Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

«Решение задачи нахождения связного подграфа максимального веса с учетом весов ребер»

Автор: Лобода Александр Александрович _____

Направление подготовки (специальность): Прикладная математика и информатика

Квалификация: бакалавр

Руководитель: Сергушичев А.А., магистр _____

К защите допустить

Зав. кафедрой Васильев В.Н., проф., д.т.н. _____

“___”_____ 20____г.

Санкт-Петербург, 2016 г.

Студент Лобода А.А. **Группа** М3439 **Кафедра** КТ **Факультет** ИТиП

Направленность (профиль), специализация Математические модели и алгоритмы в разработке программного обеспечения

Квалификационная работа выполнена с оценкой _____

Дата защиты “____”_____20 ____г.

Секретарь ГЭК _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет *Информационных Технологий и Программирования*
Кафедра *Компьютерных технологий* **Группа** *M3439*
Направление (специальность) *Прикладная математика и информатика*
Квалификация (степень) *Бакалавр прикладной математики и информатики*

З А Д А Н И Е
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студент *Лобода А.А.*
Руководитель *Сергушичев А.А., аспирант кафедры КТ, программист университета ИТМО*

1. Наименование темы *решение задачи нахождения связного подграфа максимального веса с учетом весов ребер*

2. Срок сдачи студентом законченной работы *31 мая 2016 г.*

3. Техническое задание и исходные данные к работе
В рамках работы необходимо создать солвер для обобщенной задачи поиска подграфа максимального веса, решающего возникающие при анализе представленности в сетях биохимических реакций за приемлимое время

4. Содержание выпускной работы (перечень подлежащих разработке вопросов)
1. Обзор предметной области
2. Разработка алгоритма решения рассматриваемой задачи
3. Реализация алгоритма поиска подграфа максимального веса
4. Проведение сриавнительного тестирования с известными солверами для близких задач

5. Перечень графического материала (с указанием обязательного материала)
Не предусмотрено

6. Исходные материалы и пособия

1. El-Kebir M., Klau G. W. Solving the Maximum-Weight Connected Subgraph Problem to Optimality. — 2014. — eprint: 1409.5308.

2.Álvarez-Miranda E., Ljubić I., Mutzel P. The maximum weight connected subgraph problem // Facets of Combinatorial Optimization. — Springer, 2013. — С. 245—270.

3.Haouari M., Maculan N., Mrad M. Enhanced compact models for the connected subgraph problem and for the shortest path problem in digraphs with negative cycles // Computers & Operations Research. — 2013. — Т. 40, No 10. — С. 2485—2492.

4.GAM: a web-service for integrated transcriptional and metabolic network analysis. / A. Sergushichev [и др.] // Nucleic acids research. — 2016.

7. Консультанты по работе с указанием относящихся к ним разделов работы

Раздел	Консультант	Подпись, дата	
		Задание выдал	Задание принял
Экономика и организация производства			
Технология приборостроения			
Безопасность жизнедеятельности и экология			

КАЛЕНДАРНЫЙ ПЛАН

№ № п/п	Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Примечание
1	Изучение предметной области	2.11.2015	
2	Изучение ближайших аналогов	1.12.2015	
3	Разработка правил препроцессинга	29.12.2015	
4	Разработка декомпозиции	16.02.2016	
5	Разработка линейных ограничений	2.03.2016	
6	Разработка ограничений нарушения симметрий	4.04.2016	
7	Изучение IBM ILOG CPLEX	15.04.2016	
8	Реализация солвера на языке Java	3.05.2016	
9	Проведение сравнительного тестирования	13.05.2016	
10	Написание пояснительной записки	31.05.2016	

8. Дата выдачи задания *1 сентября 2015 г.*

Руководитель _____

Задание принял к исполнению _____

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

АННОТАЦИЯ
ПО ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Студент Лобода А.А.
Факультет информационных технологий и программирования
Кафедра компьютерных технологий Группа М3439
Направление (специальность) Прикладная математика и информатика
Квалификация (степень) Бакалавр прикладной математики и информатики
Наименование темы Решение задачи нахождения связного подграфа максимального веса с учетом весов ребер
Руководитель Сергушичев А.А., аспирант кафедры КТ

КРАТКОЕ СОДЕРЖАНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ И ОСНОВНЫЕ ВЫВОДЫ

объем 34 стр., **графический материал** – стр., **библиография** 12 наим.

