = \mathbf{1}$, $\boldsymbol{q}^{T} = \mathbf{1}^{T}\boldsymbol{C}$, while the roles of the matrix $\boldsymbol{B}$ and the vector $\boldsymbol{h}$ are respectively the matrix $\boldsymbol{R}$ and the vector $(\boldsymbol{s}^{T})^{-}$.

Since the matrix $\boldsymbol{C}$ is column regular, value $\boldsymbol{q}^{T}\boldsymbol{p} = \mathbf{1}^{T}\boldsymbol{C}\mathbf{1} = \|\boldsymbol{C}\| > \mathbb{0}$ and, therefore, the conditions of the theorem 8 are satisfied.

Note that, by the definition of the matrix norm,

$$\mathbf{1}^{T}\boldsymbol{C}\boldsymbol{R}^{*}\mathbf{1} = \|\boldsymbol{C}\boldsymbol{R}^{*}\|, \tag{3.43}$$

and, by definition of the vector norm,

$$\boldsymbol{h}^{-}\boldsymbol{R}^{i}\mathbf{1} = \|\boldsymbol{h}^{-}\boldsymbol{R}^{i}\|, \quad \mathbf{1}^{T}\boldsymbol{C}\boldsymbol{R}^{j}\boldsymbol{g} = \|\boldsymbol{C}\boldsymbol{R}^{j}\boldsymbol{g}\|. \tag{3.44}$$

using the theorem 8 we get the desired result.

Note that in the context of time planning problems in project sheduling, the elements of the unknown vector $\boldsymbol{x}$ determine the start time of job, and the elements of the vector $\boldsymbol{f}$ set the latest completion time of the job. Since the indicated time is defined(i.e. greater than $\mathbb{0} = -\infty$), the vectors $\boldsymbol{x}$ and $\boldsymbol{f}$ are regular. Conditions

$\mathrm{Tr}(\boldsymbol{R}) \leq \mathbb{1}$ and $\boldsymbol{s}^T \boldsymbol{R}^* \boldsymbol{g} \leq \mathbb{1}$ ensure the consistency of restrictions of the form «start-start», «start-finish» and «finish-start», together with the given release dates and directive jobs deadlines.

Note that, as in the previous problem, the computational complexity of finding a solution does not exceed $O(n^4)$.

3.3 The fastest vaccination problem

The events of the last few years have shown that the problem of the speedy implementation of measures for the vaccination of the population is extremely urgent. and vaccination may be required, followed by mandatory observation of the patient. Especially in case of insufficient clinical trials of the vaccine or lack of time to carry out work to reduce the side effects of the drug.

Suppose that there is a need to carry out measures for the vaccination of citizens. We will assume that patients need to be observed for some time after vaccination. It is necessary to vaccinate all citizens in a minimum period of time.

Let us introduce the following notation:

x_i – time of patient i vaccination;

y_i – end time of observation period after vaccine application of patient i;

b_{ij} – minimum time period between vaccination of patient $j = 1, \ldots, n$ and vaccination of patient i (may be due to medical and bureaucratic restrictions such as isolation of patients from each other, replenishment vaccine stocks, sanitizing the premises, paperwork);

c_{ij} – minimum time interval between vaccination of patient $j = 1, \ldots, n$ and completion of observation of patient i (due to limited resources of the vaccination station);

d_{ij} – minimum period of time between completion of observation of patient $j = 1, \ldots, n$ and vaccination of patient i (due to limited resources, for example in case of risk of hospitalization);

g_i – the earliest possible time for vaccination of patient i.

h_i – the latest allowed vaccination time of patient i.

f_i is the latest reasonable time to complete a patient i follow-up.

Note that the problem under consideration is a special case of the (3.29) problem, which can be represented in terms of the semifield $\mathbb{R}_{\max,+}$ in the form (3.36).

Let us consider a conditional medical center that needs to organize vaccination of $n = 3$ citizens. Time constraints are specified using the following matrices and vectors:

$$\boldsymbol{B} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -2 & -3 & 0 \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} -4 & -5 & -6 \\ \mathbb{0} & -4 & -7 \\ -5 & \mathbb{0} & -4 \end{pmatrix}, \tag{3.45}$$

$$\boldsymbol{g} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix} \quad \boldsymbol{f} = \begin{pmatrix} 6 \\ 6 \\ 6 \end{pmatrix}. \tag{3.46}$$

In order to use the result of the theorem 9, it is necessary to check the conditions.

Let us find auxiliary matrices:

$$\boldsymbol{DC} = \begin{pmatrix} 0 & -2 & -1 \\ -1 & -3 & -2 \\ -1 & -3 & -1 \end{pmatrix}, \quad \boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -1 & -3 & 0 \end{pmatrix}. \tag{3.47}$$

Let us calculate the powers of the matrix $\boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC}$:

$$\boldsymbol{R}^2 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & -1 & 2 \\ -1 & -2 & 0 \end{pmatrix}, \quad \boldsymbol{R}^3 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix}. \tag{3.48}$$

Then we calculate the values of the traces $\operatorname{tr} \boldsymbol{R} = \operatorname{tr} \boldsymbol{R}^2 = \operatorname{tr} \boldsymbol{R}^3 = 0$. Since the value

$$\operatorname{Tr}(\boldsymbol{R}) = \operatorname{tr} \boldsymbol{R} \oplus \operatorname{tr} \boldsymbol{R}^2 \oplus \operatorname{tr} \boldsymbol{R}^3 = 0 \tag{3.49}$$

condition $\mathrm{Tr}(\boldsymbol{R}) \leq \mathbb{1} = 0$ holds. Next, we calculate

$$\boldsymbol{R}^* = \boldsymbol{I} \oplus \boldsymbol{R} \oplus \boldsymbol{R}^2 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix}, \qquad \boldsymbol{R}^*\boldsymbol{g} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \tag{3.50}$$

$$\boldsymbol{f}^-\boldsymbol{C} = \begin{pmatrix} -2 & -4 & -3 \end{pmatrix}, \qquad \boldsymbol{f}^-\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{g} = -1, \qquad \boldsymbol{h}^-\boldsymbol{B}^*\boldsymbol{g} = -1. \tag{3.51}$$

Thus, we get that the condition $\boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{g} = (\boldsymbol{f}^-\boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{R}^*\boldsymbol{g} = -1 < \mathbb{1}$ is performed.

Let us find minimum spread θ. First we will write down an auxiliary vectors

$$\boldsymbol{s}^T = \begin{pmatrix} -2 & -3 & -2 \end{pmatrix}, \quad \boldsymbol{s}^T\boldsymbol{R} = \begin{pmatrix} -2 & -3 & -1 \end{pmatrix}, \tag{3.52}$$

$$\mathbf{1}^T\boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \end{pmatrix}, \quad \boldsymbol{R}\boldsymbol{g} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}. \tag{3.53}$$

Let us find the norms

$$\|\boldsymbol{C}\boldsymbol{R}^*\| = 5, \quad \|\boldsymbol{C}\boldsymbol{g}\| = 4, \quad \|\boldsymbol{C}\boldsymbol{R}\boldsymbol{g}\| = 5, \quad \|\boldsymbol{s}^T\| = -2, \quad \|\boldsymbol{s}^T\boldsymbol{R}\| = -1. \tag{3.54}$$

Parameter value θ can be calculated using the formula

$$\theta = \|\boldsymbol{C}\boldsymbol{R}^*\| \oplus \|\boldsymbol{s}^T\|\|\boldsymbol{C}\boldsymbol{g}\| \oplus \|\boldsymbol{s}^T\boldsymbol{R}\|\|\boldsymbol{C}\boldsymbol{g}\| \oplus \|\boldsymbol{s}^T\|\|\boldsymbol{C}\boldsymbol{R}\boldsymbol{g}\| \oplus \|\boldsymbol{s}^T\boldsymbol{R}\|\|\boldsymbol{C}\boldsymbol{R}\boldsymbol{g}\| = 5. \tag{3.55}$$

Let us represent all solutions $\boldsymbol{x}$ of the problem in parametric form. Find auxiliary matrices:

$$\mathbf{1}\mathbf{1}^T\boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 4 & 2 & 3 \\ 4 & 2 & 3 \end{pmatrix}, \quad \boldsymbol{R}\mathbf{1}\mathbf{1}^T\boldsymbol{C} = \begin{pmatrix} 5 & 3 & 4 \\ 6 & 4 & 5 \\ 4 & 2 & 3 \end{pmatrix}, \mathbf{1}\mathbf{1}^T\boldsymbol{C}\boldsymbol{R} = \begin{pmatrix} 4 & 3 & 5 \\ 4 & 3 & 5 \\ 4 & 3 & 5 \end{pmatrix}. \tag{3.56}$$

Let us calculate the following matrices and vector:

$$\boldsymbol{G} = \boldsymbol{R}^* \oplus (-5)(\mathbf{1}\mathbf{1}^T\boldsymbol{C} \oplus \boldsymbol{R}\mathbf{1}\mathbf{1}^T\boldsymbol{C} \oplus \mathbf{1}\mathbf{1}^T\boldsymbol{C}\boldsymbol{R}) = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix},$$

$$\boldsymbol{C}\boldsymbol{G} = \begin{pmatrix} 4 & 3 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix}, \qquad (\boldsymbol{s}^T\boldsymbol{G})^- = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \quad (3.57)$$

Let us write down all solutions $\boldsymbol{x} = (x_1, x_2, x_3)^T$ using the parameter vector $\boldsymbol{u} = (u_1, u_2, u_3)^T$

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix} \boldsymbol{u}, \qquad \boldsymbol{y} = \begin{pmatrix} 4 & 3 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix} \boldsymbol{u}, \qquad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \le \boldsymbol{u} \le \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \quad (3.58)$$

Since the columns of the matrices $\boldsymbol{G}$ and $\boldsymbol{C}\boldsymbol{G}$ are collinear, we write them as follows

$$\boldsymbol{G} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -1 & -2 \end{pmatrix} \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix}, \quad \boldsymbol{C}\boldsymbol{G} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix}. \quad (3.59)$$

Let us define an auxiliary vector $\boldsymbol{v}$

$$\boldsymbol{v} = \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix} \boldsymbol{u}, \quad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \le \boldsymbol{u} \le \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 0 \end{pmatrix} \le \boldsymbol{v} \le \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad (3.60)$$

and use it to write the solution

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -1 & -2 \end{pmatrix} \boldsymbol{v}, \qquad \boldsymbol{y} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \end{pmatrix} \boldsymbol{v}, \qquad \begin{pmatrix} 0 \\ 0 \end{pmatrix} \le \boldsymbol{v} \le \begin{pmatrix} 2 \\ 3 \end{pmatrix}. \quad (3.61)$$

Let us represent the solutions in terms of ordinary mathematics

$$\begin{aligned} &x_1 = \max(v_1, v_2 - 1), && x_2 = \max(v_1 + 1, v_2), \quad x_3 = \max(v_1 - 1, v_2 - 2), \\ &y_1 = \max(v_1 + 4, v_2 + 3), && y_2 = \max(v_1 + 3, v_2 + 2), \quad y_3 = \max(v_1 + 2, v_2 + 1), \\ & && 0 \le v_1 \le 2, \quad 0 \le v_2 \le 3. \end{aligned} \tag{3.62}$$

Chapter 4

Minimizing the deviation from the due dates

Let us consider the problem of drawing up an optimal schedule to minimize the maximum deviation from the due dates for the execution of jobs with restrictions on the timing of their implementation. Section 4.1 presents the problem of tropical optimization with restrictions, to which in section 4.2 the problem of optimal control will be reduced. Section 4.3 proposes a practical application of the result obtained, which consists in drawing up an optimal evacuation plan in case of emergency.

4.1 Solution of an optimization problem with constraints

Let matrices $\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ and vectors $\boldsymbol{g}, \boldsymbol{h}$ and $\boldsymbol{f}$ be given. Consider the following tropical optimization problem

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{x}^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} = \boldsymbol{y}, \\ & \boldsymbol{D}\boldsymbol{y} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}, \quad \boldsymbol{y} \le \boldsymbol{f}. \end{aligned} \tag{4.1}$$

Let us formulate and prove the following theorem

Theorem 10 ([41]). *Let $\boldsymbol{B}$ and $\boldsymbol{D}$ be matrices, $\boldsymbol{C}$ be a column-regular matrix such that the matrix $\boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{D}\boldsymbol{C}$ satisfies $\mathrm{Tr}(\boldsymbol{R}) \le \mathbb{1}$. Let $\boldsymbol{g}$ be a vector, and $\boldsymbol{f}$ and $\boldsymbol{h}$ be regular vectors such that the vector $\boldsymbol{s}^T = \boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-}$ satisfies $\boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{g} \le \mathbb{1}$.*

Then the minimum value of the objective function in problem (4.1) *is equal to*

$$\theta = (\boldsymbol{q}^{-}\boldsymbol{R}^*\boldsymbol{p})^{1/2} \oplus \boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{R}^*\boldsymbol{g}, \tag{4.2}$$

and all regular solutions have the form

$$\boldsymbol{x} = \boldsymbol{R}^*\boldsymbol{u}, \quad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{R}^*\boldsymbol{u}, \tag{4.3}$$

where $\boldsymbol{u}$ is any regular vector that satisfies the conditions

$$\boldsymbol{g} \oplus \theta^{-1}\boldsymbol{p} \le \boldsymbol{u} \le ((\boldsymbol{s}^T \oplus \theta^{-1}\boldsymbol{q}^{-})\boldsymbol{R}^*)^{-}. \tag{4.4}$$

Proof. Substituting $\boldsymbol{y} = \boldsymbol{Cx}$, into inequality $\boldsymbol{Dy} \leq \boldsymbol{x}$ and $\boldsymbol{y} \leq \boldsymbol{f}$ we will get $\boldsymbol{DCx} \leq \boldsymbol{x}$ and $\boldsymbol{Cx} \leq \boldsymbol{f}$. Combining inequalities $\boldsymbol{Bx} \leq \boldsymbol{x}$ and $\boldsymbol{DCx} \leq \boldsymbol{x}$ in one $\boldsymbol{Rx} \leq \boldsymbol{x}$, where $\boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC}$, Let us write the problem as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{x}, \\ & \boldsymbol{Rx} \leq \boldsymbol{x}, \quad \boldsymbol{Cx} \leq \boldsymbol{f}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{4.5}$$

Applying theorem 1, we obtain that $\boldsymbol{x} \leq (\boldsymbol{f}^{-}\boldsymbol{C})^{-}$. Due to the antitonicity of the inversion, the inequalities that limit the vector $\boldsymbol{x}$ from above can be rewritten like this: $\boldsymbol{x}^{-} \geq \boldsymbol{f}^{-}\boldsymbol{C}$ and $\boldsymbol{x}^{-} \geq \boldsymbol{h}^{-}$. Combining the obtained inequalities into one, we obtain $\boldsymbol{x}^{-} \geq \boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-}$ or $\boldsymbol{x} \leq (\boldsymbol{f}^{-}\boldsymbol{C} \oplus \boldsymbol{h}^{-})^{-} = (\boldsymbol{s}^{T})^{-}$.

Note that now the original problem can be written as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^{-}\boldsymbol{p} \oplus \boldsymbol{q}^{-}\boldsymbol{x}, \\ & \boldsymbol{Rx} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq (\boldsymbol{s}^{T})^{-}. \end{aligned} \tag{4.6}$$

The new problem has the form of problem 2.34, where the roles of the matrix $\boldsymbol{B}$ and the vector $\boldsymbol{h}$ are taken by $\boldsymbol{R}$ and $(\boldsymbol{s}^{T})^{-}$ respectively.

Let us estimate the computational complexity of finding a solution to the problem. The most labor intensive operation is calculation of the Kleene matrix, which will require no more than $O(n^4)$ scalar operations. The calculation of the remaining elements consists in a fixed number of operations on matrices and vectors and does not exceed $O(n^3)$. Thus, the computational complexity of finding a solution to the problem does not exceed $O(n^4)$ of scalar operations.

4.2 Minimizing the deviation from the due dates

Let us consider the problem of drawing up an optimal schedule for the execution of jobs in accordance with the criterion of the maximum departure of the due dates for the implementation of the project beyond specified time interval (1.8) with restrictions «start-start», «start-finish», «finish-start», «release time», «release deadline», and «completion deadline». We write the minimization problem with

constraints in the following form

$$\begin{aligned}\min_{x_i,y_i}\quad & \max\left(\max_{1\le i\le n}(p_i-x_i),\max_{1\le i\le n}(x_i-q_i)\right)\\ & \max_{1\le j\le n}(b_{ij}+x_j)\le x_i,\quad \max_{1\le j\le n}(c_{ij}+x_j)=y_i,\\ & \max_{1\le j\le n}(d_{ij}+y_j)\le x_i,\quad g_i\le x_i\le h_i,\quad y_i\le f_i,\quad i=1,\dots,n.\end{aligned}\tag{4.7}$$

Note that if the values $p_i = q_i$ are equal, the time intervals become narrower up to scalar values of the directive dates for the start of jobs, and the optimality criterion can be represented as follows

$$\max\left(\max_{1\le i\le n}(p_i-x_i),\max_{1\le i\le n}(x_i-p_i)\right)\tag{4.8}$$

Let us formulate the problem of minimizing the maximum deviation from the due dates:

$$\begin{aligned}\min_{x_i,y_i}\quad & \max\left(\max_{1\le i\le n}(p_i-x_i),\max_{1\le i\le n}(x_i-p_i)\right)\\ & \max_{1\le j\le n}(b_{ij}+x_j)\le x_i,\quad \max_{1\le j\le n}(c_{ij}+x_j)=y_i,\\ & \max_{1\le j\le n}(d_{ij}+y_j)\le x_i,\quad g_i\le x_i\le h_i,\quad y_i\le f_i,\quad i=1,\dots,n.\end{aligned}\tag{4.9}$$

Let us represent problem (4.7) in terms of the semifield $\mathbb{R}_{\max,+}$. To do this, we define the following matrix-vector notation:

$$\boldsymbol{B}=(b_{ij}),\quad \boldsymbol{C}=(c_{ij}),\quad \boldsymbol{D}=(d_{ij}),\tag{4.10}$$

$$\boldsymbol{x}=(x_i),\quad \boldsymbol{f}=(f_i),\quad \boldsymbol{g}=(g_i),\quad \boldsymbol{h}=(h_i),\quad \boldsymbol{p}=(p_i),\quad \boldsymbol{q}=(q_i).\tag{4.11}$$

In terms of the semifield $\mathbb{R}_{\max,+}$, the maximum outcome over the lower end of the interval has the form $x_i^- p_i$, and the maximum outcome over the upper end of the interval is $q_i^- x_i$. The objective function can be written in the following form:

$$\bigoplus_{1\le i\le n} x_i^- p_i \oplus \bigoplus_{1\le i\le n} q_i^- x_i.\tag{4.12}$$

We formulate the problem in terms of the idempotent semifield $\mathbb{R}_{\max,+}$

$$\begin{aligned}\min_{x_i,y_i} \quad & \bigoplus_{1\le i\le n} x_i^- p_i \oplus \bigoplus_{1\le i\le n} q_i^- x_i, \\ & \bigoplus_{1\le j\le n} b_{ij}x_j \le x_i, \quad \bigoplus_{1\le j\le n} c_{ij}x_j = y_i, \\ & \bigoplus_{1\le j\le n} d_{ij}y_j \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1,\dots,n\end{aligned} \tag{4.13}$$

and rewrite the problem in matrix-vector form

$$\begin{aligned}\min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{x}^-\boldsymbol{p} \oplus \boldsymbol{q}^-\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} = \boldsymbol{y}, \\ & \boldsymbol{D}\boldsymbol{y} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}, \quad \boldsymbol{y} \le \boldsymbol{f}.\end{aligned} \tag{4.14}$$

Thus, the problem under consideration takes the form 4.1 and can be solved using the theorem 10, and problem (4.9) reduces to problem (4.7) and is also can be solved using the theorem 10, in which $\boldsymbol{q} = \boldsymbol{p}$.