Направление и задача исследований

Целью данной работы является разработка практического солвера для обобщенной задачи поиска связного подграфа максимального веса, который бы решал задачи, возникающие при анализе представленности в сетях биохимических реакций

Проектная или исследовательская часть (с указанием основных методов исследований, расчетов и результатов)

Были изучены существующие решения для рассматриваемой и смежных задач, были изучены методы, применяемые при решении подобных задач, разработан практический солвер, позволяющий решать возникающие на практике задачи за приемлемое время, а также проведено сравнение времени работы с ближайшими аналогами

Экономическая часть (какие использованы методики, экономическая эффективность результатов)

Данная работа не предполагает извлечения прямой экономической выгоды из полученных результатов.

Характеристика вопросов экологии, техники безопасности и др.

Результатом работы является программный продукт, не нарушающий требования экологической безопасности.

Является ли работа продолжением курсовых проектов (работ), есть ли публикации

Работа является продолжением курсового проекта, в котором рассматривалось решение задачи сведением к задаче удовлетворения ограничений. Результаты данной работы являются составной частью веб-сервиса, описанного в GAM: a web-service for integrated transcriptional and metabolic network analysis. / A. Sergushichev [и др.] // Nucleic acids research. — 2016.

Практическая ценность работы. Рекомендации по внедрению

Алгоритм реализован и является частью веб-сервиса по анализу представленности в сетях биологических реакций

Выпускник _____

Научный руководитель _____

«____» _____ 2016 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Задача поиска связного подграфа максимального веса в различных своих вариациях естественно возникает при работе в области компьютерного зрения [1, 2], при проектировании различного рода сетей [3], а также в вычислительной биологии [4]. В этой работе рассматривается обобщенный вариант задачи, решение которого важно для анализа представленности в сетях биохимических реакций [5].

Для нахождения решения в задачах данного типа используется подход, основанный на сведении задачи к задаче смешанного целочисленного программирования, так как для ее решения существуют эффективные на практике солверы (решатели).

Целью данной работы является разработка практического солвера для обобщенной задачи поиска связного подграфа максимального веса, который бы решал задачи, возникающие при анализе представленности в сетях биохимических реакций.

Реализованный алгоритм является составной частью веб-сервиса по анализу представленности в сетях биохимических реакций, описанному в [5].

Глава 1 посвящена обзору существующих солверов для разных вариантов задачи поиска связного подграфа максимального веса, подхода, используемого в современных солверах для решения задачи смешанного целочисленного программирования, и техник, часто используемых для увеличения производительности. В этой главе также приведены основные определения. В главе 2 описаны правила для упрощения экземпляров задачи, непосредственно приведено сведение и доказана его корректность. В главе 3 приведены результаты сравнительного тестирования разработанного солвера с ближайшими аналогами. В заключении дано описание полученных в ходе данной работы результатов.

ГЛАВА 1. ОБЗОР

В этом разделе мы приводим необходимые для рассуждений в этой работе определения, а также проводим обзор существующих солверов для задач поиска связного подграфа максимального веса, а также рассматриваем подход, который используется при решении задачи смешанного целочисленного программирования.

1.1. Основные определения

В этой работе мы рассматриваем задачу поиска связного подграфа максимального веса, для которой существует слегка различные формулировки. В наиболее используемом определении задачи только вершины взвешены [6, 7]. В этой работе мы рассматриваем задачу, в которой ребра также взвешены [8]. Чтобы избежать неоднозначности, первую мы будем называть вершинно-взвешенной, а последнюю — обобщенной.

Целью задачи является нахождение в данном графе связного подграфа с максимальной суммой весов. Поскольку подграф связен, мы можем рассматривать компоненты связности независимо. Поэтому ниже мы будем предполагать, что исходный граф связен.

Для начала дадим определение вершинно-взвешенной задачи поиска связного подграфа максимального веса.

Определение 1. Дан связный неориентированный граф $G = (V, E)$ и весовая функция $\omega_v : V \to \mathbb{R}$, *вершинно-взвешенная задача поиска связного подграфа максимального веса* состоит в отыскании связного подграфа $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ с максимальным суммарным весом

$$\Omega(\widetilde{G}) = \sum_{v \in \widetilde{V}} \omega(v) \to max$$

Затем определим обобщенный вариант этой задачи, в которой вершины и ребра могут быть взвешены.