4.3 The problem of organizing an evacuation

Let us consider the process of organizing the evacuation of the population in the event of an emergency (for example, an accident at a nuclear power plant). We will assume that there is a given number of settlements that need to be evacuated. Starting the evacuation of a settlement too early can cause problems associated with congestion of the road network (which may be required, for example, for emergency forces organizing the aftermath) or with the fact that local authorities do not have time to collect all the citizens to be evacuated. Hence each point has a desired evacuation start time. It is required to organize evacuation with minimal deviation from given deadlines. Let us write auxiliary notation.

x_i – start time of the evacuation of the settlement i;

y_i – evacuation completion time i;

b_{ij} – minimum time interval between the start of the evacuation of the point $j = 1, \ldots, n$ and the start of the evacuation of i (due to a possible shortage of personnel and transport to start the evacuation at the same time) ;

c_{ij} – the minimum period of time between the start of the evacuation of point $j = 1, \ldots, n$ and the completion of the evacuation of point i (may be due to evacuation sequence. For example, small settlements can be evacuated by one transport);

d_{ij} – minimum period of time between completion of evacuation $j = 1, \ldots, n$ and start of evacuation i (can be the distance from the evacuation point to the settlement, the need for sanitization of transport and personnel rest);

g_i – the earliest possible start time for the evacuation i.

h_i – the latest allowed start time for the evacuation i.

f_i – latest evacuation of i completion time allowed.

Presented problem is a special case of problem (4.7), which can be represented in terms of the semifield $\mathbb{R}_{\max,+}$ as (4.9). Let us consider the problem of evacuating $n = 3$ settlements using folowing matrices and vectors

$$\boldsymbol{B} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -2 & -3 & 0 \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} -4 & -5 & -6 \\ \mathbb{0} & -4 & -7 \\ -5 & \mathbb{0} & -4 \end{pmatrix}, \tag{4.15}$$

$$\boldsymbol{g} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} 6 \\ 6 \\ 6 \end{pmatrix}, \quad \boldsymbol{p} = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}. \tag{4.16}$$

Let us find the matrices

$$\boldsymbol{DC} = \begin{pmatrix} 0 & -2 & -1 \\ -1 & -3 & -2 \\ -1 & -3 & -1 \end{pmatrix}, \quad \boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -1 & -3 & 0 \end{pmatrix}, \tag{4.17}$$

and then calculate the powers

$$\boldsymbol{R}^2 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & -1 & 2 \\ -1 & -2 & 0 \end{pmatrix}, \quad \boldsymbol{R}^3 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix}. \tag{4.18}$$

Taking into account that holds the condition $\mathrm{Tr}(\boldsymbol{R}) = \mathrm{tr}\,\boldsymbol{R} \oplus \mathrm{tr}\,\boldsymbol{R}^2 \oplus \mathrm{tr}\,\boldsymbol{R}^3 = 0 \le \mathbb{1}$, sequentially calculate

$$\boldsymbol{R}^* = \boldsymbol{I} \oplus \boldsymbol{R} \oplus \boldsymbol{R}^2 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix}, \qquad \boldsymbol{R}^*\boldsymbol{g} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \tag{4.19}$$

$$\boldsymbol{f}^-\boldsymbol{C} = \begin{pmatrix} -2 & -4 & -3 \end{pmatrix}, \qquad \boldsymbol{f}^-\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{g} = -1, \qquad \boldsymbol{h}^-\boldsymbol{B}^*\boldsymbol{g} = -1. \tag{4.20}$$

Make sure the condition is true

$$\boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{g} = (\boldsymbol{f}^-\boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{R}^*\boldsymbol{g} = -1 < \mathbb{1}. \tag{4.21}$$

Let us find scalars

$$\boldsymbol{p}^-\boldsymbol{R}^*\boldsymbol{p} = 1, \quad \boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{p} = 1, \quad \boldsymbol{p}^-\boldsymbol{R}^*\boldsymbol{g} = 0. \tag{4.22}$$

Substituting the found values, we calculate the minimum of the objective function

$$\theta = (\boldsymbol{p}^-\boldsymbol{R}^*\boldsymbol{p})^{1/2} \oplus \boldsymbol{s}^T\boldsymbol{R}^*\boldsymbol{p} \oplus \boldsymbol{p}^-\boldsymbol{R}^*\boldsymbol{g} = 1. \tag{4.23}$$

Let us find the vectors:

$$\boldsymbol{s}^T \oplus \theta^{-1}\boldsymbol{p}^- = \begin{pmatrix} -2 & -3 & -2 \end{pmatrix},$$

$$\boldsymbol{g} \oplus \theta^{-1}\boldsymbol{p} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad ((\boldsymbol{s}^T \oplus \theta^{-1}\boldsymbol{p}^-)\boldsymbol{R}^*)^- = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \tag{4.24}$$

Let us represent all solutions $\boldsymbol{x} = (x_1, x_2, x_3)$ and $\boldsymbol{y} = (y_1, y_2, y_3)$ using the parameter vector $\boldsymbol{u} = (u_1, u_2, u_3)$ as

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix} \boldsymbol{u}, \quad \boldsymbol{y} = \begin{pmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix} \boldsymbol{x}, \quad \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \le \boldsymbol{u} \le \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \tag{4.25}$$

Since all columns of the matrix $\boldsymbol{R}^*$ are collinear, it can be written as follows:

$$\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 1 \end{pmatrix}. \tag{4.26}$$

Compute the value of the scalar $u = u_1 \oplus (-1)u_2 \oplus (1)u_3$. We represent the solution by replacing the parameter vector $\boldsymbol{u}$ with the introduced scalar and calculating the bounds for u

$$\boldsymbol{x} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} u, \quad \boldsymbol{y} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix} u, \quad 1 \leq u \leq 2. \tag{4.27}$$

Let us represent the solution in terms of ordinary arithmetic operations

$$x_1 = u,\ x_2 = u + 1,\ x_3 = u - 1,\ y_1 = u + 4,\ y_2 = u + 3,\ y_3 = u + 2.\ 1 \leq u \leq 2 \tag{4.28}$$

Chapter 5

Minimizing the spread of the jobs completion times

In this chapter formulated and solved the problem of drawing up an optimal calendar schedule for the execution of project jobs, which consists in minimizing the spread in the completion time of the project jobs with given restrictions on the sequence of jobs execution. In section 5.1, the problem of tropical optimization is formulated and solved, which will be further used in section 5.2 for solving the optimal control problem under study.

5.1 Solution of an optimization problem with constraints

Consider the following problem. Let the matrices $\boldsymbol{A}, \boldsymbol{C} \in \mathbb{X}^{m\times n}, \quad \boldsymbol{B} \in \mathbb{X}^{n\times n}$, vectors $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{f} \in \mathbb{X}^m$ and the vector $\boldsymbol{g} \in \mathbb{X}^n$ are given. Let us write down the problem of minimizing the objective function under given constraints with respect to the unknown $\boldsymbol{x} \in \mathbb{X}^n$

$$\begin{aligned} \min \quad & \boldsymbol{q}^- \boldsymbol{x} (\boldsymbol{A}\boldsymbol{x})^- \boldsymbol{p}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \le \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{f}, \quad \boldsymbol{x} \le \boldsymbol{h}. \end{aligned} \tag{5.1}$$

which is a further complication of problem (2.25), obtained by adding new constraint inequalities. Let us formulate and prove the following theorem:

Theorem 11 ([38])**.** *Let a row-regular matrix be given* $\boldsymbol{A}$*, matrix regular by columns* $\boldsymbol{C}$ *and matrix* $\boldsymbol{B}$ *such as* $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$*. Suppose that vector* $\boldsymbol{p}$ *is nonzero, vectors* $\boldsymbol{q}$*,* $\boldsymbol{f}$ *and* $\boldsymbol{h}$ *are regular, and satisfy the condition* $\boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{g} \le ((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{B}^* (\boldsymbol{q}^- \boldsymbol{B}^*)^-)^{-1}$.

Then the minimum in problem (5.1) *is equal to* $\Delta = (\boldsymbol{A}\boldsymbol{B}^*(\boldsymbol{q}^- \boldsymbol{B}^*)^-)^- \boldsymbol{p}$ *and is achieved at*

$$\boldsymbol{x} = \alpha \boldsymbol{B}^* \left(\boldsymbol{q}^- \boldsymbol{B}^*\right)^-, \qquad \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{g} \le \alpha \le ((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{B}^* (\boldsymbol{q}^- \boldsymbol{B}^*)^-)^{-1}. \tag{5.2}$$

Proof. Let us note that to solve the first inequality constraint $\boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \le \boldsymbol{x}$ we can use the theorem 3. Let us write instead of the inequality its decision

$\boldsymbol{x} = \boldsymbol{B}^*\boldsymbol{u}$, where $\boldsymbol{u}$ – is a regular vector such as $\boldsymbol{u} \geq \boldsymbol{g}$. After substitution, we get a new optimization problem with respect to the unknown vector $\boldsymbol{u}$:

$$\begin{aligned} \min \quad & \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{u} (\boldsymbol{A}\boldsymbol{B}^* \boldsymbol{u})^- \boldsymbol{p}, \\ & \boldsymbol{B}^* \boldsymbol{u} \leq \boldsymbol{h}, \quad \boldsymbol{C}\boldsymbol{B}^* \boldsymbol{u} \leq \boldsymbol{f}, \quad \boldsymbol{u} \geq \boldsymbol{g}. \end{aligned} \tag{5.3}$$

Since the vectors $\boldsymbol{f}$ and $\boldsymbol{h}$ are regular, and the matrix $\boldsymbol{C}$ is regular by columns, $\boldsymbol{B}^* \geq \boldsymbol{I}$, matrices $\boldsymbol{B}^*$ and $\boldsymbol{C}\boldsymbol{B}^*$ are also regular by columns. Thus, for the inequalities $\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{u} \leq \boldsymbol{f}$ and $\boldsymbol{B}^*\boldsymbol{u} \leq \boldsymbol{h}$ the conditions of the theorem 1 are true. After solving the inequality, we obtain the problem with respect to the vector $\boldsymbol{u}$, written as follows

$$\begin{aligned} \min \quad & \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{u} (\boldsymbol{A}\boldsymbol{B}^* \boldsymbol{u})^- \boldsymbol{p}, \\ & \boldsymbol{g} \leq \boldsymbol{u} \leq (\boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^*)^-, \quad \boldsymbol{u} \leq (\boldsymbol{h}^- \boldsymbol{B}^*)^-. \end{aligned} \tag{5.4}$$

Let us study this problem without restrictions. Since the matrix $\boldsymbol{A}$ is row regular, and the vector $\boldsymbol{q}$ is regular, the matrix $\boldsymbol{A}\boldsymbol{B}^*$ is also regular by rows, and the vector $(\boldsymbol{q}^- \boldsymbol{B}^*)^-$ is regular. Using the theorem 4, we will find that in the absence of restrictions, the minimum in the problem is equal to $\Delta = (\boldsymbol{A}\boldsymbol{B}^*(\boldsymbol{q}^- \boldsymbol{B}^*)^-)^- \boldsymbol{p}$ and is reached on any vector

$$\boldsymbol{u} = \alpha \left(\boldsymbol{q}^- \boldsymbol{B}^*\right)^-, \qquad \alpha > \mathbb{0}. \tag{5.5}$$

Let us determine which of the found solutions satisfy the existing constraints. Note that the restrictions $\boldsymbol{u} \leq (\boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^*)^-$ и $\boldsymbol{u} \leq (\boldsymbol{h}^- \boldsymbol{B}^*)^-$ can be represented using the inequalities $\boldsymbol{u}^- \geq \boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^*$ и $\boldsymbol{u}^- \geq \boldsymbol{h}^- \boldsymbol{B}^*$, or by one equivalent inequality $\boldsymbol{u}^- \geq \boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^* \oplus \boldsymbol{h}^- \boldsymbol{B}^*$. The resulting inequality is equivalent to the condition $\boldsymbol{u} \leq (\boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^* \oplus \boldsymbol{h}^- \boldsymbol{B}^*)^-$. To find the values of α for which the constraints of the problem are satisfied, we need to solve the following double inequality

$$\boldsymbol{g} \leq \alpha (\boldsymbol{q}^- \boldsymbol{B}^*)^- \leq (\boldsymbol{f}^- \boldsymbol{C}\boldsymbol{B}^* \oplus \boldsymbol{h}^- \boldsymbol{B}^*)^-. \tag{5.6}$$

After solving the left and right inequalities, we obtain

$$\boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{g} \leq \alpha \leq ((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-) \boldsymbol{B}^* (\boldsymbol{q}^- \boldsymbol{B}^*)^-)^{-1}. \tag{5.7}$$

By the conditions of the theorem, the set of α values that satisfy the inequality is not empty. After returning to the original problem with respect to the vector $\boldsymbol{x}$, we obtain the desired result.

Let us estimate the computational complexity of the found solution. In the problem under study, the most complex operation is the calculation of the Kleene matrix, which consists of computing the first $n-1$ powers of the matrix bmB and requires at most $O(n^4)$ scalar operations. The calculation of the other elements of the solution is in algebraic operations on matrices and vectors with complexity not exceeding $O(n^3)$. Therefore, the search for a solution to the problem requires no more than than $O(n^4)$ scalar operations.

5.2 Minimizing spread in completion times

This section presents a solution to the network planning problem, which consists in minimizing the criterion for the maximum spread in the execution time of all project jobs (1.9) with the addition of restrictions on their execution time «start-start», «start-finish», «release time», «release deadline» and «completion deadline»

$$\begin{aligned} \min_{x_i,y_i} \quad & \max_{1\le i\le n} y_i + \max_{1\le i\le n} (-y_i)\,, \\ & \max(\max_{1\le j\le n} (b_{ij}+x_j)\,, g_i) \le x_i, \quad \max_{1\le j\le n} (c_{ij}+x_j) = y_i, \\ & y_i \le f_i \quad x_i \le h_i, \quad i=1,\dots,n. \end{aligned} \tag{5.8}$$

Note that when writing problem (5.8) in the usual notation, we use only the operations of addition, subtraction and determination of the maximum, and, therefore, the problem can be easily represented in a tropical form in terms of the semifield $\mathbb{R}_{\max,+}$. The constraint equations and inequalities in terms of the semifield $\mathbb{R}_{\max,+}$ can be written as follows:

$$\bigoplus_{j=1}^{n} b_{ij}x_j \oplus g_i \le x_i, \quad \bigoplus_{j=1}^{n} c_{ij}x_j = y_i, \quad x_i \le h_i, \quad y_i \le f_i \quad i=1,\dots,n. \tag{5.9}$$

We define the following matrix-vector notation

$$\boldsymbol{B}=(b_{ij}), \qquad \boldsymbol{C}=(c_{ij}), \qquad \boldsymbol{y}=(y_i), \qquad \boldsymbol{g}=(g_i), \qquad \boldsymbol{h}=(h_i), \qquad \boldsymbol{f}=(f_i) \tag{5.10}$$

and represent the constraints in matrix-vector form

$$\boldsymbol{Bx} \oplus \boldsymbol{g} \leq \boldsymbol{x}, \quad \boldsymbol{Cx} = \boldsymbol{y}, \quad \boldsymbol{x} \leq \boldsymbol{h}, \quad \boldsymbol{y} \leq \boldsymbol{f}. \tag{5.11}$$

We write the objective function of the problem in tropical form

$$\bigoplus_{i=1}^{n} y_i \bigoplus_{j=1}^{n} y_j = \mathbf{1}^T \boldsymbol{y} \boldsymbol{y}^{-} \mathbf{1}. \tag{5.12}$$

Thus, the whole problem (5.8) can be written as follows:

$$\begin{aligned} \min \ & \mathbf{1}^T \boldsymbol{y} \boldsymbol{y}^{-} \mathbf{1}, \\ & \boldsymbol{Bx} \oplus \boldsymbol{g} \leq \boldsymbol{x}, \quad \boldsymbol{Cx} = \boldsymbol{y}, \\ & \boldsymbol{x} \leq \boldsymbol{h}, \quad \boldsymbol{y} \leq \boldsymbol{f}. \end{aligned} \tag{5.13}$$

Theorem 12 ([38])**.** *Let the regular matrix is given* $\boldsymbol{C}$ *and matrix* $\boldsymbol{B}$ *such as* $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$*. We will assume that* $\boldsymbol{f}$ *u* $\boldsymbol{h}$ *are regular and satisfy the following condition*

$$\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* \boldsymbol{g} \leq \left((\boldsymbol{f}^{-} \boldsymbol{C} \oplus \boldsymbol{h}^{-}) \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^{-} \right)^{-1}. \tag{5.14}$$

Then the minimum in problem (5.13) *is equal to*

$$\Delta = (\boldsymbol{C} \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^{-})^{-} \mathbf{1} \tag{5.15}$$

and is achieved at

$$\boldsymbol{x} = \alpha \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^{-}, \qquad \boldsymbol{y} = \alpha \boldsymbol{C} \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^{-}, \tag{5.16}$$

where

$$\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* \boldsymbol{g} \leq \alpha \leq \left((\boldsymbol{f}^{-} \boldsymbol{C} \oplus \boldsymbol{h}^{-}) \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^{-} \right)^{-1}. \tag{5.17}$$

Proof. After substituting the equality $\boldsymbol{y} = \boldsymbol{Cx}$ into the objective function and constraints, the problem takes the following form

$$\begin{aligned} \min \ & \mathbf{1}^T \boldsymbol{Cx} (\boldsymbol{Cx})^{-} \mathbf{1}, \\ & \boldsymbol{Bx} \oplus \boldsymbol{g} \leq \boldsymbol{x}, \\ & \boldsymbol{Cx} \leq \boldsymbol{f}, \\ & \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{5.18}$$

Note that the resulting problem has the form of a problem (5.1), in which the vector $\boldsymbol{q}^-$ is replaced by the unit vector $\mathbf{1}^T\boldsymbol{C}$ and $\boldsymbol{C}$ by $\boldsymbol{C}$. Note that since the matrix $\boldsymbol{C}$ is regular, then the vector $\mathbf{1}^T\boldsymbol{C}$ is also regular. Moreover, since $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$, while the vectors $\boldsymbol{f}$ and $\boldsymbol{h}$ are regular and satisfy the constraint $\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{g} \leq \big((\boldsymbol{f}^-\boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{B}^*(\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*\big)^-)^{-1}$, the theorem conditions 11 are hold. Using the theorem, we find the minimum in the problem $\Delta = (\boldsymbol{C}\boldsymbol{B}^*(\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*)^-)^-\mathbf{1}$, which is achieved with $\boldsymbol{x} = \alpha\boldsymbol{B}^*(\mathbf{1}^T\boldsymbol{A}\boldsymbol{B}^*)^-$, where the parameter α is given by the condition $\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{g} \leq \alpha \leq \big((\boldsymbol{f}^-\boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{B}^*(\mathbf{1}^T\boldsymbol{C}\boldsymbol{B}^*\big)^-)^{-1}$. Taking into account that $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, we obtain the desired result.

5.3 Optimization of clinical examination

Regular clinical examinations allow monitoring the health status of citizens and detect a large number of diseases at an early stage, which ultimately allows to increase the life expectancy of the population and reduce the burden on the health-care system in the long term. Nevertheless, regular quality medical examinations by themselves are quite costly and create a significant burden on the healthcare system. These facts make the problem of optimizing medical examinations important and relevant. The medical examination itself consists in carrying out a certain number of medical procedures. Let us formulate the task of passing the prophylactic clinical examination in more detail.

We assume that it is necessary to minimize the spread the time of completion of the procedures, which may be required, for example, when performing an on-site inspection or for the subsequent performance of the necessary jobs associated with a general suspension of activities (for example, during disinfection). Let us introduce the following auxiliary notation:

x_i – start time of the procedure i (testing, examination by a specialist, registration);

y_i – completion time of the procedure i;

b_{ij} – the minimum period of time between the start of $j = 1, \ldots, n$ and the start of i. As a practical example of such a restriction, one can consider, for example, a ban on too frequent blood sampling;

c_{ij} – the minimum time interval between the beginning of the procedure $j = 1, \ldots, n$ and the completion of i. It may be due, for example, to the fact that a subsequent consultation with a doctor makes sense only after receiving the results of the tests;

g_i – the earliest possible start time for the procedure i;

h_i – latest possible procedure i start time;

f_i – the latest allowed procedure completion time of the procedure i.

Note that the presented problem is a special case of the above network planning problem (5.8) and it can be written in the form (5.13).

For a more visual description of the result obtained, consider the conditional medical examination, which consists in the implementation of $n = 3$ procedures with restrictions on the time of their execution, specified in terms of the semifield $\mathbb{R}_{\max,+}$ using the following matrices:

$$\boldsymbol{B} = \begin{pmatrix} \mathbb{0} & -2 & 1 \\ 0 & \mathbb{0} & 2 \\ -1 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \qquad \boldsymbol{C} = \begin{pmatrix} 4 & 0 & \mathbb{0} \\ 2 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}, \tag{5.19}$$

где $\mathbb{0} = -\infty$, and vectors

$$\boldsymbol{g} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \qquad \boldsymbol{h} = \begin{pmatrix} 5 \\ 4 \\ 4 \end{pmatrix}, \qquad \boldsymbol{f} = \begin{pmatrix} 8 \\ 8 \\ 10 \end{pmatrix}. \tag{5.20}$$

Let us make sure that the conditions of the lemma 12 are satisfied. It is easy to see that the matrix $\boldsymbol{C}$ is regular. Let us check that $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$. To do this, we need to calculate the following matrices:

$$\boldsymbol{B}^2 = \begin{pmatrix} 0 & \mathbb{0} & 0 \\ 1 & -2 & 1 \\ \mathbb{0} & -3 & 0 \end{pmatrix}, \qquad \boldsymbol{B}^3 = \begin{pmatrix} -1 & -2 & 1 \\ 0 & -1 & 2 \\ -1 & \mathbb{0} & -1 \end{pmatrix}. \tag{5.21}$$

After calculating the traces, Let us find the values $\mathrm{Tr}(\boldsymbol{B}) = \mathrm{tr}\,\boldsymbol{B} \oplus \mathrm{tr}\,\boldsymbol{B}^2 \oplus \mathrm{tr}\,\boldsymbol{B}^3 = 0 = \mathbb{1}$. Thus, the Kleene matrix is defined by

$$\boldsymbol{B}^* = \boldsymbol{I} \oplus \boldsymbol{B} \oplus \boldsymbol{B}^2 = \begin{pmatrix} 0 & -2 & 1 \\ 1 & 0 & 2 \\ -1 & -3 & 0 \end{pmatrix}. \tag{5.22}$$

Let us verify that the inequality is true

$$\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* \boldsymbol{g} \leq \left((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-) \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^- \right)^{-1}. \tag{5.23}$$

First we will find

$$\begin{gathered} \boldsymbol{C}\boldsymbol{B}^* = \begin{pmatrix} 4 & 2 & 5 \\ 4 & 3 & 5 \\ 2 & 1 & 3 \end{pmatrix}, \qquad \mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* = \begin{pmatrix} 4 & 3 & 5 \end{pmatrix}, \\ \mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* \boldsymbol{g} = 5, \qquad \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^- = \begin{pmatrix} -4 \\ -3 \\ -5 \end{pmatrix}. \end{gathered} \tag{5.24}$$

Let us compute auxiliary vectors

$$\boldsymbol{f}^- \boldsymbol{C} = \begin{pmatrix} -4 & -5 & -7 \end{pmatrix}, \qquad \boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^- = \begin{pmatrix} -4 & -4 & -4 \end{pmatrix}. \tag{5.25}$$

Using the obtained vectors, we will find

$$((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-) \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^-)^{-1} = 7, \tag{5.26}$$

hence holds the condition $\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^* \boldsymbol{g} \leq \left((\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-) \boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^- \right)^{-1}$.

Using the theorem 12, we obtain the minimum of the problem Δ

$$\boldsymbol{C}\boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^- = \begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix}, \qquad \Delta = (\boldsymbol{C}\boldsymbol{B}^* (\mathbf{1}^T \boldsymbol{C} \boldsymbol{B}^*)^-)^- \mathbf{1} = 2, \tag{5.27}$$

which is achieved with

$$\boldsymbol{x} = \alpha \boldsymbol{B}^*(\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^*)^- = \alpha \begin{pmatrix} -4 \\ -3 \\ -5 \end{pmatrix},\ \boldsymbol{y} = \alpha \boldsymbol{C}\boldsymbol{B}^*(\mathbf{1}^T \boldsymbol{C}\boldsymbol{B}^*)^- =$$

$$= \alpha \begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix},\ 5 \leq \alpha \leq 7. \quad (5.28)$$

Note that with the value $\alpha = 5$ we have a plan with the earliest dates for the start of procedures, which is given by the vectors

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} 5 \\ 5 \\ 3 \end{pmatrix}, \quad (5.29)$$

and with the value $\alpha = 7$ we have a plan with the latest dates for the start of procedures, for which

$$\boldsymbol{x} = \begin{pmatrix} 3 \\ 4 \\ 2 \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} 7 \\ 7 \\ 5 \end{pmatrix}. \quad (5.30)$$

Chapter 6

Maximizing the spread of the jobs completion times

Let us consider the problem of drawing up an optimal schedule for the execution of project jobs, which consists in maximizing the spread in the completion time of the project job under given constraints. Section 6.1 presents a solution to the tropical optimization problem, which will be further used in section 6.2 to solve the network planning problem. Section 6.3 proposes a practical application of the result obtained to optimize the work of the medical center, conducting medical procedures associated with harmful effects on the organism.

6.1 Solving a tropical optimization problem with constraints

Let matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{X}^{n\times n}$ and vectors $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{g}, \boldsymbol{h} \in \mathbb{X}^n$. Consider the following optimization problem

$$\begin{aligned} \max_{\boldsymbol{x}} \ & \boldsymbol{q}^- \boldsymbol{x} (\boldsymbol{A}\boldsymbol{x})^- \boldsymbol{p}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{h}. \end{aligned} \tag{6.1}$$

Let us present a theorem that proposes a solution to the problem under study.

Theorem 13 ([39]). *Suppose that the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are such that $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$, matrix $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^*$ has regular columns $\boldsymbol{d}_j = (d_{ij})$, where $\boldsymbol{B}^* = (\boldsymbol{b}_j^*)$ is a Kleene matrix, and the matrix $\boldsymbol{C}$ is column regular. Let the vectors $\boldsymbol{p} = (p_j)$, $\boldsymbol{q}$ and $\boldsymbol{h}$ be regular, and the vector $\boldsymbol{r} = (r_j)$ be defined as $\boldsymbol{r} = (\boldsymbol{h}^- \boldsymbol{C}\boldsymbol{B}^*)^-$.*

Then the maximum in the (6.1) *problem is equal to*

$$\Delta = \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{D}^- \boldsymbol{p} \tag{6.2}$$

and is reached when $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$, where $\boldsymbol{u} = (u_j)$ is any vector with components

$$u_k = \alpha d_{sk}^{-1} p_s, \quad u_j \le \alpha d_{sj}^{-1} p_s, \quad j \ne k, \tag{6.3}$$

under the condition that

$$\alpha \le \min_{1\le j\le n} d_{sj} p_s^{-1} r_j, \quad k = \arg\max_{1\le j\le n} \boldsymbol{q}^- \boldsymbol{b}_j^* \boldsymbol{d}_j^- \boldsymbol{p}, \quad s = \arg\max_{1\le j\le n} d_{jk}^{-1} p_j. \tag{6.4}$$

Proof. Replacing the inequality $\boldsymbol{Bx} \leq \boldsymbol{x}$ to its solution $\boldsymbol{x} = \boldsymbol{B}^*\boldsymbol{u}$, where $\boldsymbol{u}$ is any regular vector of suitable size we obtain a new problem for the vector $\boldsymbol{u}$ in the following form

$$\begin{gathered} \max \ \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{u} \left(\boldsymbol{D}\boldsymbol{u}\right)^- \boldsymbol{p}, \\ \boldsymbol{C}\boldsymbol{B}^*\boldsymbol{u} \leq \boldsymbol{h}. \end{gathered} \tag{6.5}$$

Using theorem 7 to calculate the maximum in a problem without limits we get the maximum equal to $\Delta = \boldsymbol{q}^- \boldsymbol{B}^* \boldsymbol{D}^- \boldsymbol{p}$, which is achieved if and only if the vector $\boldsymbol{u}$ has components determined by the conditions (8.5)-(8.6).

Let us find out which of the found solutions satisfy the inequality $\boldsymbol{C}\boldsymbol{B}^*\boldsymbol{u} \leq \boldsymbol{h}$. Since the matrix $\boldsymbol{C}$ is regular in columns and $\boldsymbol{B}^* \geq \boldsymbol{I}$, the matrix $\boldsymbol{C}\boldsymbol{B}^*$ is column regular. Therefore, the considered inequality satisfies the condition of the theorem 1 and can be solved as follows $\boldsymbol{u} \leq (\boldsymbol{h}^- \boldsymbol{C}\boldsymbol{B}^*)^- = \boldsymbol{r}$.

Let us find for what values of α any vector $\boldsymbol{u}$, which is given by the conditions (8.5), satisfies the inequality $\boldsymbol{u} \leq \boldsymbol{r}$. For this it is sufficient that the inequalities $\alpha d_{sj}^{-1} p_s \leq r_j$ hold for all $j = 1, \ldots, n$. After solving these inequalities, we obtain for all j the inequalities $\alpha \leq d_{sj} p_s^{-1} r_j$, which can be replaced by the equivalent inequality

$$\alpha \leq \min_{1 \leq j \leq n} d_{sj} p_s^{-1} r_j. \tag{6.6}$$

The parameter α that satisfies this condition specifies the vectors $\boldsymbol{u}$, with which all vectors $\boldsymbol{x} = \boldsymbol{B}^*\boldsymbol{u}$, are solutions to the problem (6.1).

The main computational complexity in the problem is the search for the Kleene matrix, which requires no more than $O(n^4)$ scalar operations. The calculation of the remaining elements consists in a fixed number of operations on matrices and vectors, as well as in the search for indices, and its computational complexity does not exceed $O(n^3)$. Therefore, finding a solution will require no more than $O(n^4)$ scalar operations.

6.2 Maximizing completion time spread

Let us represent the solution of the problem of maximizing the criterion of maximum project completion time scatter (1.9) with restrictions of the form «start-start»,

«start-finish» and «completion deadline»

$$\begin{aligned} \max_{x_i,y_i} \quad & \max_{1\le i\le n} y_i + \max_{1\le i\le n} (-y_i)\,, \\ & \max_{1\le j\le n} (a_{ij}+x_j) = y_i, \quad \max_{1\le j\le n} (b_{ij}+x_j) \le x_i, \\ & y_i \le h_i, \quad i = 1,\dots,n. \end{aligned} \tag{6.7}$$

Note that the (6.7) problem can be easily represented in terms of the $\mathbb{R}_{\max,+}$ semifield, since it includes only the operations of determining the maximum, addition, and subtraction. Let us represent the equations and constraint inequalities in terms of the semifield $\mathbb{R}_{\max,+}$

$$\bigoplus_{j=1}^{n} a_{ij}x_j \le x_i, \quad \bigoplus_{j=1}^{n} b_{ij}x_j = y_i, \quad y_i \le h_i, \quad i = 1,\dots,n. \tag{6.8}$$

Let us define the following notations

$$\boldsymbol{A} = (a_{ij}), \quad \boldsymbol{B} = (b_{ij}), \quad \boldsymbol{x} = (x_i), \quad \boldsymbol{y} = (y_i), \quad \boldsymbol{h} = (h_i). \tag{6.9}$$

We write the target function in vector form

$$\bigoplus_{i=1}^{n} y_i \bigoplus_{j=1}^{n} y_j = \mathbf{1}^T \boldsymbol{y}\boldsymbol{y}^{-}\mathbf{1}. \tag{6.10}$$

We represent the constraints of the problem in the form of vector equations and inequalities

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \quad \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{y} \le \boldsymbol{h}, \tag{6.11}$$

and write the problem in vector form

$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{y}} \quad & \mathbf{1}^T \boldsymbol{y}\boldsymbol{y}^{-}\mathbf{1} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \quad \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{y} \le \boldsymbol{h}. \end{aligned} \tag{6.12}$$

Let us present the solution of problem (6.12)

Theorem 14 ([39]). *Suppose that the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are such that the matrix $\boldsymbol{A}$ is regular,* $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$, *while the matrix $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^*$ has regular columns $\boldsymbol{d}_i = (d_{ij})$, where $\boldsymbol{B}^*$ is the Kleene matrix.*

Then the maximum in the (6.12) *problem is* $\Delta = \mathbf{1}^T \boldsymbol{D}\boldsymbol{D}^{-}\mathbf{1}$ *and is reached when* $\boldsymbol{x} = \boldsymbol{B}^{*}\boldsymbol{u}$ *and* $\boldsymbol{y} = \boldsymbol{D}\boldsymbol{u}$, *where* $\boldsymbol{u} = (u_j)$ – *any vector with components*

$$u_k = \alpha d_{sk}^{-1}, \quad u_j \le \alpha d_{sj}^{-1}, \quad j \ne k, \tag{6.13}$$

under the condition that

$$\alpha \le \min_{1\le j\le n} d_{sj}(\boldsymbol{h}^{-}\boldsymbol{d}_j)^{-}, \quad k = \arg\max_{1\le j\le n} \mathbf{1}^T \boldsymbol{d}_j \boldsymbol{d}_j^{-}\mathbf{1}, \quad s = \arg\max_{1\le j\le n} d_{jk}^{-1}. \tag{6.14}$$

Proof. Substituting the expression $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ into the objective function and the constraint we obtain the problem with respect to the unknown vector $\boldsymbol{x}$ in the following form

$$\begin{aligned} \max \ & \mathbf{1}^T \boldsymbol{A}\boldsymbol{x}(\boldsymbol{A}\boldsymbol{x})^{-}\mathbf{1}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \\ & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{h}. \end{aligned} \tag{6.15}$$

The resulting problem can be reduced to problem (6.1) under the condition that $\boldsymbol{C} = \boldsymbol{A}$, $\boldsymbol{p} = \mathbf{1}$ and $\boldsymbol{q} = (\mathbf{1}^T\boldsymbol{A})^{-}$. Let us check the fulfillment of the theorem conditions 13. The matrix $\boldsymbol{B}$ satisfies the existence condition for the Kleene matrix $\mathrm{Tr}(\boldsymbol{B}) \le \mathbb{1}$, all columns of the matrix $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^{*}$ are regular and the matrix $\boldsymbol{C}$ is regular. We introduce an auxiliary vector $\boldsymbol{r} = (\boldsymbol{h}^{-}\boldsymbol{A}\boldsymbol{B}^{*})^{-} = (\boldsymbol{h}^{-}\boldsymbol{D})^{-}$. Since the vectors $\boldsymbol{p}$ и $\boldsymbol{q}$ are regular, we conclude that these conditions are satisfied.

Using the theorem 13 we obtain that the maximum in the problem is equal to $\Delta = \mathbf{1}^T\boldsymbol{D}\boldsymbol{D}^{-}\mathbf{1}$ and is reached if and only if $\boldsymbol{x} = \boldsymbol{B}^{*}\boldsymbol{u}$, where $\boldsymbol{u}$ is any vector with components (6.13) that satisfy the conditions (6.14).

6.3 Optimization of harmful medical intervention

Unfortunately, many necessary medical procedures are harmful to the health of both medical personnel and the patient. Examples of such medical interventions include, for example, taking x-rays, computed tomography and magnetic resonance imaging, various types of surgery and many types of anesthesia. When scheduling such procedures, it is highly desirable to spread as much as possible them in time. Given the problem presented, it makes sense when scheduling to render to maximize the variation in the completion time of the procedures.

Let us introduce auxiliary notation.

x_i – start time of procedure i;

y_i – completion time of procedure i;

b_{ij} – the minimum period of time between the start of $j = 1, \dots, n$ and the start of i. Such limitations may arise during scheduling of operations or during repeated contrast CT;

c_{ij} – minimum time interval between the beginning of the procedure $j = 1, \dots, n$ and the completion of the procedure i. Such restrictions may be required if necessary to minimize the exposure of medical personnel to X-rays or radioactive radiation;

h_i – the latest possible time of the procedure i.

The presented task is a special case of the task (6.7), which can be represented in terms of the semifield $\mathbb{R}_{\max,+}$ as (6.12).

Let us describe the work of a medical center, which needs to optimize $n = 3$ harmful medical interventions. Restrictions on the reception time are given by the following matrices and vectors:

$$\boldsymbol{B} = \begin{pmatrix} \mathbb{0} & -2 & 1 \\ 0 & \mathbb{0} & 2 \\ -1 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 4 & 1 & 1 \\ 2 & 2 & 0 \\ 0 & 1 & 3 \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} 5 \\ 4 \\ 4 \end{pmatrix}. \tag{6.16}$$

Note that the same matrix $\boldsymbol{B}$ is used as in the previous example. thus we know that $\mathrm{Tr}(\boldsymbol{B}) = \mathbb{1}$. Compute matrices

$$\boldsymbol{B}^* = \begin{pmatrix} 0 & -2 & 1 \\ 1 & 0 & 2 \\ -1 & -3 & 0 \end{pmatrix}, \quad \boldsymbol{D} = \boldsymbol{C}\boldsymbol{B}^* = \begin{pmatrix} 4 & 2 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix}. \tag{6.17}$$

Then we calculate the matrices

$$\boldsymbol{D}^- = \begin{pmatrix} -4 & -3 & -2 \\ -2 & -2 & -1 \\ -5 & -4 & -3 \end{pmatrix}, \quad \boldsymbol{D}\boldsymbol{D}^- = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}, \tag{6.18}$$

and find the maximum in the problem equal to $\Delta = \mathbf{1}^T \boldsymbol{D}\boldsymbol{D}^{-}\mathbf{1} = 2$.