Определение 2. Дан связный неориентированный граф $G = (V, E)$ и весовая функция $\omega : (V \cup E) \to \mathbb{R}$, *обобщенная задача нахождения связного подграфа максимального веса* состоит в отыскании связного подграфа $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ с максимальным суммарным весом

$$\Omega(\widetilde{G}) = \sum_{v \in \widetilde{V}} \omega(v) + \sum_{e \in \widetilde{E}} \omega(e) \to max$$

Теперь определим корневой вариант обобщенной задачи, в которой требуется, чтобы одна из вершин была в решении. Эта задача используется как вспомогательная при решении обобщенной задачи.

Определение 3. Дан связный неориентированный граф $G = (V, E)$, весовая функция $\omega : (V \cup E) \to \mathbb{R}$ и вершина $r \in V$, называемая корнем. *Корневая обобщенная задача поиска связного подграфа максимального веса* состоит в отыскании связного подграфа $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ такого, что $r \in \widetilde{V}$ и

$$\Omega(\widetilde{G}) = \sum_{v \in \widetilde{V}} \omega(v) + \sum_{e \in \widetilde{E}} \omega(e) \to max$$

El-Kebir и др. в [7] показали, что вершинно-взвешенная задача *NP*-трудна. Поскольку вершинно-взвешенная задача — частный случай обобщенной, то обобщенная задача также *NP*-трудна. Корневая задача тоже *NP*-трудна, поскольку обобщенная задача может быть решена посредством решения $|V|$ корневых задач, используя каждый раз новую вершину исходного графа в качестве корня.

Наконец, в работе мы будем использовать n как краткое обозначение для числа вершин $|V|$ графа G.

Введем задачу смешанного целочисленного (линейного) программирования.

Определение 4. Пусть x_i — целочисленные переменные, принимающие неотрицательные значения, пусть y_j — вещественные переменные, принимающие неотрицательные значения. Пусть также c_x, c_y — векторы вещественных коэффициентов целевой функции для целочисленных и вещественных переменных соответственно, а A_x, A_y — матрицы вещественных коэффициентов для ограничений. Кроме того, пусть b — вектор-столбец вещественных коэффициентов. Тогда, *задача смешанного целочисленного линейного программирования* заключается в отыскании значений для векторов x и y, при которых целевая функция

$$c_x x + c_y y \to max$$

и выполняется следующее ограничение: $A_x x + A_y y$ покомпонентно не больше b.

Задачи в которых целочисленные переменные могут принимать значения только 0 или 1 называются бинарными.

1.2. Метод ветвей и сечений

В этом параграфе описывается метод, который используется в современных солверах, применяемых для решения задачи смешанного целочисленного программирования, а также отмечаются особенности используемого нами солвера *IBM ILOG CPLEX*.

Метод ветвей и сечений является результатом объединения двух алгоритмов решения задачи смешанного целочисленного программирования, а именно метода ветвей и границ и метода секущих плоскостей. Далее мы опишем принципы работы каждого из них и то, как они объединены в *IBM ILOG CPLEX*.

Оба этих метода используют тот факт, что снятие ограничения целочисленности с переменных преобразует задачу в задачу линейного программирования. Для решения этой задачи существует алгоритм, работающий за полиномиальное время. Однако чаще используется симплекс-метод, хорошо показывающий себя на практике, но неполиномиальный в общем случае.

Определение 5. *Линейным приближением* задачи смешанного целочисленного программирования называется задача, возникающая при удалении всех ограничений на целочисленность переменных из исходной задачи.

1.2.1. Метод ветвей и границ

Метод ветвей и границ основан на подходе «разделяй и властвуй». Рассмотрим произвольную задачу смешанного целочисленного программирования, рассмотрим решение ее линейного приближения. Если в решении для какой-то из целочисленных переменных, скажем z, получилось нецелое значение, то есть $\lfloor z \rfloor = c_1$, $\lceil z \rceil = c_2$ и $c_1 \neq c_2$, то в таком случае мы можем разбить задачу на две подзадачи:

1. с дополнительным ограничением $z \leqslant c_1$,
2. с дополнительным ограничением $z \geqslant c_2$.

Эта операция называется *ветвлением*. Возникшие подзадачи являются задачами смешанного целочисленного программирования и решаются аналогично. Кроме того, оптимальное решение для подзадачи является также решением исходной задачи, но, возможно, неоптимальным.