Let us determine the value of the indices k and s and the lower bound for the parameter α, which are given by the conditions (6.14). First Let us find

$$\mathbf{1}^T \boldsymbol{d}_1 \boldsymbol{d}_1^{-}\mathbf{1} = 2, \quad \mathbf{1}^T \boldsymbol{d}_2 \boldsymbol{d}_2^{-}\mathbf{1} = 1, \quad \mathbf{1}^T \boldsymbol{d}_3 \boldsymbol{d}_3^{-}\mathbf{1} = 2. \tag{6.19}$$

Therefore, the index k can take two values: 1 and 3.

Consider the value $k = 1$. Taking into account that

$$d_{11}^{-1} = -4, \quad d_{21}^{-1} = -3, \quad d_{31}^{-1} = -2, \tag{6.20}$$

choose $s = 3$. It is also checked that $s = 3$ should be taken again for $k = 3$.

It remains to find the upper bound of the α parameter. write an auxiliary vector

$$\boldsymbol{h}^{-}\boldsymbol{D} = \begin{pmatrix} -1 & -2 & 0 \end{pmatrix}, \tag{6.21}$$

and consider the values

$$d_{31}(\boldsymbol{h}^{-}\boldsymbol{d}_1)^{-} = 3, \quad d_{32}(\boldsymbol{h}^{-}\boldsymbol{d}_2)^{-} = 3, \quad d_{33}(\boldsymbol{h}^{-}\boldsymbol{d}_3)^{-} = 3. \tag{6.22}$$

Since the minimum of these quantities is 3, we assume that$\alpha \leq 3$.

To obtain a solution corresponding to the latest possible reception start time, we set $\alpha = 3$. Let us find the components of the vector $\boldsymbol{u} = (u_1, u_2, u_3)^T$ provided that $k = 1$. Applying the formulas (6.13) we calculate

$$u_1 = 1, \quad u_2 \leq 2, \quad u_3 \leq 0. \tag{6.23}$$

For the case $k = 3$, the components of the vector $\boldsymbol{u}$ satisfy the conditions

$$u_1 \leq 1, \quad u_2 \leq 2, \quad u_3 = 0. \tag{6.24}$$

Choosing the vector $\boldsymbol{u} = (1, 2, 0)^T$, we obtain the solution of the problem in the following form

$$\boldsymbol{x} = \boldsymbol{B}^{*}\boldsymbol{u} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \quad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{u} = \begin{pmatrix} 5 \\ 4 \\ 3 \end{pmatrix}. \tag{6.25}$$

Chapter 7

Minimizing the spread of the jobs start times

In this chapter, the problem of drawing up an optimal calendar plan for the implementation of project jobs, which consists in minimizing the maximum spread in the start time of all project activities, with a full set of constraints is formulated and solved. The proposed solution method is to represent the planning problem in the language of tropical mathematics and then its solution using tropical optimization methods. In section 7.1 the optimal planning problem is formulated and solved. As in the previous problems, the solution is to reduce to the well-known tropical optimization problem. Further, in section 7.2, an example of using the obtained result for solving the problem of two-component vaccination is proposed.

7.1 Minimizing the dispersion in start times

This section proposes a solution to the problem of drawing up an optimal jobs plan in accordance with the criterion (1.10) of the minimum of the maximum spread in the start time of all jobs under constraints «start-start», «start-finish», «finish-start», «release time», «release deadline» and «completion deadline». The presented solution method consists in presenting the problem in the language of idempotent algebra in the form proposed in section 3.1 and its subsequent solution using tropical optimization methods. We write the problem under study in the notation from chapter 3 in the following form

$$\begin{aligned} \min_{x_i,y_i} \quad & \max_{1\le i\le n} x_i + \max_{1\le i\le n}(-x_i); \\ & \max_{1\le j\le n}(b_{ij}+x_j) \le x_i, \quad \max_{1\le j\le n}(c_{ij}+x_j) = y_i, \\ & \max_{1\le j\le n}(d_{ij}+y_j) \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1,\dots,n. \end{aligned} \tag{7.1}$$

It is easy to understand that the (7.1) problem can be written in the form of a linear programming problem and solved numerically using well-known computational procedures, such as the [107–109] simplex algorithm and the Karmarkar algorithm [110].

Let us reformulate the problem of minimizing the scatter in the start time of jobs in the language of tropical mathematics. Note that in terms of the idempotent semifield $\mathbb{R}_{\max,+}$ is the objective function

$$\max_{1\le i\le n} x_i + \max_{1\le i\le n} (-x_i) \tag{7.2}$$

can be written as follows:

$$\bigoplus_{1\le i\le n} x_i \bigoplus_{1\le j\le n} x_j^{-1}. \tag{7.3}$$

Let us imagine the constraints

$$\max_{1\le j\le n} (b_{ij} + x_j) \le x_i,\ \max_{1\le j\le n} (c_{ij} + x_j) = y_i,\ \max_{1\le j\le n} (d_{ij} + y_j) \le x_i,\ i = 1, \dots, n, \tag{7.4}$$

in a tropical way

$$\bigoplus_{1\le j\le n} b_{ij}x_j \le x_i, \quad \bigoplus_{1\le j\le n} c_{ij}x_j = y_i, \quad \bigoplus_{1\le j\le n} d_{ij}y_j \le x_i, \quad i = 1, \dots, n. \tag{7.5}$$

Problem (7.1) can be formulated in tropical form:

$$\begin{aligned} \min_{x_i, y_i} \ & \bigoplus_{1\le i\le n} x_i \bigoplus_{1\le j\le n} x_j^{-1}, \\ & \bigoplus_{1\le j\le n} b_{ij}x_j \le x_i, \quad \bigoplus_{1\le j\le n} c_{ij}x_j = y_i, \\ & \bigoplus_{1\le j\le n} d_{ij}y_j \le x_i, \quad g_i \le x_i \le h_i, \quad y_i \le f_i, \quad i = 1, \dots, n. \end{aligned} \tag{7.6}$$

Let us define the following vector notations

$$\boldsymbol{B} = (b_{ij}),\ \boldsymbol{C} = (c_{ij}),\ \boldsymbol{D} = (d_{ij}),\ \boldsymbol{x} = (x_i),\ \boldsymbol{f} = (f_i),\ \boldsymbol{g} = (g_i),\ \boldsymbol{h} = (h_i) \tag{7.7}$$

and write down the optimization problem

$$\begin{aligned} \min_{\boldsymbol{x}} \ & \boldsymbol{x}^{-}\mathbf{1}\mathbf{1}^T\boldsymbol{x}, \\ & \boldsymbol{B}\boldsymbol{x} \le \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} = \boldsymbol{y}, \\ & \boldsymbol{D}\boldsymbol{y} \le \boldsymbol{x}, \quad \boldsymbol{g} \le \boldsymbol{x} \le \boldsymbol{h}, \quad \boldsymbol{y} \le \boldsymbol{f}. \end{aligned} \tag{7.8}$$

Let us formulate and prove the following assertion

Theorem 15. *Let $\boldsymbol{B}$ and $\boldsymbol{D}$ be matrices, and $\boldsymbol{C}$ be a column-regular matrix such that the condition $\boldsymbol{R} = \boldsymbol{B} \oplus \boldsymbol{DC}$ is met* $\operatorname{Tr} \boldsymbol{R} \leq \mathbb{1}$. *Let $\boldsymbol{g}$ be a vector, and $\boldsymbol{f}$ and $\boldsymbol{h}$ be regular vectors such that for the vector $\boldsymbol{s}^T = \boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-$ the condition $\boldsymbol{s}^T \boldsymbol{R}^* \boldsymbol{g} \leq \mathbb{1}$ is met.*

Then the minimum value of the objective function in problem (7.8) *is equal to*

$$\theta = \|\boldsymbol{R}^*\| \oplus \bigoplus_{0 \leq i+j \leq n-2} \|\boldsymbol{s}^T \boldsymbol{R}^i\| \|\boldsymbol{R}^j \boldsymbol{g}\|, \tag{7.9}$$

and all regular solutions have the form

$$\boldsymbol{x} = \boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{G}\boldsymbol{u}, \qquad \boldsymbol{G} = \boldsymbol{R}^* \oplus \bigoplus_{0 \leq i+j \leq n-2} \theta^{-1} \boldsymbol{R}^i \mathbf{1}\mathbf{1}^T \boldsymbol{R}^j, \tag{7.10}$$

where $\boldsymbol{u}$ is any regular vector that satisfies the conditions

$$\boldsymbol{g} \leq \boldsymbol{u} \leq (\boldsymbol{s}^T \boldsymbol{G})^-. \tag{7.11}$$

Proof. Since $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$, the problem can be represented as follows

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^- \mathbf{1}\mathbf{1}^T \boldsymbol{x}, \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq \boldsymbol{h}. \end{aligned} \tag{7.12}$$

Let us combine inequalities $\boldsymbol{x} \leq \boldsymbol{h}$ и $\boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{f}$ in one $\boldsymbol{x} \leq (\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)^- = (\boldsymbol{s}^T)^-$.

Note that the original problem can be written in the following form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^- \mathbf{1}\mathbf{1}^T \boldsymbol{C}\boldsymbol{x}; \\ & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{x}, \quad \boldsymbol{g} \leq \boldsymbol{x} \leq (\boldsymbol{s}^T)^-. \end{aligned} \tag{7.13}$$

The resulting problem is a special case of the problem 3.1, where $\boldsymbol{p} = \boldsymbol{q} = \mathbf{1}$, and instead of the matrix $\boldsymbol{B}$ and $\boldsymbol{h}$ vectors are used the matrix $\boldsymbol{R}$ and the vector $(\boldsymbol{s}^T)^-$ respectively.

Note that since the value $\boldsymbol{q}^T \boldsymbol{p} = \mathbf{1}^T \mathbf{1} = \mathbb{1} > \mathbb{0}$ approval conditions are met 8.

Applying the theorem 8 and taking into account which, by the definition of the matrix norm,

$$\mathbf{1}^T \boldsymbol{C}\boldsymbol{R}^* \mathbf{1} = \|\boldsymbol{C}\boldsymbol{R}^*\|, \tag{7.14}$$

and, by definition of the vector norm,

$$\boldsymbol{h}^{-}\boldsymbol{R}^{i}\mathbf{1} = \|\boldsymbol{h}^{-}\boldsymbol{R}^{i}\|, \quad \mathbf{1}^{T}\boldsymbol{C}\boldsymbol{R}^{j}\boldsymbol{g} = \|\boldsymbol{C}\boldsymbol{R}^{j}\boldsymbol{g}\|, \tag{7.15}$$

we obtain the theorem under consideration.

To illustrate the result obtained, let us present an example of solving a practical problem related to emergency asistance.

7.2 Assistance to the wounded in an emergency

Let us simulate measures to help the wounded in an emergency (for example, a man-made or natural disaster). In such circumstances, from structures involved in rescuing the casualties requires prompt first aid to the wounded and their subsequent transfer to a safe medical hospital, where they can receive further assistance. Considering the presented problem, when scheduling assistance, it is required to provide first aid to the victims as soon as possible. However, the severity of injuries can vary greatly. and hospitalization of the more severely wounded may take precedence over first aid for the lightly wounded.

Let us introduce the following auxiliary notation.

x_i – first aid time for injured i;

y_i – time of sending the injured i to a safe hospital;

b_{ij} – the minimum time interval between rendering first aid to the injured $j = 1, \ldots, n$ and rendering first aid to the injured i (occurs, for example, due to a shortage of necessary specialists or equipment. It may also be due to the time it takes to get to the injured person.) ;

c_{ij} – the minimum time interval between rendering assistance to the injured person $j = 1, \ldots, n$ and sending the injured person i (due, for example, to the same blood group in the injured persons);

d_{ij} – the minimum period of time between sending the patient $j = 1, \ldots, n$ to the hospital and providing assistance to the patient i (such restrictions may

be related to the urgency of providing assistance to the injured j and the relatively good condition of the injured i);

g_i – the earliest possible time to provide first aid to an injured person i;

h_i – the latest possible time to provide first aid to an injured person i;

f_i – the latest allowable time for sending an injured person to the hospital i.

The task under study is a special case of the (7.1) task, which can be reformulated in terms of a semifield $\mathbb{R}_{\max,+}$ in the form of a problem (7.8).

Let us imagine the problem of optimizing the operation of a conditional point of care for the wounded, for which it is required to create a schedule of care for $n = 3$ victims. Time constraints are given by the following matrices and vectors:

$$\boldsymbol{B} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & -1 & 2 \\ -2 & -3 & 0 \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} -4 & -5 & -6 \\ \mathbb{0} & -4 & -7 \\ -5 & \mathbb{0} & -4 \end{pmatrix}, \tag{7.16}$$

$$\boldsymbol{g} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix} \quad \boldsymbol{f} = \begin{pmatrix} 6 \\ 6 \\ 6 \end{pmatrix}. \tag{7.17}$$

Let us check the conditions of the theorem 15. Note that the same matrices are used in the problem, as in the problem of the fastest possible vaccination. Thus we know that the conditions $\mathrm{Tr}(\boldsymbol{R}) \leq \mathbb{1} = 0$ and $\boldsymbol{s}^T \boldsymbol{R}^* \boldsymbol{g} = (\boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^-)\boldsymbol{R}^* \boldsymbol{g} = -1 < \mathbb{1}$ are executed.

Let us find the minimum spread θ of the start time. First, we calculate the auxiliary vectors:

$$\boldsymbol{s}^T = \boldsymbol{f}^- \boldsymbol{C} \oplus \boldsymbol{h}^- = \begin{pmatrix} -2 & -3 & -2 \end{pmatrix}, \; \boldsymbol{s}^T \boldsymbol{R} = \begin{pmatrix} -2 & -3 & -1 \end{pmatrix}, \; \boldsymbol{R}\boldsymbol{g} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}. \tag{7.18}$$

Find the following norms of matrices and vectors

$$\|\boldsymbol{R}^*\| = 2, \quad \|\boldsymbol{s}^T\| = -2, \quad \|\boldsymbol{s}^T \boldsymbol{R}\| = -1, \quad \|\boldsymbol{g}\| = 0, \quad \|\boldsymbol{R}\boldsymbol{g}\| = 2. \tag{7.19}$$

Find the the parameter value θ according to the formula

$$\theta = \|\boldsymbol{B}^*\| \oplus \|\boldsymbol{s}^T\|\|\boldsymbol{g}\| \oplus \|\boldsymbol{s}^T\boldsymbol{R}\|\|\boldsymbol{g}\| \oplus \|\boldsymbol{s}^T\|\|\boldsymbol{R}\boldsymbol{g}\| \oplus \|\boldsymbol{s}^T\boldsymbol{R}\|\|\boldsymbol{B}\boldsymbol{g}\| = 2. \quad (7.20)$$

Let us represent all solutions $\boldsymbol{x}$ of the problem in parametric form. Find auxiliary matrices:

$$\boldsymbol{R}\mathbf{1}\mathbf{1}^T = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{1}\mathbf{1}^T\boldsymbol{R} = \begin{pmatrix} 0 & -1 & 2 \\ 0 & -1 & 2 \\ 0 & -1 & 2 \end{pmatrix}. \quad (7.21)$$

We write the following matrices and a vector:

$$\boldsymbol{G} = \boldsymbol{R}^* \oplus (-2)(\mathbf{1}\mathbf{1}^T \oplus \boldsymbol{R}\mathbf{1}\mathbf{1}^T \oplus \mathbf{1}\mathbf{1}^T\boldsymbol{R}) = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix},$$

$$\boldsymbol{C}\boldsymbol{G} = \begin{pmatrix} 4 & 3 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix}, \qquad (\boldsymbol{s}^T\boldsymbol{G})^- = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \quad (7.22)$$

Finally, we write out all the solutions $\boldsymbol{x} = (x_1, x_2, x_3)^T$ using the parameter vector $\boldsymbol{u} = (u_1, u_2, u_3)^T$ as follows:

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix} \boldsymbol{u}, \qquad \boldsymbol{y} = \begin{pmatrix} 4 & 3 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{pmatrix} \boldsymbol{u}, \qquad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \le \boldsymbol{u} \le \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}. \quad (7.23)$$

Note that the columns of the matrices $\boldsymbol{G}$ and $\boldsymbol{C}\boldsymbol{G}$ are collinear and we rewrite them as follows

$$\boldsymbol{G} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -1 & -2 \end{pmatrix} \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix}, \quad \boldsymbol{C}\boldsymbol{G} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix}. \quad (7.24)$$

define an auxiliary vector $\boldsymbol{v}$

$$\boldsymbol{v} = \begin{pmatrix} 0 & \mathbb{0} & 1 \\ \mathbb{0} & 0 & \mathbb{0} \end{pmatrix} \boldsymbol{u}, \quad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \leq \boldsymbol{u} \leq \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \boldsymbol{v} \leq \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{7.25}$$

and use it to solve

$$\boldsymbol{x} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -1 & -2 \end{pmatrix} \boldsymbol{v}, \qquad \boldsymbol{y} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \end{pmatrix} \boldsymbol{v}, \qquad \begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \boldsymbol{v} \leq \begin{pmatrix} 2 \\ 3 \end{pmatrix}. \tag{7.26}$$

We write the result in terms of ordinary mathematics

$$\begin{aligned} &x_1 = \max(v_1, v_2 - 1), && x_2 = \max(v_1 + 1, v_2),\ x_3 = \max(v_1 - 1, v_2 - 2), \\ &y_1 = \max(v_1 + 4, v_2 + 3), && y_2 = \max(v_1 + 3, v_2 + 2),\ y_3 = \max(v_1 + 2, v_2 + 1), \\ & && 0 \leq v_1 \leq 2,\ 0 \leq v_2 \leq 2. \end{aligned} \tag{7.27}$$

Chapter 8

Minimizing the spread of the jobs start times

Let us study the problem of drawing up an optimal schedule for the execution of project jobs, which consists in maximizing the spread of the start time for the project jobs under given constraints. Section 8.2 proposes a solution to the problem under consideration network planning, which is based on the results presented in section 8.1. Section 8.3 discusses a practical example of using the obtained result to optimize the work of a polyclinic during quarantine.

8.1 Solution of an optimization problem with constraints

Assume that the matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{X}^{n\times n}$ and vectors $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{g} \in \mathbb{X}^n$ are given. Let us study the following optimization problem

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \boldsymbol{q}^{-}\boldsymbol{x}\,(\boldsymbol{A}\boldsymbol{x})^{-}\,\boldsymbol{p}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x}. \end{aligned} \tag{8.1}$$

Let us present a theorem proposing a solution to the problem under consideration.