Не имеет смысла осуществлять ветвление в следующих ситуациях:

1. линейное приближение не имеет решения, так как в этом случае и исходная подзадача не может иметь решения;
2. все необходимые переменные уже целочисленны, в данном случае мы нашли оптимальное решение подзадачи;
3. если значение целевой функции решения линейного приближения не больше решения какой-то из уже рассмотренных подзадач. Это значит, что данная подзадача не может иметь решения лучше, чем мы уже нашли в другой подзадаче, поэтому новое решение получится заведомо хуже.

Работа алгоритма может быть отображена в виде дерева подзадач:

— в корне дерева находится исходная задача;
— если в результате рассмотрения подзадачи произведено ветвление, то соответствующая вершина имеет двух детей-подзадач;
— в листьях могут быть пометки: неразрешимая подзадача, отсеченная подзадача или нерассмотренная подзадача.

Пример такого дерева для пояснения работы алгоритма приведен на рисунке 1.

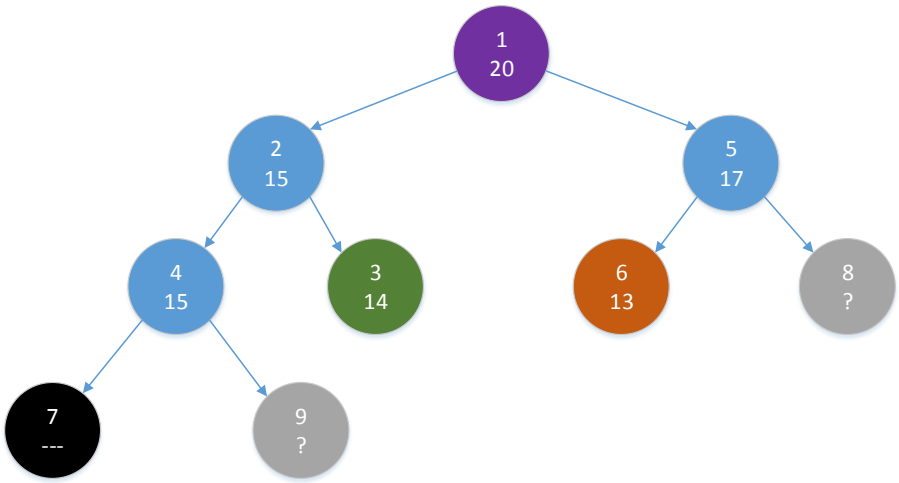

Рисунок 1 – Пример дерева подзадач алгоритма ветвей и границ, первое число в вершине — порядковый номер рассмотрения вершины, второе — значение целевой функции линейного приближения. Фиолетовая вершина — исходная задача, синие — подзадачи, после рассмотрения которых произошло ветвление, зеленая — целочисленное решение, оранжевая — подзадача, отсеченная на основании существования лучшего решения, черная — неразрешимая подзадача, серые — еще не рассмотренные подзадачи

1.2.2. Метод секущих плоскостей

Метод секущих плоскостей основан на добавлении дополнительных ограничений в формулировку задачи смешанного целочисленного программирования. Алгоритм состоит из двух этапов:

1. решение линейного приближения задачи,
2. добавление ограничений в задачу, используя значения переменных решения линейного приближения.

Эти этапы выполняются поочередно до тех пор, пока все целочисленные переменные не будут иметь целочисленные значения.

Gomory в [9] показал, что если для какой-то переменной нарушено требование целочисленности после решения линейного приближения, то можно построить новое ограничение, которое окажется нарушенным при данных значениях переменных и при этом не противоречит исходной задаче.

Данный алгоритм полезен также и в тех случаях, когда имеется множество ограничений большой мощности, описывающих задачу. В таком случае после решения линейного приближения мы можем как-то попытаться найти в этом множестве нарушенные ограничения и добавить их в формулировку.

1.2.3. Комбинация методов

Метод ветвей и сечений является комбинацией двух вышеупомянутых методов. Идея заключается в том, что при решении подзадачи в алгоритме ветвей и границ мы можем сделать несколько итераций метода секущих плоскостей, добавив некоторые нарушенные ограничения.