Theorem 16 ([39]). *Let the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ be such that $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$ and the matrix $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^{*}$ has regular columns $\boldsymbol{d}_j = (d_{ij})$, where $\boldsymbol{B}^{*} = (\boldsymbol{b}_j^{*})$ is the Kleene matrix. Assume that the vectors $\boldsymbol{p} = (p_j)$ and $\boldsymbol{q} = (q_j)$ are regular.*

Then the maximum in the (8.1) *problem is $\Delta = \boldsymbol{q}^{-}\boldsymbol{B}^{*}\boldsymbol{D}^{-}\boldsymbol{p}$ and is achieved if and only if $\boldsymbol{x} = \boldsymbol{B}^{*}\boldsymbol{u}$, where $\boldsymbol{u} = (u_j)$ – any vector with components*

$$u_k = \alpha d_{sk}^{-1} p_s, \quad g_j \leq u_j \leq \alpha d_{sj}^{-1} p_s, \quad j \neq k, \tag{8.2}$$

provided that

$$\alpha \geq \max_{1\leq j\leq n} g_j d_{sj} q_s^{-1}, \quad k = \arg\max_{1\leq j\leq n} \boldsymbol{q}^{-}\boldsymbol{b}_j^{*}\boldsymbol{d}_j^{-}\boldsymbol{p}, \quad s = \arg\max_{1\leq j\leq n} d_{jk}^{-1} p_j. \tag{8.3}$$

Proof. Since for the inequality $\boldsymbol{Bx} \oplus \boldsymbol{g} \leq \boldsymbol{x}$ the condition of the 3 theorem is satisfied, we can replace the inequality with its solution $\boldsymbol{x} = \boldsymbol{B}^{*}\boldsymbol{u}$, where $\boldsymbol{u}$ – any regular vector such that $\boldsymbol{u} \geq \boldsymbol{g}$.

Substituting the solution of the inequality and replacing $\boldsymbol{AB}^{*}$ with $\boldsymbol{D}$ we obtain a new problem for the unknown vector $\boldsymbol{u}$:

$$\begin{gathered} \max_{\boldsymbol{u}} \ \boldsymbol{q}^{-}\boldsymbol{B}^{*}\boldsymbol{u}\,(\boldsymbol{Du})^{-}\,\boldsymbol{p}, \\ \boldsymbol{u} \geq \boldsymbol{g}. \end{gathered} \tag{8.4}$$

Consider this problem without the constraint $\boldsymbol{u} \geq \boldsymbol{g}$. Note that the matrix $\boldsymbol{D} = \boldsymbol{AB}^{*}$ has regular columns, and from the regularity of the vector $\boldsymbol{q}$ it follows that the vector $(\boldsymbol{q}^{-}\boldsymbol{B}^{*})^{-}$ is regular. Using the result of the theorem 7, we obtain that the maximum in the problem is equal to $\Delta = \boldsymbol{q}^{-}\boldsymbol{B}^{*}\boldsymbol{D}^{-}\boldsymbol{p}$ and is achieved if and only if the vector $\boldsymbol{u}$ has components

$$u_k = \alpha d_{sk}^{-1} p_s, \quad u_j \leq \alpha d_{sj}^{-1} p_s, \quad j \neq k, \tag{8.5}$$

for all scalars $\alpha > \mathbb{0}$ and indices k and s, which are determined by the following conditions

$$k = \arg\max_{1 \leq j \leq n} \boldsymbol{q}^{-}\boldsymbol{b}_j^{*}\boldsymbol{d}_j^{-}\boldsymbol{p}, \quad s = \arg\max_{1 \leq j \leq n} d_{jk}^{-1} p_j. \tag{8.6}$$

Let us find under what restrictions on the parameter α the vector $\boldsymbol{u}$ will satisfy the inequality $\boldsymbol{u} \geq \boldsymbol{g}$. Using the formula (8.5), we write out the conditions for the existence of such a vector $\boldsymbol{u}$ as

$$\alpha d_{sj}^{-1} p_s \geq g_j, \quad j = 1, \ldots, n. \tag{8.7}$$

Since all columns of the matrix $\boldsymbol{D}$ are regular, and the vector $\boldsymbol{p}$ is regular, the scalar $d_{sj}p_s^{-1}$ is non-zero. Multiplying both parts of the last inequality by $d_{sj}p_s^{-1}$ and taking into account monotonicity of multiplication, we obtain the inequality $\alpha \geq g_j d_{sj} p_s^{-1}$.

Then under the condition

$$\alpha \geq \max_{1 \leq j \leq n} g_j d_{sj} p_s^{-1}, \tag{8.8}$$

vector $\boldsymbol{u}$ whose components are calculated by the formulas (8.5)-(8.6), satisfies the inequality $\boldsymbol{u} \geq \boldsymbol{g}$, and the vector $\boldsymbol{x} = \boldsymbol{B}^{*}\boldsymbol{u}$ is the solution to problem (8.1).

Note that for an indecomposable matrix $\boldsymbol{B}$ the Kleene matrix $\boldsymbol{B}^*$ has no zero elements. Thus, if the matrix $\boldsymbol{A}$ is row regular, then the matrix $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^*$ will have regular columns. Therefore, to fulfill the conditions of the theorem 16 it suffices that the matrix $\boldsymbol{B}$ is indecomposable and regular in the rows of the matrix$\boldsymbol{A}$. Moreover, if the matrix $\boldsymbol{A}$ has only regular columns, the columns of the matrix $\boldsymbol{D}$ will be regular for any matrix $\boldsymbol{B}$.

Let us find the computational complexity of finding a solution to the problem. The main complexity is the search for the Kleene matrix, which will require no more than $O(n^4)$ scalar operations. The computational complexity of finding the remaining elements of the solution does not exceed $O(n^2)$, hence the total computational complexity of the problem does not exceed $O(n^4)$.

8.2 Maximization of start time spread

Let us imagine the solution of the problem of drawing up the optimal schedule for the execution of jobs in accordance with the criterion for the maximum spread of the start time of job (1.10) with type "start-start", restrictions as well as restrictions on the earliest start time of job

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \max_{1\le i\le n} x_i + \max_{1\le i\le n} (-x_i)\,, \\ & \max\left(\max_{1\le j\le n} (b_{ij} + x_j)\,, g_i\right) \le x_i, \quad i = 1,\dots,n. \end{aligned} \tag{8.9}$$

Note that problem (8.9) includes only the operations of determining the maximum, addition and subtraction, and therefore can be naturally represented in terms of the semifield $\mathbb{R}_{\max,+}$.

We represent the objective function in tropical form

$$\bigoplus_{i=1}^{n} x_i \bigoplus_{j=1}^{n} x_j. \tag{8.10}$$

Let us write the constraint inequality in terms of the semifield $\mathbb{R}_{\max,+}$

$$\bigoplus_{j=1}^{n} b_{ij}x_j \oplus g_i \le x_i, \qquad i = 1,\dots,n. \tag{8.11}$$

We introduce the following matrix-vector notation

$$\boldsymbol{B} = (b_{ij}), \quad \boldsymbol{g} = (g_i), \quad \boldsymbol{x} = (x_i) \tag{8.12}$$

and represent the objective function as an expression

$$\bigoplus_{i=1}^{n} x_i \bigoplus_{j=1}^{n} x_j = \mathbf{1}^T \boldsymbol{x} \boldsymbol{x}^{-} \mathbf{1}, \tag{8.13}$$

and constraints in the form of inequality

$$\boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x}. \tag{8.14}$$

Let us write the problem in vector form

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \mathbf{1}^T \boldsymbol{x} \boldsymbol{x}^{-} \mathbf{1}, \\ & \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{g} \leq \boldsymbol{x}. \end{aligned} \tag{8.15}$$

Let us formulate and prove the following theorem.

Theorem 17 ([39]). *Let the matrix $\boldsymbol{B} = (\boldsymbol{b}_j)$ with columns $\boldsymbol{b}_j = (b_{ij})$ be indecomposable and $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$. Then the maximum in* (8.15) *is $\Delta = \mathbf{1}^T \boldsymbol{B}^* (\boldsymbol{B}^*)^{-} \mathbf{1}$, where $\boldsymbol{B}^*$ is the Kleene matrix with columns $\boldsymbol{b}_j^* = (b_{ij}^*)$, and is reached if and only if $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$, where $\boldsymbol{u} = (u_j)$ – any vector with components*

$$u_k = \alpha (b_{sk}^*)^{-1}, \quad g_j \leq u_j \leq \alpha (b_{sj}^*)^{-1}, \quad j \neq k, \tag{8.16}$$

under the condition that

$$\alpha \geq \max_{1 \leq j \leq n} g_j b_{sj}^*, \quad k = \arg\max_{1 \leq j \leq n} \mathbf{1}^T \boldsymbol{b}_j^* (\boldsymbol{b}_j^*)^{-} \mathbf{1}, \quad s = \arg\max_{1 \leq j \leq n} (b_{jk}^*)^{-1}. \tag{8.17}$$

Proof. Consider problem (8.1) and set $\boldsymbol{p} = \mathbf{1}$, $\boldsymbol{q} = \mathbf{1}$ and$\boldsymbol{A} = \boldsymbol{I}$. In this case, problem (8.15) has the form of a problem (8.1). Make sure the theorem conditions are met 16. It is easy to see that the vectors $\boldsymbol{p}$ and $\mathrm{Tr}(\boldsymbol{B}) \leq \boldsymbol{q}$ are regular. Since the matrix $\boldsymbol{B}$ is indecomposable and $\mathbb{1}$, the matrix $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}^* = \boldsymbol{B}^*$ has regular columns. Therefore, the conditions of the theorem are satisfied.

Using the theorem, we get that the maximum in the problem is equal to $\Delta = \mathbf{1}^T \boldsymbol{B}^* (\boldsymbol{B}^*)^{-} \mathbf{1}$ and achieved f and only if $\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u}$, where $\boldsymbol{u}$ is a vector with components (8.16) satisfying the condition (8.17).

8.3 Optimizing the work of the clinic during quarantine

Consider the work of the clinic during the quarantine period. Even in a pandemic, many people need to carry out planned medical activities, such as visiting a dentist, therapist, endocrinologist. One way to reduce the number of social contacts is the maximum separation of the periods of visits to the polyclinic by citizens in time, that is, the maximization of the spread in the start time of the procedures.

Let us introduce auxiliary notation.

x_i – patient i visit time;

b_{ij} – the minimum time period between the visit of a patient $j = 1, \ldots, n$ and the visit of a patient i;

g_i – earliest possible patient i visit time.

The presented problem is a particular case of the (8.9) problem, which can be naturally represented in terms of the idemepotent semifield $\mathbb{R}_{\max,+}$ in the form (8.15).

Let us describe the work of a conditional polyclinic, for which it is necessary to draw up a schedule for receiving $n = 3$ patients with restrictions on the time of admission, specified using a matrix and a vector:

$$\boldsymbol{B} = \begin{pmatrix} \mathbb{0} & -2 & 1 \\ 0 & \mathbb{0} & 2 \\ -1 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}. \tag{8.18}$$

Let us verify that the condition $\mathrm{Tr}(\boldsymbol{B}) \leq \mathbb{1}$ is satisfied and find the Kleene matrix $\boldsymbol{B}^*$. First, we calculate the matrices

$$\boldsymbol{B}^2 = \begin{pmatrix} 0 & \mathbb{0} & 0 \\ 1 & -2 & 1 \\ \mathbb{0} & -3 & 0 \end{pmatrix}, \quad \boldsymbol{B}^3 = \begin{pmatrix} -1 & -2 & 1 \\ 0 & -1 & 2 \\ -1 & \mathbb{0} & -1 \end{pmatrix}. \tag{8.19}$$

Then we find the value of the idempotent analogue of the determinant $\mathrm{Tr}(\boldsymbol{B}) = \mathrm{tr}\,\boldsymbol{B} \oplus \mathrm{tr}\,\boldsymbol{B}^2 \oplus \mathrm{tr}\,\boldsymbol{B}^3 = 0 = \mathbb{1}$. Therefore, the conditions of the theorem 17 are

satisfied. We write the Kleene matrix

$$\boldsymbol{B}^* = \boldsymbol{I} \oplus \boldsymbol{B} \oplus \boldsymbol{B}^2 = \begin{pmatrix} 0 & -2 & 1 \\ 1 & 0 & 2 \\ -1 & -3 & 0 \end{pmatrix}. \tag{8.20}$$

Knowing the Kleene matrix $\boldsymbol{B}^*$, we find the matrices

$$(\boldsymbol{B}^*)^- = \begin{pmatrix} 0 & -1 & 1 \\ 2 & 0 & 3 \\ -1 & -2 & 0 \end{pmatrix}, \quad \boldsymbol{B}^*(\boldsymbol{B}^*)^- = \begin{pmatrix} 0 & -1 & 1 \\ 2 & 0 & 3 \\ -1 & -2 & 0 \end{pmatrix}, \tag{8.21}$$

and calculate the maximum $\Delta = \mathbf{1}^T \boldsymbol{B}^* (\boldsymbol{B}^*)^- \mathbf{1} = 3$ objective function of the problem.

Then we find the indices k, s and the lower bound of the parameter α, which are determined using the conditions (8.17). First we calculate

$$\mathbf{1}^T \boldsymbol{b}_1^* (\boldsymbol{b}_1^*)^- \mathbf{1} = 2, \quad \mathbf{1}^T \boldsymbol{b}_2^* (\boldsymbol{b}_2^*)^- \mathbf{1} = 3, \quad \mathbf{1}^T \boldsymbol{b}_3^* (\boldsymbol{b}_3^*)^- \mathbf{1} = 2, \tag{8.22}$$

whence it follows that the value of the index k is equal to 2. Next we find

$$(b_{12}^*)^{-1} = 2, \quad (b_{22}^*)^{-1} = 0, \quad (b_{32}^*)^{-1} = 3, \tag{8.23}$$

and choose the value $s = 3$.

After calculation

$$g_1 b_{31}^* = 1, \quad g_2 b_{32}^* = -3, \quad g_1 b_{31}^* = 0, \tag{8.24}$$

we find the maximum equal to 1, from which it follows that $\alpha \geq 1$.

Let us define the vector $\boldsymbol{x}$ of the earliest jobs start dates corresponding to the value of the parameter $\alpha = 1$. First, Let us find the boundaries (8.16) for all components of the vector $\boldsymbol{u} = (u_1, u_2, u_3)^T$ provided that $\alpha = 1$. Having carried out the corresponding calculations, we obtain

$$u_1 = 2, \quad u_2 = 4, \quad 0 \leq u_3 \leq 1. \tag{8.25}$$

Assume that $u_3 = 0$ and from the vector $\boldsymbol{u} = (2, 4, 0)^T$ we find the vector

$$\boldsymbol{x} = \boldsymbol{B}^* \boldsymbol{u} = \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix}. \tag{8.26}$$

Taking into account that for $u_3 = 1$ the solution remains the same, we will assume that the earliest start time of jobs is given by the equalities

$$x_1 = 2, \quad x_2 = 4, \quad x_3 = 1. \tag{8.27}$$

Conclusion

Thus, the following results were obtained in the paper.

- The problem of maximizing the maximum duration of the project under constraints of the form «start-start», «start-finish», «finish-start», «release time», «release deadline» and «completion deadline» is studied .
- The problem of minimizing the output of the maximum deviation from the due dates for the execution of jobs under constraints of the form «start-start», «start-finish», «finish-start», «release time», «release deadline» and «completion deadline» is considered .
- The problem of minimizing the maximum scatter of project completion time under constraints of the form «start-start» and «start-finish» is considered.
- The problem of maximizing the maximum scatter of the start time of the project execution under constraints of the form «start-start» and «release time» is investigated.
- The problem of minimizing the maximum spread of the start time of the project under constraints such as «start-start», «release time» and «completion deadline» is studied.
- The problem of maximizing the maximum scatter of project completion time under constraints of the «start-start», «start-finish» and «completion deadline» types is studied.
- The presented problems are reformulated in the language of tropical optimization.
- Direct analytical solutions of the considered optimization problems are proposed, which can be used both for practical problems and for formal analysis.
- For matrix-vector operations of idempotent algebra and solutions of the optimization problems under consideration, their software implementation is presented.

- For all the problems under consideration, practical measurements of their use for solving real practical problems are presented and explanatory numerical examples are proposed.

Bibliography

1. Kelley J., Walker M. Critical-path planning and scheduling // Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM Computer Conference. ACM. 1959. P. 160–173.

2. Saplosky H. The Polaris system development. Cambridge: Harvard University Press, 1972. 272 p.

3. Pinedo M. Scheduling. 3rd ed. New York: Springer, 2008.

4. Vanhoucke M. Project Management with Dynamic Scheduling. 2nd ed. Berlin: Springer, 2012.

5. Blazewicz J., Ecker K., Pesch E. et al. Handbookon Scheduling. In: International Handbooks on Information Systems. Cham, Springer, 2019.

6. Maslov, V., Kolokolzov, V. Idempotent analysis and its application in optimal control. Moscow: Fizmatlit, 1994. 144 p.

7. Krivulin, N. Methods of idempotent algebra in problems of modeling and analysis of complex systems. Publishing house of St. Petersburg State University, 2009. 256 p.

8. Cuninghame-Green R. Minimax algebra and applications // Advances in Imaging and Electron Physics. 1994. Vol. 90. P. 1–121.

9. Zimmermann U. Linear and combinatorial optimization in ordered algebraic structures. Amsterdam: Elsevier, 2011. 390 p.

10. Baccelli F.L., Cohen G., Olsder G.J., Quadrat J.P. Synchronization and linearity: an algebra for discrete event systems. Chichester: Wiley, 1992. 514 p.

11. Grigoriev D., Shpilrain V. Tropical cryptography // Communications in Algebra. 2014. Vol. 42, no. 6. P. 2624–2632.

12. Zimmermann K. Disjunctive optimization, max-separable problems and extremal algebras // Theoretical Computer Science. 2003. Vol. 293, no. 1. P. 45–54.

13. Tharwat A., Zimmermann K. One class of separable optimization problems: solution method, application // Optimization. 2010. Vol. 59, no. 5. P. 619–625.

14. Krivulin, N., Plotnikov, P. Using Tropical Optimization to Solve Minimax Location Problems with Rectangular Metric on a Line // Vestnik of St. Petersburg University. Series 1. Mathematics. Mechanics. Astronomy. 2016. Vol. 3, no. 4. P. 602–614.

15. Cuninghame-Green R. Describing industrial processes with interference and approximating their steady-state behaviour // Journal of the Operational Research Society. 1962. Vol. 13, no. 1. P. 95–100.

16. Nikolaev, D. Modeling and controlling the movement of an agent in an uncertain environment by idempotent algebra methods // Large Systems Management: Proceedings. 2012. no. 40. P. 311–328.

17. Matveenko V.D. Optimal trajectories of the dynamic programming scheme and extremal degrees of nonnegative particles // Discrete Math. 1990. Vol. 2, no. 1. P. 59–71.

18. Krivulin, N., Romanovskiy, I. Solving mathematical programming problems using tropical optimization methods // Vestnik of St. Petersburg University. Series 1. Mathematics. Mechanics. Astronomy. 2017. Vol. 4, no. 3. P. 448–458.

19. Cuninghame-Green, R.A. Projections in minimax algebra // Mathematical Programming. 1976. Vol. 10, no. 1. P. 111–123.

20. Krivulin N. A multidimensional tropical optimization problem with a non-linear objective function and linear constraints // Optimization. 2015. Vol. 64, no. 5. P. 1107–1129.

21. Butkovic P., Aminu A. Introduction to max-linear programming // IMA Journal of Management Mathematics. 2009. Vol. 20, no. 3. P. 233–249.