1.2.4. Жизненный цикл *IBM ILOG CPLEX*

Солвер задачи смешанного целочисленного программирования *IBM ILOG CPLEX* использует в качестве своей основы метод ветвей и сечений. В нем есть два режима работы: «традиционный» и «динамический поиск». В первом случае позволяется непосредственно управлять процессом работы алгоритма ветвей и сечений: выбирать переменную для ветвления, выбирать нерассмотренную подзадачу, добавлять ограничения, использовать эвристики для поиска решения. Блок-схема работы солвера представлена на рисунке 2.

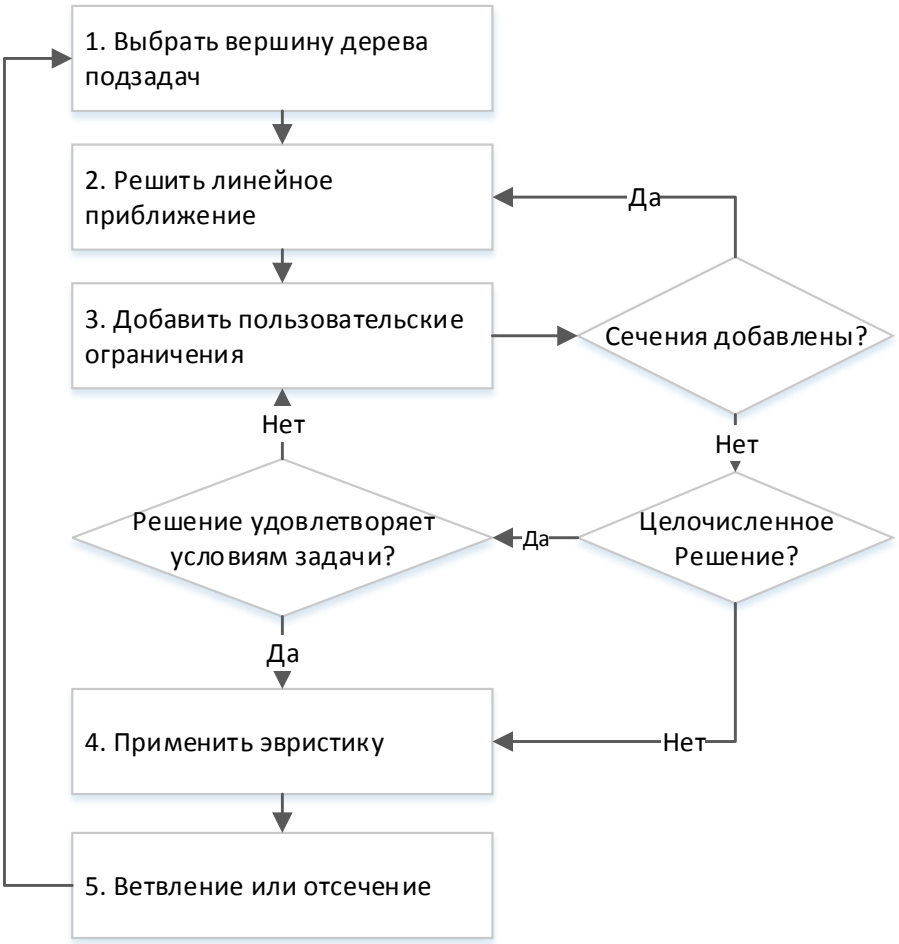

Рисунок 2 – Блок схема работы *IBM ILOG CPLEX*

В *IBM ILOG CPLEX* реализованы алгоритмы по генерации дополнительных сечений нескольких видов, в том числе и сечений Gomory, о которых упоминалось выше.

Вторая стратегия с названием «динамический поиск» использует те же элементы, что и «традиционный» режим, однако программист не имеет доступа к управлению процессом. В этом случае необходимо задать задачу смешанного целочисленного программирования и настроить некоторые параметры, после чего начнется решение задачи.

1.3. Известные сведения

В этом параграфе мы опишем два существующих солвера. Первый из них работает достаточно быстро на практике, однако решает вершинно-взвешенную задачу. Второй решает обобщенную задачу, однако недостаточно хорошо показывает себя на практике.

1.3.1. Обобщенная задача поиска связного подграфа максимального веса

Haouari и др. в [8] разработали сведение обобщенной задачи к задаче смешанного целочисленного программирования. Сперва мы рассмотрим предложенную нелинейную формулировку, а затем посмотрим на результат применения техники переформулировки-линеаризации, примененной к этой формулировке.

Обобщенная задача может быть разбита на две части: целевую функцию (вес подграфа), которая должна быть максимизирована и ограничения, которые гарантируют, что подграф связен. Целевая функция линейна и может быть перенесена в задачу смешанного целочисленного программирования как есть. Однако, получение линейных ограничений на связность подграфа нетривиально.

1.3.1.1. Представление подграфа

В сведении используется по одной бинарной переменной на каждую вершину и ребро. Эти переменные отображают присутствие вершины или ребра в подграфе.

1. Бинарная переменная y_v принимает значение 1 тогда и только тогда, когда $v \in V$ принадлежит подграфу.
2. Бинарная переменная w_e принимает значение 1 тогда и только тогда, когда $e \in E$ принадлежит подграфу.

Для того, чтобы переменные представляли корректный подграф (не обязательно связный) вводятся следующие ограничения:

$$w_e \leqslant y_v, \qquad \forall v \in V, e \in \delta_v. \tag{1}$$

В этих ограничениях утверждается, что ребро может быть частью подграфа, только если инцидентные ему вершины также являются частью подграфа.

1.3.1.2. Нелинейная формулировка

Нелинейная формулировка ограничений связности подграфа основана на идее, что любой связный подграф может быть обойден из любой его вершины. Результат такого обхода может быть представлен в виде дерева обхода, где

дуга (v, u) означает, что вершина v была посещена до u. Соответственно, мы можем гарантировать связность если мы сможем предоставить дерево обхода, соответствующее обходу этого подграфа.

Для графа $G = (V, E)$ пусть $S = (V, A)$ — ориентированный граф, где A — множество дуг, полученное из E путем замены каждого неориентированного ребра $e = (v, u)$ на две дуги (v, u) и (u, v).

Далее мы рассмотрим переменные, которые используются в сведении и посмотрим на нелинейную формулировку задачи.

1. Бинарная переменная x_a принимает значение 1, если и только если $a \in A$ принадлежит дереву обхода,
2. Бинарная переменная r_v принимает значение 1, если и только если $v \in V$ является корнем дерева обхода.
3. Вещественная переменная d_v принимает значение n, если путь в дереве обхода от корня до вершины v содержит n вершин. Если v не принадлежит подграфу, то значение может быть произвольным.

Тогда мы можем записать следующую нелинейную формулировку обобщенной задачи поиска связного подграфа максимального веса. Формулировка требует, чтобы подграф соответствующий переменным y_v, и w_e был связным и при этом максимизирует целевую функцию:

$$\sum_{e \in E} \omega(e) w_e + \sum_{v \in V} \omega(v) y_v \to max; \tag{2}$$

$$\sum_{v \in V} r_v = 1; \tag{3}$$

$$1 \leqslant d_v \leqslant n, \qquad \forall v \in V; \tag{4}$$

$$\sum_{(u,v) \in A} x_{uv} + r_v = y_v, \qquad \forall v \in V; \tag{5}$$

$$x_{vu} + x_{uv} \leqslant w_e, \qquad \forall e = (v, u) \in E; \tag{6}$$

$$d_v r_v = r_v, \qquad \forall v \in V; \tag{7}$$

$$d_u x_{vu} = (d_v + 1) x_{vu}, \qquad \forall (v, u) \in A. \tag{8}$$

Неравенство (3) требует того, чтобы был ровно один корень дерева обхода; (4) — это ограничение на расстояние между вершиной и корнем; (5) требует, чтобы при взятии вершины в подграф она была либо корнем, либо в нее входи-

ло ребро дерева; (6) утверждает, что дуга дерева обхода может быть в решении, только если соответствующее неоиентированное ребро в нем. Последние два неравенства (7) и (8) контролируют корректные дистанции в дереве обхода.

Haouari и др. показали в [8], что эта нелинейная система корректно описывает обобщенную задачу. То есть то, что дерево обхода покрывает все вершины и то, что решение может породить такое дерево обхода.

Однако неравенства (7) и (8) нелинейны и должны быть записаны в линейном виде, чтобы эту формулировку можно было представить как задачу смешанного целочисленного программирования.