22. Heidergott B., Olsder G. Max-plus at Work: Modeling and Analysis of Synchronized Systems. Princeton Series in Applied Mathematics. Princeton: Princeton University Press, 2006. 226 p.

23. Krivulin N. A constrained tropical optimization problem: Complete solution and application example // Tropical and Idempotent Mathematics and Applications. 2014. Vol. 616. P. 163–177.

24. Krivulin N. Complete solution of a constrained tropical optimization problem with application to location analysis // International Conference on Relational and Algebraic Methods in Computer Science / Springer. 2014. P. 362–378.

25. Krivulin N. Extremal eigenvalue property of indecomposable matrices in idempotent algebra and solution of the Rawls location problem // Vestnik of St. Petersburg University. Series 1. Mathematics. Mechanics. Astronomy. 2011. Vol. 44, no. 4. P. 272–284.

26. Butkovič P. Max-linear systems: theory and algorithms. Springer Science & Business Media, 2010.

27. Grigoriev D. On a tropical dual Nullstellensatz // Advances in Applied Mathematics. 2012. Vol. 48, no. 2. P. 457–464.

28. Grigoriev D., Koshevoy G. Complexity of tropical Schur polynomials // Journal of Symbolic Computation. 2016. Vol. 74. P. 46–54.

29. Krivulin N. Tropical optimization problems // Advances in Economics and Optimization: Collected Scientific Studies Dedicated to the Memory of L. V. Kantorovich. 2014. P. 195–214.

30. Krivulin N. Extremal properties of tropical eigenvalues and solutions to tropical optimization problems // Linear Algebra and its Applications. 2015. Vol. 468. P. 211–232.

31. Zimmermann K. Interval linear systems and optimization problems over max-algebras // Linear Optimization Problems with Inexact Data. 2009. P. 165–193.

32. Krivulin N. Explicit solution of a tropical optimization problem with application to project scheduling // Mathematical Methods and Optimization Techniques in Engineering. 2013. Vol. 20, no. 3. P. 39–45.

33. Krivulin N. A maximization problem in tropical mathematics: A complete solution and application examples // Informatica. 2016. Vol. 27, no. 3. P. 587–606.

34. Krivulin N. Tropical optimization problems in project scheduling // MISTA 2015:Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications. 2015. P. 492–506.

35. Krivulin N. Direct solution to constrained tropical optimization problems with application to project scheduling // Computational Management Science. 2017. Vol. 14, no. 1. P. 91–113.

36. Krivulin N. Tropical optimization problems with application to project scheduling with minimum makespan // Annals of Operations Research. 2017. Vol. 256, no 1. P. 75–92.

37. Krivulin N. Tropical optimization problems in time-constrained project scheduling // Optimization. 2017. Vol. 66, no. 2. P. 205–224.

38. Krivulin, N., Gubanov, S. Solving the network planning problem based on tropical optimization methods // Vestnik of St. Petersburg University. Series 10. Applied Mathematics. Informatics. Management processes. 2016. no. 3. P. 62–72.

39. Krivulin, N., Gubanov, S. Using tropical optimization methods in network planning problems // Vestnik of St. Petersburg University. Series 10. Applied Mathematics. Informatics. Management processes. 2017. no. 4. P. 384–397.

40. Krivulin, N., Gubanov, S. Algebraic solution of the problem of optimal project scheduling in project management // Vestnik of St. Petersburg University. Maths. Mechanics. Astronomy. 2021. Vol. 8(66), no. 1. P. 73–87.

41. Gubanov, S. Algebraic solution of optimal planning problems, taking into account the directive dates for the start of the project // Vestnik of St. Petersburg University. Maths. Mechanics. Astronomy. 2022. Vol. 9(67), no. 4. P. 602–611.

42. Krivulin, N., Gubanov, S. Tropical optimization problem solution with application to network planning // Interdisciplinary research in the field of mathematical modeling and informatics. Materials of the 6th scientific and practical internet-conference. Tolyatti May 14-15, 2015. no. 4. P. 224–227.

43. Krivulin, N., Gubanov, S. Using tropical optimization techniques for optimal scheduling // Proceedings of the 7th All-Russian Scientific Conference on Informatics Problems April 26-28, 2017. P. 515–522.

44. Krivulin, N., Gubanov, S. Job scheduling using tropical optimization methods // Materials of the All-Russian Scientific Conference on Informatics Problems April 23-26, 2019. P. 319–325.

45. Krivulin, N., Gubanov, S. Solving network planning problems based on tropical mathematics methods // Models and methods of tropical mathematics in applied problems of economics and management. Collection of scientific articles. 2014. no. 2. P. 99–121.

46. Gubanov S. Algebraic solution of the planning problem in project management // Polynomial computer algebra. Materials of the international conference. 2022. P. 28–32.

47. Gubanov S. Solving the problem of optimal project management using tropical optimization methods // Abstracts of the XXIII All-Russian Conference of Young Scientists on Mathematical Modeling and Information Technologies. 2022. P. 50.

48. Krivulin, N., Gubanov, S. Solving a problem of optimal project scheduling by methods of tropical mathematics // Mathematical methods and models in high-tech production. Collection of abstracts of reports of the II International Forum. 2022. P. 366–369.

49. Gubanov, S. Solving the problem of compiling the optimal calendar schedule for the implementation of project jobs using tropical optimization methods // Modern problems of mechanical engineering. Proceedings of the XV International Scientific and Technical Conference. 2022. P. 193–194.

50. Demeulemeester E., Herroelen W. Project scheduling: a research handbook. International Series in Operations Research and Management Science. Springer, 2002.

51. T'kindt V., Billaut J.-C. Multicriteria scheduling: theory, models and algorithms. Springer Science & Business Media, 2006.

52. Cuninghame-Green R. Projections in minimax algebra // Mathematical Programming. 1976. Vol. 10, no. 1. P. 111–123.

53. Zimmermann U. Linear and combinatorial optimization in ordered algebraic structures. 1981. Vol. 10.

54. Golan J. Semirings and affine equations over them: theory and applications. Springer Science & Business Media, 2013. Vol. 556.

55. Gondran M., Minoux M. Graphs, dioids and semirings: new models and algorithms. Springer Science & Business Media, 2008. Vol. 41.

56. Pandit S. A new matrix calculus // Journal of the Society for Industrial and Applied Mathematics. 1961. Vol. 9, no. 4. P. 632–639.

57. Ciffler B. Scheduling general production systems using schedule algebra // Naval Research Logistics Quarterly. 1963. Vol. 10, no. 1. P. 237–255.

58. Dedekind R. Über die Theorie der ganzen algebraischen Zahlen // Vorlesungen Ëuber Zahlentheorie. Druck und Verlag Braunschweig, 1894. P. 434–657.

59. Golan J. Semirings and their Applications. Springer Science & Business Media, 1999.

60. Glazek K. A guide to the literature on semirings and their applications in mathematics and information sciences: with complete bibliography. Springer Science & Business Media, 2002.

61. Vandiver H. Note on a simple type of algebra in which the cancellation law of addition does not hold // Vestnik of the American Mathematical Society. 1934. Vol. 40, no. 12. P. 914–920.

62. Stephen C. Kleene. Representation of events in nerve nets and finite automata // Automata Studies. 1956. P. 3–43.

63. Cuninghame-Green R. Describing industrial processes with interference and approximating their steady-state behaviour // Journal of the Operational Research Society. 1962. Vol. 13, no. 1. P. 95–100.

64. Vorobiev, N. Extreme Matrix Algebra // Reports of the Academy of Sciences of the USSR. 1963. Vol. 152, no. 1. P. 24–27.

65. Vorobiev, N. Extremal algebra of positive matrices // Elektronische Informationsverarbeitung und Kybernetik. 1967. Vol. 3, no. 1. P. 39.

66. Vorobiev, N. Extremal algebra of positive matrices // Elektronische Informationsverarbeitung und Kybernetik. 1970. Vol. 6, no. 4-5. P. 303–312.

67. Romanovsky, I. A Few Remarks on the Bellman-Karush Functional Transformation // Vestnik of LGU. 1962. P. 148–150.

68. Romanovsky, I. Asymptotic behavior of dynamic programming processes with a continuous set of states // Reports of the Academy of Sciences of the USSR / Russian Academy of Sciences. Vol. 159. 1964. P. 1224–1227.

69. Birkhoff G. Lattice theory. 1948. Vol. 25. 311 p.

70. Kantorovich, L. Functional analysis in semi-ordered spaces. State. publishing house of technical-theoretic literature, 1950. 548 p.

71. Lunz, A. Algebraic methods of analysis and synthesis of contact circuits // News of the Russian Academy of Sciences. Mathematical series. 1952. Vol. 16, no. 5. P. 405–426.

72. Povarov, N. Matrix Methods for Analyzing Relay-Contact Circuits by Failure Conditions // Automation and telemechanics. 1954. Vol. 15, no. 4. P. 332–335.

73. Shimbel A. Structure in communication nets // Proceedings of the symposium on information networks / Polytechnic Institute of Brooklyn. 1954. P. 119–203.

74. Bellman R., Karush W. On a new functional transform in analysis: the maximum transform: Tech. rep.: System Development Corp Santa Monica CA, 1961.

75. Kantorovich, L. On the movement of masses // Reports of the Academy of Sciences of the USSR. 1942. Vol. 37, no. 7-8. P. 227—-229.

76. Korbut, A. Extreme spaces // Reports of the Academy of Sciences of the USSR / Russian Academy of Sciences. Vol. 164. 1965. P. 1229–1231.

77. Korbut, A. Extremal vector spaces and their properties // Elektronische Informationsverarbeitung und Kybernetik. Vol. 8/9. 1972. P. 525–536.

78. Dudnikov, P., Samboskiy, S. Endomorphisms of semimodules over semirings with idempotent operation // News of the Russian Academy of Sciences. Mathematical series. 1991. Vol. 55, no. 1. P. 93–109.

79. Maslov V. On a new superposition principle for optimization problem // Séminaire Équations aux dérivées partielles (Polytechnique). 1985. P. 1–14.

80. Maslov, V.P. On a new superposition principle for optimization problems // Advances in Mathematical Sciences. 1987. Vol. 42, no. 3 (255. P. 39–48.

81. Dobrokhotov S.Y., Kolokoltsov V.N., Maslov V.P. Quantization of the Bellman equation, exponential asymptotics and tunneling // Advances in Soviet Mathematics. 1992. Vol. 13. P. 1–46.

82. Maslov V.P., Samborskii S.N. Stationary Hamilton–Jacobi and Bellman equations (existence and uniqueness of solutions) // Idempotent Analysis. 1992. Vol. 41. P. 87–101.

83. Litvinov G.L., Maslov V.P., Shpiz G.B. Idempotent functional analysis. Algebraic approach // Math notes. 2001. Vol. 69, no. 5. P. 758–797.

84. Litvinov G., Maslov V. The correspondence principle for idempotent calculus and some computer applications // Idempotency. 1998. Vol. 11. P. 420–443.

85. Litvinov G.L. Maslov dequantization, idempotent and tropical mathematics: a short introduction // Notes of scientific seminars POMI. 2005. Vol. 326, no. 0. P. 145–182.

86. Sergeev S. Minimal elements and cellular closures over the max-plus semiring // Idempotent and Tropical Mathematics and Problems of Mathematical Physics. 2007. Vol. 2. P. 49–52.

87. Gaubert, S., Katz, R.D., Sergeev, S. Tropical linear-fractional programming and parametric mean payoff games // Journal of Symbolic Computation. 2012. Vol. 47, no. 12. P. 1447–1478.

88. Sergeev S. Max-algebraic attraction cones of nonnegative irreducible matrices // Linear Algebra and its Applications. 2011. P. 1736–1757.

89. Mikhalkin G. Amoebas of algebraic varieties and tropical geometry // Different Faces of Geometry. 2004. P. 257–300.

90. Itenberg I., Mikhalkin G., Shustin E. Tropical algebraic geometry. Springer Science & Business Media, 2009. Vol. 35.

91. Brugallé E., Itenberg I., Mikhalkin G., Shaw K. Brief introduction to tropical geometry. 2015. P. 1–75.

92. Kazaryan M.E. Tropical geometry. 2012. 43 p.

93. Zimmermann K. Some optimization problems with extremal operations // Mathematical Programming at Berwolfach. 1984. Vol. 22. P. 237–251.

94. Butkovič P, Tam K.P. On some properties of the image set of a max-linear mapping // Tropical and Idempotent Mathematics. 2009. Vol. 495. P. 233–249.

95. Butkovič P., Aminu A. Non-linear programs with max-linear constraints: a heuristic approach // IMA Journal of Management Mathematics. 2012. no. 1. P. 41–66.

96. Hoffman A. J. On abstract dual linear programs // Naval Research Logistics Quarterly. 1963. no. 1. P. 369–373.

97. Zimmermann, K., Gavalec, M. Duality for max-separable problems // Central European Journal of Operations Research. 2012. P. 409–419.

98. Superville L. Various aspects of max-algebra // PhD thesis, The City University of New York. 1978. P. 409–419.

99. Zimmermann K. On max-separable optimization problems // Algebraic and Combinatorial Methods in Operations Research. 1984. Vol. 95. P. 357–362.

100. Zimmermann K. Optimization problems with unimodal functions in max-separabal constraints // Optimization. 1992. no. 1-2. P. 31–41.

101. Cuninghame-Green R. A., Butkovič P. The equation $a \oplus x = b \oplus y$ over (max,+) // Theoretical Computer Science. 2003. no. 1. P. 3–12.

102. Butkovič P. On properties of solution sets of extremal linear programs // Algebraic and Combinatorial Methods in Operations Research. 1984. P. 41–54.

103. Akian M., Gaubert S., A. Guterman A. Tropical polyhedra are equivalent to mean payoff games // International Journal of Algebra and Computation. 2012. Vol. 22, no. 1.

104. Krivulin N. Eigenvalues and eigenvectors of matrices in idempotent algebra // Vestnik St. Petersburg University, Mathematics. 2006. no. 2. P. 72–83.

105. Krivulin N. Solution of generalized linear vector equations in idempotent algebra // Vestnik St. Petersburg University, Mathematics. 2006. no. 1. P. 16–26.

106. Krivulin N. A new algebraic solution to multidimensional minimax location problems with Chebyshev distance // WSEAS Transactions on Mathematics. 2012. no. 7. P. 146–151.

107. Kantorovich L.V. Mathematical methods of organizing and planning production // Management Sci. 1960. Vol. 6, no. 4. P. 366–422.

108. Romanovskiy, I. Algorithms for solving extreme problems. Moscow: Science, 1977. 352 p.

109. Vasiliev, F., Ivanizkiy, A. Linear programming. 3rd ed. Moscow: Factorial Press, 2008.

110. Karmarkar N. A new polynomial-time algorithm for linear programming // Combinatorica. 1984. Vol. 4, no. 4. P. 373–395.

Appendix A **Software implementation of idempotent algebra scalar operations**

First, we consider the software implementation of the idempotent algebra scalar operations necessary for solving tropical optimization problems. The presented version of the software implementation is implemented in C++ high-level language.

A.1 Software structure

- Abstract class `Algebra`, which implements assignment of an idempotent algebra. The class has the following public methods:
  - Function `double oplus(const double& a, const double& b)` defines the operation of idempotent addition. The function takes scalar arguments a and b as input and returns the value of the sum $a \oplus b$;
  - Function `double otimes(const double& a, const double& b)` defines idempotent multiplication. The function takes scalar arguments a and b as input and returns the value of the product $a \otimes b$;
  - Function `bool order(const double& a, const double& b)` specifies a partial order relation. In case? if $a \leq b$ the function returns true, and false otherwise;
  - Function `pseudo_inverse(const double& a)` implements the calculation of the pseudo-inverse element The function takes a scalar argument as input a and returns a^{-};
  - Function `double get_zero()` returns idempotent zero;
  - Function `get_unit()` returns an idempotent one;
- Class `R_max_plus` is an inheritor of the `Algebra` class and is used to implement algebra $\mathbb{R}_{\max,+}$.

A.2 Program listing

```
class Algebra
{
public:
    virtual const double oplus(const double& a,const double& b)= 0;
    virtual const double otimes(const double& a,const double& b)= 0;
    virtual const bool order(const double& a,const double& b)= 0;
    virtual const double pseudo_inverse(const double& a)= 0;
    virtual const double power(const double& x,const double& y)= 0;
    virtual const double get_zero() {return idemp_zero;}
    virtual const double get_unit() {return idemp_unit;}

protected:
    double idemp_zero;
    double idemp_unit;
};

class R_max_plus : public Algebra
{
public:
    R_max_plus()
    {
        idemp_zero = -1.0e+10;
        idemp_unit = 0.0;
    }
    const double oplus(const double& a, const double& b)
    {
        return std::max(a,b);
    }
    const double otimes(const double& a, const double& b)
    {
        return (a + b);
    }
    const bool order(const double& a, const double& b)
    {
        return a <= b;
    }
    const double pseudo_inverse(const double& a)
    {
        return -a;
    }
    const double power(const double& x, const double& y)
    {
        return x * y;
    }
};
```

Appendix B **Software implementation of matrix-vector operations in idempotent algebra**

The software representation of the above optimization problems will also require implementation of matrix-vector operations. Let us propose a variant of such an implementation in the high-level C++ language.

B.1 Software structure

- Function `std::vector<std::vector<size_t>> get_all_pairs (size_t n_min_k)` returns all vectors consisting of two indices such that their sum does not exceed `n_min_k`;
- Template class `Matrix<T>`, used to represent matrix operations on an idempotent algebra. As a template class, the inheritor class of the one described in the application A class `Algebra` is used . The class has the following public methods:
  - `Matrix()` – default constructor;
  - `Matrix(size_t _row,size_t _col)` – constructor with parameters. The number of lines `_row` and number of columns`_col` is taken as input.
  - `Matrix& operator = (const Matrix& matrix)` – assignment operator
  - Function `double get_zero()` – returns the matrix zero element of ;
  - Function `double get_unit()` – returns the matrix identity element;
  - Function `double get_rows()` – returns the number of matrix rows;
  - Function `double get_cols()` – returns the number of matrix columns;
  - Function `bool is_quadr()` – returns true if the matrix is square;
  - Function `void Zero()` – fills matrix with zeros (from template class);
  - Function `void Identity()` – makes the matrix identity;
  - Function `void Identity_vec()` – makes the vector (matrix-column) identity ;

- Operator `double& operator()(int _row, int _col)` – gives access to the element with `_row` and `_col` indices;
- Operator `Matrix operator*(const Matrix& matrix)` – performs matrix multiplication;
- Operator `Matrix operator*(double alpha)` – performs matrix multiplication by scalar;
- Operator `Matrix operator+(const Matrix& matrix)` – performs matrix addition;
- Operator `Matrix operator^(size_t power)` – provides raising the matrix to the power;
- Function `Matrix ast()` – calculates the Kleene matrix;
- Function `double tr()` – evaluates the tr operator;
- Function `double Tr()` – evaluates the Tr operator;
- Function `Matrix transpose()` – returns the transposed matrix;
- Function `Matrix pseudo_inverse()` – returns pseudo inverse matrix;
- Function `double norm()` – calculates the norm of a matrix
- Function `bool is_row_regular()` – checks that the matrix is row regular;
- Function `bool is_col_regular()` – checks that the matrix is column regular;
- Function `bool is_regular()` – checks if a matrix is regular;
- Function `bool regular_cols()` – checks if a matrix has regular columns;
- Function `void to_file(std::ofstream& outf)` – outputs the matrix to a file;
- Function `Matrix get_col(size_t j)` – returns a column;
- Function `bool is_razl()` – checks if the matrix is decomposable;

B.2 Program listing

```
typedef std::unique_ptr<double[]> DBPT;
std::vector<std::vector<size_t>> get_all_pairs(size_t n_min_k)
{
    std::vector<std::vector<size_t>> res;
    if(n_min_k < 0)
    {
        return res;
    }
    for(size_t i = 0; i <= n_min_k; i++)
        for(size_t j = 0; j <= n_min_k; j++)
            if(i+j <= n_min_k)
            {
                std::vector<size_t> new_res;
                new_res.push_back(i);
                new_res.push_back(j);
                res.push_back(new_res);
            }
    return res;
}

template<class T>
class Matrix
{
private:
    T alg;
    size_t row;
    size_t col;
public:
    double get_zero() {return alg.get_zero();}
    double get_unit() {return alg.get_unit();}


    std::unique_ptr<DBPT[]> elems;
    string errbuf;
    //char* errbuf;

    Matrix()
    {
        row = 0;
        col = 0;
    }

    ~Matrix()
    {
    }

    Matrix(size_t _row,size_t _col)
    {
        row = _row;
        col = _col;
        elems.reset(new DBPT[row]);
        for(int i = 0; i < row; i++)
```

```
        elems.get()[i].reset(new double[col]);
    Zero();
}

Matrix(Matrix& matrix)   //copy constructor
{
    *this = matrix;
}

Matrix& operator = (const Matrix& matrix)
{
    row = matrix.row;
    col = matrix.col;
    elems.reset(new DBPT[row]);
    for(int i = 0; i < row; i++)
        elems.get()[i].reset(new double[col]);
    for(int i = 0; i < row; i++)
        for(int j = 0; j < col; j++)
            (*this)(i,j) = matrix(i, j);
    return *this;
}

double get_rows()
{
    return row;
}

double get_cols()
{
    return col;
}

bool is_quadr()
{
    return row == col;
}

void Zero()
{
    for(int i = 0; i < row; i++)
        for(int j = 0 ; j < col; j++)
            (*this)(i,j) = alg.get_zero();
}

void Identity()
{
    if (row != col)
        throw "Error identity, matrix is not squared";
    Zero();
    for(int i = 0; i < row; i++)
            (*this)(i,i) = alg.get_unit();
}

void Identity_vec()
```

```
{
    if (col != 1)
        throw "Error identity, object is not a vector";
    for(int i = 0; i < row; i++)
            (*this)(i,0) = alg.get_unit();
}

double& operator()(int _row, int _col)
{
    return elems[_row].get()[_col];
}

double operator()(int _row, int _col) const
{
    return elems[_row].get()[_col];
}


Matrix operator*(const Matrix& matrix)
{
    if (col != matrix.row)
        throw "Error * , different matrix dimentions ";

    Matrix res(row, matrix.col);
    res.Zero();

    for(int i = 0 ; i < row ; i++){
        for(int j = 0 ; j < matrix.col; j++)
            for(int k = 0 ; k < col; k++){
                res(i,j)
                = alg.oplus(
                    res(i,j),
                    alg.otimes(
                        this->operator ()(i,k)
                        , matrix(k,j)));
            }
    }
    return res;
}

Matrix operator*(double alpha)
{
    Matrix res(row, col);
    res.Zero();

    for(int i = 0 ; i < row ; i++){
        for(int j = 0 ; j < col; j++){
                res(i,j) =
                 alg.otimes(this->operator ()(i,j), alpha);
            }
    }
    return res;
}
```

```
Matrix operator+(const Matrix& matrix)
{
    if (row != matrix.row || col != matrix.col)
    {
        throw "Error + , different matrix dimentions ";
    }
    Matrix res(row, col);
    for(int i = 0; i < row; i++)
        for(int j=0; j < col; j++)
            res(i,j) =
             alg.oplus(this->operator()(i, j), matrix(i, j));
    return res;
}

Matrix operator^(size_t power)
{
    if (row != col)
        throw "Error in power, matrix is not squared";
    Matrix res(row, col);
    res.Identity();
    for(size_t i = 0; i < power; i++)
    {
        res = res * (*this);
    }
    return res;
}

Matrix ast()
{
    if (row != col)
        throw "Error ast, matrix is not squared";
    if(!alg.order(Tr(), alg.get_unit()))
        throw "Error ast, Tr(A) >= 1";
    Matrix res(row, col);
    for(size_t i = 0; i < row; i++)
    {
        res = res + (*this)^i;
    }
    return res;
}

double tr()
{
    if (row != col)
        throw "Error identity, matrix is not squared";
    double res = alg.get_zero();
    for(size_t i = 0; i < row; i++)
    {
        res = alg.oplus(res, (*this)(i,i));
    }
    return res;
}

double Tr()
{
```

```
    if (row != col)
        throw "Error identity, matrix is not squared";
    double res = alg.get_zero();
    for(size_t i = 0; i < row; i++)
    {
        res = alg.oplus(res, ((*this)^(i+1)).tr());
    }
    return res;
}

Matrix transpose()
{
    Matrix res(col, row);
    for(size_t i = 0; i < row; i++)
    {
        for(size_t j = 0; j < col; j++)
        {
            res(j,i) = (*this)(i,j);
        }
    }
    return res;
}

Matrix pseudo_inverse()
{
    Matrix res(col, row);
    res = transpose();
    for(size_t i = 0; i < res.row; i++)
        for(size_t j = 0; j < res.col; j++)
            res(i,j) = alg.pseudo_inverse(res(i,j));
    return res;
}

double norm()
{
    double res = get_zero();
    for(size_t i = 0; i < row; i++){
        for(size_t j = 0; j < col; j++)
            res = alg.oplus(this->operator ()(i,j), res);
    }
    return res;
}

void out() const
{
    for(size_t i = 0; i < row; i++)
    {
        for(size_t j = 0; j < col; j++)
        {
            std::cout << (*this)(i,j) << "    ";
        }
        std::cout << std::endl;
    }
}
```

```
bool is_row_regular()
{
    for(size_t i = 0; i < row; i++){
        size_t fail_counter = 0;
        for(size_t j = 0; j < col; j++)
        {
            if((*this)(i,j) != alg.get_zero())
                break;
            else
                fail_counter++;
        }
        if(fail_counter == col)
            return false;
    }
    return true;
}

bool is_col_regular()
{
    for(size_t i = 0; i < col; i++){
        size_t fail_counter = 0;
        for(size_t j = 0; j < row; j++)
        {
            if((*this)(i,j) != alg.get_zero())
                break;
            else
                fail_counter++;
        }
        if(fail_counter == row)
            return false;
    }
    return true;
}

bool is_regular()
{
    return is_row_regular() && is_col_regular();
}

bool regular_cols()
{
    bool res = true;
    Matrix<T> b_j;
    for(size_t j = 0; j < col; j++)
    {
        b_j = this->get_col(j);
        if(!b_j.is_row_regular())
            res = false;
    }
    return res;
}

void to_file(std::ofstream& outf)
{
    for(int i = 0; i < row; i++)
```

```
        {
            outf << "\n";
            for(int j=0; j < col; j++)
            {
                if ((*this)(i,j) == get_zero())
                {
                    outf  << " z " ;
                }
                else
                    if ((*this)(i,j) < 0.0)
                    {
                        outf   << (*this)(i,j) << " ";}
                    else
                    {
                        outf  << " " << (*this)(i,j) << " ";
                    }
            }

        }
        outf << "\n";
    }

    Matrix<T> get_col(size_t j)
    {
        Matrix<T> res(row, 1);
        res.Zero();
        if(j >= row)
            return res;
        for(size_t i = 0; i < row; i++)
        {
            res(i,0) = this->operator ()(i,j);
        }
        return res;
    }

    bool is_razl()
    {
        for(size_t i = 0; i < row; i++)
        {
            for(size_t j = 0; j < col; j++)
            {
                bool visited[row];
                if(!dfs(i,j,row, visited))
                {
                    return false;
                }
            }
        }
        return true;
    }


private:
    bool dfs(size_t u, size_t t, size_t n, bool* visited)
    {
```

```
        if(row != col || n != row)
            return false;
        visited[u] = true;
        for(size_t v = 0; v < n; v++)
        {
            if(!visited[v])
                if(dfs(v,t,n,visited))
                    return true;
        }
        return false;
    }
};
```

Appendix C Software implementation of the project management solution problems discussed above

This program is also implemented in the C++ language. The program is written in object-oriented style. The classes used to implement network planning tasks are inherited from the `Task` base class. The tasks themselves are formulated in the form of a specially generated text file. The results are also output to a text file.

C.1 Abstract task description class

The abstract class `Task<T>` is designed to provide reading and writing tasks. As a template element, the inheritor of the `Task<T>` class, described in the previous section, is used. The class has the following public methods:

- `Task(const std::string& _order, const std::string& _solution)` – class constructor. The input is the address string of the input text file `_order` and the class string of the output test file `_solution`, in which the solution is written;
- The `solve()` function solves the problem and writes the solution to the output file

C.1.1 Task input/output format

- Any task starts with a keyword `task`;
- Next, the matrices and vectors that make up the conditions of the problem are written. The beginning of the matrix specification is highlighted with a keyword `rowcol`. The number of rows and columns of the matrix is displayed separately on each line. Then the elements of the matrix are specified. The output of each matrix must end with the word `_rowcol`. An idempotent zero is given by the symbol z.

C.1.2 Class implementation listing

```
template<class T>
class Task
{
```

```
public:
    Task(const std::string& _order, const std::string& _solution)
    {
        order = _order;
        solution = _solution;
    }
    virtual void solve() = 0;
    double get_zero()
    {
        return alg.get_zero();
    }

    double get_unit()
    {
        return alg.get_unit();
    }

protected:
    virtual void read() = 0;
    std::vector<string> read_start(size_t& position)
    {
        std::vector<string> buf_tmp = Task<T>::read_task_file();
        for (int k = 0; k < buf_tmp.size(); k++)
        {
            if (buf_tmp[k] == "z")
                buf_tmp[k] = std::to_string(get_zero());
        }

        // разберём буфер и подготовим данные для класса задач

        for (position = 0; position < buf_tmp.size(); position++)
            if (buf_tmp[position] == "task") break;

        // проверим нашли ли слово "task"
        if (position == buf_tmp.size())
        {
            cout << "Invalid task file format ";
            exit(4);
        }
        return buf_tmp;
    }
    virtual void write() = 0;
    std::vector<string> read_task_file()
    {
        std::vector<string> buf;
        string buf_string;
        std::ifstream tskfile;  //поток ввода задачи
        int i=0;
        tskfile.open(order);
        if (!tskfile.is_open())
        {
            cout << "File does not exist ";
            exit(3);
        }
```

```
//  The task file exists. Let us take it apart
        while (!tskfile.eof())
        {
            tskfile >> buf_string;      // read line from file
            buf.push_back(buf_string); // write it to buffer
        }
        tskfile.close();
        return buf;
    }
    Matrix<T> read_next_matrix_from_position(
    const std::vector<string>& buf_tmp, size_t& position)
    {
        while
        (
            buf_tmp[position] != "rowcol"
            && position < buf_tmp.size()
        )
            position++;
        if (buf_tmp[position] == "rowcol")
        {
            position++;
            int row, col;
            row = std::stoi(buf_tmp[position]);
            position++;
            col = std::stoi(buf_tmp[position]);
            Matrix<T> res(row, col);
            position++;
            while (buf_tmp[position] != "_rowcol")
            {
                for(size_t k = 0; k < row; k++)
                    for(size_t l = 0; l < col; l++)
                    {
                        res(k,l) = std::stod(buf_tmp[position]);
                        position++;
                        if(buf_tmp[position] == "_rowcol")
                            return res;
                    }
            }
            return res;
        }
        else
        {
            Matrix<T> res(0, 0);
            return res;
        }
    }
protected:
    std::string order;
    std::string solution;
    T alg;
};
```

C.2 Project duration minimization class

The `Min_duration_time<T>` class, which is used to solve the problem (3.36), is a subclass of the `Task<T>` class. The class has the following public methods:

- `Min_duration_time(const std::string& _order, const std::string& _solution)` – class constructor. The input is the address string of the input text file `_order` and the class string of the output test file `_solution`, in which the solution is written;
- The `solve()` function solves the problem and writes the solution to the output file

C.2.1 Input/output format

The keyword `task` is written at the beginning of the file. Then, in accordance with the above rules for writing matrices matrices $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ and vectors $\boldsymbol{g}$, $\boldsymbol{f}$ and $\boldsymbol{h}$. Let us give an example of an input file for the network planning problem considered in the example above.

```
task
rowcol
3
3
0 -1 1
0 -1 2
-2 -3 0
_rowcol
rowcol
3
3
4 2 3
3 1 2
2 1 3
_rowcol
rowcol
3
3
-4 -5 -6
z -4 -7
-5 z -4
_rowcol
rowcol
3
1
0
0
0
```

```
_rowcol
rowcol
3
1
4
3
2
_rowcol
rowcol
3
1
6
6
6
_rowcol
```

The output file first displays the minimum of the task (3.36), which is equal to θ, then the range of values for the $\boldsymbol{u}$ vector is given. The vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are given at the end. Unit vectors are denoted as `id_vec`. Here is an output file example:

```
theta = 5
 0
 0
 3
<= u <=
 2
 3
 1
x = G * u
G =
 0  0 -2
 1  0  2
 1  1  0
y = C * G * u
 C * G =
 4  3 5
 3  2 4
 2  1 3
```

C.2.2 Class implementation listing

```
template<class T>
class Min_duration_time: protected Task<T>
{
public:
    Min_duration_time
    (
        const std::string& _order
        , const std::string& _solution
    ):Task<T>(_order, _solution){}

    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
```

```
    B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    C=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    D=Task<T>::read_next_matrix_from_position(buf_tmp,position);

    g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    h=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    f=Task<T>::read_next_matrix_from_position(buf_tmp,position);
}
void write()
{
    std::ofstream solvefile;
    solvefile.open(this->solution);
    solvefile << "theta = " << _theta << std::endl;
    u_left.to_file(solvefile);
    solvefile << "<= u <= " << std::endl;
    u_right.to_file(solvefile);
    solvefile
    << "x = G * u"<< std::endl;
    << "G = "<< std::endl;
    G.to_file(solvefile);
            solvefile
    << "y = C * G * u" << std::endl;
    << "CG = "<< std::endl;
    CG.to_file(solvefile);
    solvefile.close();
}
void solve()
{
    read();
    Matrix<T> DC(B.get_rows(),B.get_cols());
    DC = D * C;
    R = B + DC;
    Matrix<T> sT(B.get_rows(),1);
    sT = f.pseudo_inverse() * C + h.pseudo_inverse();
    Matrix<T> _sT_R_ast_g;
    _sT_R_ast_g = sT * R.ast() * g;
    if
    (
        this->alg.order
        (
            R.ast().Tr()
            , this->get_unit())
        )
        &&
        this->alg.order
        (
            _sT_R_ast_g(0,0)
            , this->get_unit())
        )
    )
    {

       _theta = this->alg.oplus
                (
```

```
                    (C * R).norm()
                    , this->alg.otimes
                          (
                              (sT * R).norm()
                              , (C * g).norm()
                          )
                );
        _theta = this->alg.oplus
                (
                    _theta
                    , this->alg.otimes
                          (
                              sT.norm()
                              , (C * g).norm()
                          )
                );

         _theta = this->alg.oplus
                (
                    _theta
                    , this->alg.otimes
                          (
                              sT.norm()
                              , (C * R * g).norm()
                          )
                );

        _theta = this->alg.oplus
                (
                    _theta
                    , this->alg.otimes
                          (
                              (sT * R).norm()
                              , (C * R * g).norm()
                          )
                );
        Matrix<T> id_vec(B.get_rows(),1);
        id_vec.Identity_vec();
        G
        =
        R.ast
        +
        _theta *
        (
            (id_vec * id_vec.transpose()) * C
            + R * (id_vec * id_vec.transpose()) * C
            + (id_vec * id_vec.transpose()) * C * R
        );
        CG = C * G;
        u_left = g;
        u_right =
         (sT * G).pseudo_inverse();
        write();
    }
}
```

```
 protected:
    Matrix<T> B;
    Matrix<T> C;
    Matrix<T> D;
    Matrix<T> R;

    Matrix<T> g;
    Matrix<T> f;
    Matrix<T> h;

    Matrix<T> u;
    Matrix<T> u_left, u_right;
    Matrix<T> G;
    Matrix<T> CG;

    double _theta;
};
```

C.3 Maximum deviation minimization class

The `Min_max_deviation<T>` class implements an algorithm for solving the problem (4.14). The class is an inheritor of the `Task<T>` class. Let us describe its public methods:

- `Min_max_deviation(const std::string& _order, const std::string& _solution)` – class constructor. The input is the address string of the input text file `_order` and the class string of the output test file `_solution`, in which the solution is written;
- The `solve()` function solves the problem and writes the solution to the output file

C.3.1 Input/output format

The keyword `task` is written at the beginning of the file. Further, in accordance with the previously introduced rules for writing matrices, matrices $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ and vectors $\boldsymbol{g}$, $\boldsymbol{f}$, $\boldsymbol{h}$ and $\boldsymbol{p}$. Let us give an example of an input file for the network planning problem presented in the example above.

```
task
rowcol
3
3
0 -1 1
0 -1 2
```

```
-2 -3 0
_rowcol
rowcol
3
3
4 2 3
3 1 2
2 1 3
_rowcol
rowcol
3
3
-4 -5 -6
z -4 -7
-5 z -4
_rowcol
rowcol
3
1
0
0
0
_rowcol
rowcol
3
1
4
3
2
_rowcol
rowcol
3
1
6
6
6
_rowcol
rowcol
3
1
2
2
1
_rowcol
```

In the output file, the minimum of the task (4.14) equal to θ, is displayed first, then the range of values for the $\boldsymbol{u}$ vector is given. At the end are the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Unit vectors are denoted as `id_vec`. Here is an example output file:

```
theta = 1
 1
 1
 0
<= u <=
 2
 3
```

```
 1
x = G * u
G =
 0  -1 1
 1  0  2
 -1  -2  0
y = C * G * u
 C * G =
 4  2 3
 3  1 2
 2  1 3
```