1.3.1.3. Применение техники переформулировки-линеаризации

Haouari и др. использовали технику переформулировки-линеаризации, предложенную Sherali в [10, 11]. В результате, в формулировку были добавлены переменные t_{ij} для каждого ребра из A и переменные z_v и u_v для каждой вершины. Кроме того, в результате, переменные d_v были исключены из формулировки. Формулировка приняла следующий вид:

$$\sum_{e \in E} \omega(e) w_e + \sum_{v \in V} \omega(v) y_v \to max; \tag{9}$$

$$\sum_{v \in V} r_v = 1; \tag{10}$$

$$\sum_{(u,v) \in A} x_{uv} + r_v = y_v, \qquad \forall v \in V; \tag{11}$$

$$\sum_{(u,v) \in A} t_{uv} + u_v + \sum_{(u,v) \in A} x_{uv} = z_v, \qquad \forall v \in V; \tag{12}$$

$$2y_v - r_v \leqslant z_v, \qquad \forall v \in V; \tag{13}$$

$$z_v \leqslant n y_v - (n-1) r_v, \qquad \forall v \in V; \tag{14}$$

$$x_{uv} \leqslant t_{uv}, \qquad \forall (u,v) \in A; \tag{15}$$

$$t_{uv} \leqslant (n-1) x_{uv}, \qquad \forall (u,v) \in A; \tag{16}$$

$$x_{uv} + x_{vu} \leqslant w_e, \qquad \forall e = (v,u) \in E; \tag{17}$$

$$z_v - y_v + x_{vu} \geqslant t_{vu} + t_{uv}, \qquad \forall (u,v) \in A, v \in V; \tag{18}$$

$$z_v - n y_v + (n-1) x_{uv} + n x_{vu} \leqslant t_{uv} + t_{vu}, \qquad \forall (u,v) \in A, v \in V; \tag{19}$$

$$w_e \leqslant y_v, \qquad \forall v \in V, e \in \delta_v; \tag{20}$$

$$z_v \geqslant 0, \qquad \forall v \in V; \tag{21}$$

$$t_{uv} \geqslant 0, \qquad \forall (u,v) \in A, u \in V. \quad (22)$$

Было доказано, что существование решения в таких ограничениях эквивалентно существованию решения задачи в нелинейной формулировке.

1.3.2. Вершинно-взвешенная задача поиска связного подграфа максимального веса

Один из лучших алгоритмов для решения вершинно-взвешенной задачи поиска связного подграфа максимального веса описан в [7] и реализован в программном продукте *Heinz2*. Алгоритм состоит из препроцессинга, декомпозиции, и сведения, использующего метод ветвей и сечений.

1.3.2.1. Препроцессинг

Препроцессинг для вершинно-взвешенной задачи поиска связного подграфа максимального веса в *Heinz2* состоит из пяти правил, которые применяются в определенном порядке.

1. Правило удаления изолированной вершины. Если в графе есть изолированная вершина отрицательного веса, то мы можем удалить ее.
2. Правило объединения групп смежных неотрицательных вершин.
3. Правило объединения отрицательных цепей. Если в графе есть цепь из вершин отрицательного веса со степенью два, то можно заменить всю цепь на одну вершину со степенью два.
4. Зеркальное правило. Если две отрицательные вершины смежны одним и тем же вершинам, то вершину меньшего веса можно удалить.
5. Правило меньшей цены. Если между смежными отрицательной вершине со степенью два существует путь, суммарный вес вершин в котором меньше веса вершины, то эту вершину можно удалить. Данное правило может быть частично применено с помощью алгоритма Дейкстры [12].

1.3.2.2. Декомпозиция

В *Heinz2* декомпозиция устроена следующим образом: если одна из компонент двусвязности содержит только одну точку сочленения, то после решения корневой и некорневой задач для этой компоненты можно удалить все вершины компоненты, кроме точки сочленения, изменив при этом значение $\omega(ap)$, где ap — точка сочленения. Утверждается при этом, что при правильном порядке действий мы не упустим оптимального решения исходной задачи.

1.3.2.3. Сведение к задаче смешанного программирования

В *Heinz2* используется сведение с экспоненциальным числом ограничений относительно числа вершин. Как было сказано в разделе 1.2.2, в этом случае можно пытаться добавлять только необходимые сечения. В этом сведении используются следующие переменные:

1. бинарная переменная x_v, принимающая значение 1 в случае, если вершина v принадлежит подграфу,
2. бинарная переменная y_v, принимающая значение 1 в случае, если это корневая вершина.

Корневая вершина в этом сведении — это одна из вершин подграфа, мы будем считать что подграф не связен, если из какой-то вершины не существует пути до корневой.