C.3.2 Class implementation listing

```
template<class T>
class Min_max_deviation: protected Task<T>
{
public:
    Min_max_deviation
    (
        const std::string& _order
        , const std::string& _solution
    ):Task<T>(_order, _solution){}

    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        C=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        D=Task<T>::read_next_matrix_from_position(buf_tmp,position);

        g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        h=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        f=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        p=Task<T>::read_next_matrix_from_position(buf_tmp,position);

    }
    void write()
    {
        std::ofstream solvefile;
        solvefile.open(this->solution);
        solvefile << "theta = " << _theta << std::endl;
        u_left.to_file(solvefile);
        solvefile << "<= u <= " << std::endl;
        u_right.to_file(solvefile);
        solvefile
        << "x = R^(ast) * u"<< std::endl;
        << "R^(ast) = "<< std::endl;
        R_ast.to_file(solvefile);
                solvefile
        << "y = C * R^(ast) * u" << std::endl;
        << "CG = "<< std::endl;
        CR_ast.to_file(solvefile);
        solvefile.close();
```

```
}
void solve()
{
    read();
    Matrix<T> DC(B.get_rows(),B.get_cols());
    DC = D * C;
    R = B + DC;
    Matrix<T> sT(B.get_rows(),1);
    sT = f.pseudo_inverse() * C + h.pseudo_inverse();
    Matrix<T> _sT_R_ast_g;
    _sT_R_ast_g = sT * R.ast() * g;
    if
    (
        this->alg.order
        (
            R.ast().Tr()
            , this->get_unit())
        )
        &&
        this->alg.order
        (
            _sT_R_ast_g(0,0)
            , this->get_unit())
        )
    )
    {
       _theta
       =
       (
           (
               p.pseudo_inverse() * R.ast() * p)^(1/2)
           )
           +
           sT * R.ast() * p
           +
           p.pseudo_inverse() * R.ast() * g
       )(0,0);

       Matrix<T> id_vec(B.get_rows(),1);
       id_vec.Identity_vec();
       G
       =
       R.ast
       +
       _theta *
       (
           (id_vec * id_vec.transpose()) * C
           + R * (id_vec * id_vec.transpose()) * C
           + (id_vec * id_vec.transpose()) * C * R
       );
       R_ast = R.ast();
       CR_ast = C * R_ast;
       u_left = g + this->alg.pseudo_inverse(_theta) * p;
       u_right =
       (
```

```
                (
                    sT + this->alg.pseudo_inverse(_theta)
                         * p.pseudo_inverse()
                ) * R_ast
            ).pseudo_inverse();
            write();
        }
    }
 protected:
    Matrix<T> B;
    Matrix<T> C;
    Matrix<T> D;
    Matrix<T> R;

    Matrix<T> g;
    Matrix<T> f;
    Matrix<T> h;

    Matrix<T> u;
    Matrix<T> u_left, u_right;
    Matrix<T> R_ast;
    Matrix<T> CR_ast;

    double _theta;
};
```

C.4 Completion time spread minimization class

The `Min_fin_time<T>` class used to solve problem (5.13), inherited from class `Task<T>`, described in the previous section. The following public methods are implemented in the class:

- `Min_fin_time(const std::string&_order, const std::string&_solution)` – class constructor. The input is the address string of the input text file `_order` and the class string of the output test file `_solution`, in which the solution is written;
- The `solve()` function solves the problem and writes the solution to the output file

C.4.1 Input/output format

The keyword `task` is written at the beginning of the file. Then, in accordance with the above rules for writing matrices, matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and vectors $\boldsymbol{g}$, $\boldsymbol{h}$, $\boldsymbol{f}$ are

given. Let us give an example of an input file for the network planning problem considered in the example above.

```
task
rowcol
3
3
4 0 z
2 3 1
1 1 3
_rowcol
rowcol
3
3
z -2 1
0 z 2
-1 z z
_rowcol
rowcol
3
1
1
0
0
_rowcol
rowcol
3
1
5
4
4
_rowcol
rowcol
3
1
8
8
10
_rowcol
```

The output file first displays the solution to the (5.13) problem, which is equal to Δ, then the limit on α is written. At the end are the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ on which the solution is reached. Here is an output file example:

```
Delta = 2
5 <= alpha <= 7
x = alpha *
-4
-3
-5
y = alpha *
 0
 0
-2
```

C.4.2 Class implementation listing

```
template<class T>
class Min_fin_time: protected Task<T>
{
public:
    Min_fin_time
    (
        const std::string& _order, const std::string& _solution
    ):Task<T>(_order, _solution){}

    void solve()
    {
        read();
        Matrix<T> id_vec(A.get_rows(),1);
        id_vec.Identity_vec();
        Matrix<T> alpha_left;
        Matrix<T> alpha_right;
        Matrix<T> delta;

        alpha_left = (id_vec.transpose() * A) * (B.ast()) * g;
        alpha_left.out();
        alpha_right =
          (((f.pseudo_inverse()* A) + h.pseudo_inverse())
          * B.ast()
          * (id_vec.transpose() * A * B.ast()).pseudo_inverse()
          ).pseudo_inverse();
        _alpha_left = alpha_left(0,0);
        _alpha_right = alpha_right(0,0);
        if(_alpha_left <= _alpha_right)
        {
            delta =
            (
             A * B.ast() *
              (
               id_vec.transpose() * A * B.ast()
              ).pseudo_inverse()
            ).pseudo_inverse() * id_vec;
            _delta = delta(0,0);

            x =
             B.ast() *
             (
              id_vec.transpose() * A * B.ast()
             ).pseudo_inverse();
            y = A * x;
            write();
        }
    }

protected:
    void read()
    {
        size_t position;
```

```
        auto buf_tmp = Task<T>::read_start(position);
        A=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        h=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        f=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    }

    void write()
    {
        std::ofstream solvefile;
        solvefile.open(this->solution);
        solvefile << "Delta = " << _delta << std::endl;
        solvefile << _alpha_left
                  << "<= alpha <= "
                  << _alpha_right << std::endl;
        solvefile << "x = alpha * ";
        x.to_file(solvefile);
        solvefile << "y = alpha * ";
        y.to_file(solvefile);
        solvefile.close();
    }
 protected:
    Matrix<T> A;
    Matrix<T> B;
    Matrix<T> f;
    Matrix<T> g;
    Matrix<T> h;
    Matrix<T> x;
    Matrix<T> y;
    double _alpha_left, _alpha_right, _delta;
};
```

C.5 Completion time spread maximization class

The class `Max_fin_time<T>`, which is used to solve the problem(6.12), inherited from class `Task<T>`, described in the previous section. The following public methods are implemented in the class:

- `Max_fin_time(const std::string& _order, const std::string& _solution)` – class constructor. The input is the address string of the input text file `_order` and the class string of the output test file `_solution`, in which the solution is written;

- The `solve()` function solves the problem and writes the solution to the output file

C.5.1 Input/output format

The keyword `task` is written at the beginning of the file. Then, in accordance with the above rules for writing matrices, the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and the vector $\boldsymbol{h}$ are given. Let us give an example of an input file for the network planning problem considered in the example above.

```
task
rowcol
3
3
4 1 1
2 2 0
0 1 3
_rowcol
rowcol
3
3
z -2 1
0 z 2
-1 z z
_rowcol
rowcol
3
1
5
4
4
_rowcol
```

The output file first displays the solution to the (6.12) problem, which is equal to Δ. Then the Kleene matrix $\boldsymbol{B}^*$ and the matrix $\boldsymbol{D}$ are given. Next, the range of values of the scalar $\boldsymbol{alpha}$ is specified, values of indices k, s and range of values for the vector $\boldsymbol{u}$ At the end are the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Unit vectors are denoted by the word `id_vec`. Let us write output file an example:

```
Delta = 2

B_ast =
 0 -2  1
 1  0  2
-1 -3  0

D =
 4  2  5
 3  2  4
 2  1  3
alpha <= 3, k = 3, s =  3
u_3= alpha *-3
u_1 <= alpha *-2
u_2 <= alpha *-1
```

```
x = B_ast * u
y = D * u
```

C.5.2 Class implementation listing

```
template<class T>
class Max_fin_time: protected Task<T>
{
public:
    Max_fin_time
    (
        const std::string& _order, const std::string& _solution
    ):Task<T>(_order, _solution){}
    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
        A=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        h=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    }
    void write()
    {
        std::ofstream solvefile;
        solvefile.open(this->solution);
        solvefile << "Delta = " << delta << std::endl;
        solvefile << "B_ast =" << std::endl;
        B_ast.to_file(solvefile);
        solvefile << "D =" << std::endl;
        D.to_file(solvefile);
        solvefile << "alpha <= "
                  << alpha
                  << ", k = "
                  << k + 1 <<", s =  " << s + 1 << std::endl;
        solvefile << "u_"
                  << k + 1
                  << "= alpha *"
                  << this->alg.power(D(s,k), -1) << std::endl;
        for(size_t i = 0; i < D.get_rows(); i++)
            if(i != k)
        {
                solvefile << "u_"
                          << i + 1
                          << " <= "
                          << "alpha *"
                          << this->alg.power(D(s,i), -1)
                          << std::endl;
        }
        solvefile << "x = " << "B_ast * u" << std::endl;
        solvefile << "y = " << "D * u" << std::endl;
        solvefile.close();
    }
    void solve()
    {
```

```
read();
B_ast = B.ast();
D = A * B_ast;
if(A.is_regular() &&
        this->alg.order(B.Tr(), this->get_unit()) &&
        D.regular_cols())
{
   Matrix<T> _delta;
   Matrix<T> id_vec(B.get_rows(),1);
   id_vec.Identity_vec();
   _delta = id_vec.transpose()
            * D * D.pseudo_inverse() * id_vec;
   delta = _delta(0,0);
   k = 0;
   s = 0;
   double val = this->alg.get_zero();
   for(size_t j = 0; j < D.get_cols(); j++)
   {
       Matrix<T> d_ast_j;
       d_ast_j = D.get_col(j);
       Matrix<T> id_d_di_id;
       id_d_di_id =
           id_vec.transpose()
           * d_ast_j * d_ast_j.pseudo_inverse() * id_vec;
       if(this->alg.order(val, id_d_di_id(0,0)))
       {
           k = j;
           val = id_d_di_id(0,0);
       }
   }
   val = this->alg.get_zero();
   for(size_t j = 0; j < D.get_cols(); j++)
   {
       if(this->alg.order(val, this->alg.power(D(j,k), -1)))
       {
           s = j;
           val = this->alg.power(D(j,k), -1);
       }
   }
   val = this->alg.get_zero();
   Matrix<T> d_0;
   d_0 = D.get_col(0);
   Matrix<T> buff;
   buff = (h.pseudo_inverse() * d_0).pseudo_inverse();
   buff = buff * D(s,0);
   alpha = buff(0,0);
   for(size_t j = 1; j < D.get_cols(); j++)
   {
       Matrix<T> d_j;
       d_j = D.get_col(j);
       buff = (h.pseudo_inverse() * d_j).pseudo_inverse();
       buff = buff * D(s,j);
```

```
                if(this->alg.order(buff(0,0), alpha))
                {
                    alpha = buff(0,0);
                }
            }
            write();
        }
    }
 protected:
    Matrix<T> A;
    Matrix<T> B;
    Matrix<T> h;
    Matrix<T> x;
    Matrix<T> y;
    double delta;
    double alpha;
    size_t k,s;
    Matrix<T> B_ast;
    Matrix<T> D;
};
```

C.6 Spread of the jobs start time minimization class

The class `Min_start_time<T>`, which is used to solve problem (7.8), inherited from class `Task<T>`, described in the previous section. The following public methods are implemented in the class:

- `Min_start_time(const std::string&_order, const std::string&_solution)` – class constructor. The input is the address string of the input text file `_order` and the class string of the output test file `_solution`, in which the solution is written;
- The `solve()` function solves the problem and writes the solution to the output file

C.6.1 Input/output format

The keyword `task` is written at the beginning of the file. Then, in accordance with the above rules for writing matrices, the matrix $\boldsymbol{B}$ and the vectors $\boldsymbol{g}$ and $\boldsymbol{h}$ are given. Let us give an example of an input file for the network planning problem considered in the example above.

```
task
rowcol
3
```

```
3
z 0 -2
-1 z -2
1 1 z
_rowcol
rowcol
3
1
0
0
3
_rowcol
rowcol
3
1
2
1
5
_rowcol
```

The output file first displays the minimum of problem (7.8), which is equal to θ, then the range of values for the $\boldsymbol{u}$ vector is given. The vector $\boldsymbol{x}$ is given at the end. Unit vectors are denoted as `id_vec`. Here is an output file example:

```
theta = 2
 0
 0
 3
<= u <=
 2
 1
 3
x = (theta^{-1} sum(B_i * id_vec * id_vec^{T} B_j) oplus B^{*} * u
 0  0 -2
-1  0 -2
 1  1  0
```

C.6.2 Class implementation listing

```
template<class T>
class Min_start_time: protected Task<T>
{
public:
    Min_start_time
    (
        const std::string& _order, const std::string& _solution
    ):Task<T>(_order, _solution){}
    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        l=Task<T>::read_next_matrix_from_position(buf_tmp,position);
```

```
}
void write()
{
    std::ofstream solvefile;
    solvefile.open(this->solution);
    solvefile << "theta = " << _theta << std::endl;
    u_left.to_file(solvefile);
    solvefile << "<= u <= " << std::endl;
    u_right.to_file(solvefile);
    solvefile <<
    "x = (theta^{-1} sum(B_i * id_vec * id_vec^{T} B_j)
     oplus B^{*} * u";
    theta_min_one_B_B_Bi_ast.to_file(solvefile);
    solvefile.close();
}
void solve()
{
    read();
    Matrix<T> id_vec(B.get_rows(),1);
    id_vec.Identity_vec();
    Matrix<T> _l_B_g_ast;
    _l_B_g_ast = l.pseudo_inverse() * B.ast() * g;
    if(
    this->alg.order(
    this->alg.oplus(B.Tr(), _l_B_g_ast(0,0)), this->get_unit()))
    {
       size_t n = B.get_rows();
       auto powers = get_all_pairs(n-2);
       double elem = this->get_zero();
       for(size_t i = 0; i < powers.size(); i++)
       {
          double lB =
           (l.pseudo_inverse() * (B^(powers[i][0]))).norm();
          double Bg = ((B^(powers[i][1])) * g).norm();
          elem=this->alg.oplus(elem, this->alg.otimes(lB, Bg));
       }
       _theta = this->alg.oplus(elem, B.ast().norm());
       for(size_t i = 0; i < powers.size(); i++)
       {
          theta_min_one_B_B_Bi_ast =
            (B^powers[i][0]) *
              (id_vec * id_vec.transpose()) * (B^powers[i][1]);
       }
       theta_min_one_B_B_Bi_ast =
          theta_min_one_B_B_Bi_ast * this->alg.power(_theta,-1);
       theta_min_one_B_B_Bi_ast =
          theta_min_one_B_B_Bi_ast + B.ast();

       u_left = g;
       u_right =
       (
           l.pseudo_inverse() * theta_min_one_B_B_Bi_ast
       ).pseudo_inverse();
```

```
            write();
        }
    }
 protected:
    Matrix<T> B;
    Matrix<T> g;
    Matrix<T> l;
    Matrix<T> u;
    Matrix<T> u_left, u_right;
    Matrix<T> theta_min_one_B_B_Bi_ast;
    double _theta;
};
```

C.7 Spread of the jobs start time minimization class

The class `Max_start_time<T>`, which is used to solve problem (8.15), inherited from class `Task<T>`, described in the previous section. The following public methods are implemented in the class:

- `Max_start_time(const std::string&_order, const std::string& solution)` – class constructor. The input is the address string of the input text file `_order` and the class string of the output test file `_solution`, in which the solution is written;

- The `solve()` function solves the problem and writes the solution to the output file

C.7.1 Input/output format

The keyword `task` is written at the beginning of the file. Then, in accordance with the above rules for writing matrices, the matrix $\boldsymbol{B}$ and the vector $\boldsymbol{g}$ are given. Let us give an example of an input file for the network planning problem considered in the example above.

```
task
rowcol
3
3
z -2 1
0 z 2
-1 z z
_rowcol
rowcol
3
1
```

```
2
0
0
_rowcol
```

The output file first displays the solution to the (8.15) problem, which is equal to Δ. Then the Kleene matrix $\boldsymbol{B}^*$ is given. Next, the range of values for α is set, values of indices k, s and range of values for the vector $\boldsymbol{u}$ The vector $\boldsymbol{x}$ is given at the end. Unit vectors are denoted as `id_vec`. Let us write an output file example:

```
Delta = 3
B_ast =
 0 -2  1
 1  0  2
-1 -3  0
alpha >= 1, k = 2, s =  3
u_2= alpha *3
2 <= u_1 <= alpha *1
0 <= u_3 <= alpha *-0
x = B_ast * u
```

C.7.2 Class implementation listing

```
template<class T>
class Max_start_time: protected Task<T>
{
public:
    Max_start_time
    (
        const std::string& _order, const std::string& _solution
    ):Task<T>(_order, _solution){}
    void read()
    {
        size_t position;
        auto buf_tmp = Task<T>::read_start(position);
        B=Task<T>::read_next_matrix_from_position(buf_tmp,position);
        g=Task<T>::read_next_matrix_from_position(buf_tmp,position);
    }
    void write()
    {
        std::ofstream solvefile;
        solvefile.open(this->solution);
        solvefile << "Delta = " << delta << std::endl;
        solvefile << "B_ast =" << std::endl;
        B_ast.to_file(solvefile);
        solvefile << "alpha >= "
                  << alpha_bound
                  << ", k = "
                  << k + 1
                  <<", s =  "
                  << s + 1
                  << std::endl;
        solvefile << "u_"
```

```
                << k + 1
                << "= alpha *"
                << this->alg.power(B_ast(s,k), -1)
                << std::endl;
    for(size_t i = 0; i < B_ast.get_rows(); i++)
        if(i != k)
    {
            solvefile << g(i,0)
                      << " <= "
                      << "u_"
                      << i + 1
                      << " <= "
                      << "alpha *"
                      << this->alg.power(B_ast(s,i), -1)
                      << std::endl;
    }
    solvefile << "x = "
              << "B_ast * u"
              << std::endl;
    solvefile.close();
}
void solve()
{
    read();
    if(
       this->alg.order
        (B.Tr(), this->get_unit())
       &&
       !B.is_razl() && B.is_quadr()
      )
    {
       Matrix<T> _delta;
       Matrix<T> id_vec(B.get_rows(),1);
       id_vec.Identity_vec();
       B_ast = B.ast();
       _delta =
        id_vec.transpose()
          * B_ast * B_ast.pseudo_inverse() * id_vec;
       delta = _delta(0,0);
       k = 0;
       s = 0;
       double val = this->alg.get_zero();
       for(size_t j = 0; j < B_ast.get_cols(); j++)
       {
           Matrix<T> b_ast_j;
           b_ast_j = B_ast.get_col(j);
           Matrix<T> id_b_bi_id;
           id_b_bi_id
            = id_vec.transpose()
            * b_ast_j * b_ast_j.pseudo_inverse() * id_vec;
           if(this->alg.order(val, id_b_bi_id(0,0)))
           {
               k = j;
               val = id_b_bi_id(0,0);
```

```
                }
            }
            val = this->alg.get_zero();
            for(size_t j = 0; j < B_ast.get_cols(); j++)
            {
                if
                (
                    this->alg.order
                    (
                        val, this->alg.power(B_ast(j,k), -1)
                    )
                )
                {
                    s = j;
                    val = this->alg.power(B_ast(j,k), -1);
                }
            }
            val = this->alg.get_zero();

            for(size_t j = 0; j < B_ast.get_cols(); j++)
            {
                if
                (
                    this->alg.order
                    (
                        val, this->alg.otimes(g(j,0),B_ast(s,j))
                    )
                )
                {
                    val = this->alg.otimes(g(j,0),B_ast(s,j));
                }
            }
            alpha_bound = val;
            write();
        }
    }
 protected:
    Matrix<T> B;
    Matrix<T> g;
    double delta;
    size_t k,s;
    Matrix<T> B_ast;
    double alpha_bound;

};
```