Тогда формулировка принимает вид:

$$\sum_{v \in V} \omega(v) x_v \to max; \tag{23}$$

$$\sum_{v \in V} r_v = 1; \tag{24}$$

$$y_v \leqslant x_v, \qquad \forall v \in V; \tag{25}$$

$$x_v \leqslant \sum_{u \in \delta(S)} x_u + \sum_{u \in S} y_u, \qquad \forall v \in V, \{v\} \subset S \subset V. \tag{26}$$

Ограничения (26) требуют, чтобы для всех подмножеств S множества вершин, содержащих вершину подграфа v было верно одно из двух:

1. внутри S есть корневая вершина,
2. какая-то из вершин $\delta(S)$ взята в решение.

Очевидно, что в случае несвязности подграфа будет нарушено одно из ограничений такого вида: в качестве S необходимо взять компоненту связности, не содержащую корень, а в качестве v — любую вершину из S. Тогда очевидно, что внутри S не будет корня, а все соседние вершины с S не включены в решение, поскольку в обратном случае S не было бы компонентой связности. Álvarez-Miranda и др. в [6] показали, что поиск нарушенных ограничений такого вида после решения линейного приближения в точности соответствует задаче поиска минимального разреза во вспомогательном графе, полученном

простым преобразованием из исходного графа и значений переменных решения линейного приближения.

1.4. Задачи

Для достижения поставленной во введении цели в результате обзора предметной области были сформулированы следующие задачи:

1. Модифицировать правила препроцессинга для обобщенной задачи.
2. Сформулировать правила для наиболее удобной на практике декомпозиции.
3. Линеаризовать нелинейное сведение компактными и простыми для понимания ограничениями, не требующими введения дополнительных переменных.
4. Добавить к сведению ограничения нарушения симметрии.
5. Модифицировать дополнительные сечения для обобщенной задачи и разработать алгоритм для поиска нарушенных ограничений.

Выводы по главе 1

В этой главе были рассмотрены основные методы решения задачи смешанного целочисленного программирования, а также описаны два солвера и методы, используемые в них. Кроме того, были введены основные термины и понятия. Эта глава носит обзорный характер. Она необходима для понимания остальной части работы.

ГЛАВА 2. ОПИСАНИЕ ПРЕДЛАГАЕМОГО МЕТОДА

В этой главе мы даем описание применяемого алгоритма а также показываем его корректность. Данный алгоритм состоит из нескольких частей, которые подробно описаны в данной главе.

1. Препроцессинг, позволяющий рассматривать вместо исходного графа граф с меньшим числом вершин или ребер. Данный этап может существенно упростить рассматриваемую задачу.
2. Декомпозиция, позволяющая решать вместо исходной задачи три задачи меньшего размера.
3. Сведение. На этом этапе для каждой возникшей на прошлом этапе задачи формируется экземпляр задачи смешанного целочисленного линейного программирования.
4. Решение. Задача решается с помощью солвера задачи смешанного целочисленного программирования.
5. Восстановление решения. На основе данных, полученных с предыдущих этапов работы алгоритма можно получить ответ на исходную задачу.

2.1. Препроцессинг

Мы представляем два правила для препроцессинга, которые были адаптированы из [7]. Они позволяют упростить задачу, создавая новый граф с меньшим числом ребер и вершин таким образом, чтобы решение обобщенной задачи можно было легко восстановить из решения обобщенной задачи для упрощенного графа.

Для начала мы объединяем группы соседних вершин, которые могут войти в решение только полностью (рис. 3). Пусть $e = (u, v)$ — ребро с весом $\omega(e) \geqslant 0$ и одновременно $\omega(e) + \omega(v) \geqslant 0$ и $\omega(e) + \omega(u) \geqslant 0$. В этом случае если одна из вершин присутствует в решении, то ребро и другая вершина могут быть добавлены в решение без уменьшения суммарного веса. Поэтому, мы можем стянуть ребро e в одну вершину w с весом $\omega(w) = \omega(e) + \omega(u) + \omega(v)$. После данной процедуры могут возникнуть парные ребра между вершиной w и какой-то вершиной t. В этом случае мы объединяем все неотрицательные ребра в одно с весом, равным сумме его составных частей. После этого мы удаляем все ребра между w и t, кроме одного с максимальным весом. Мы пытаемся применять это правило в цикле, пока граф меняется.

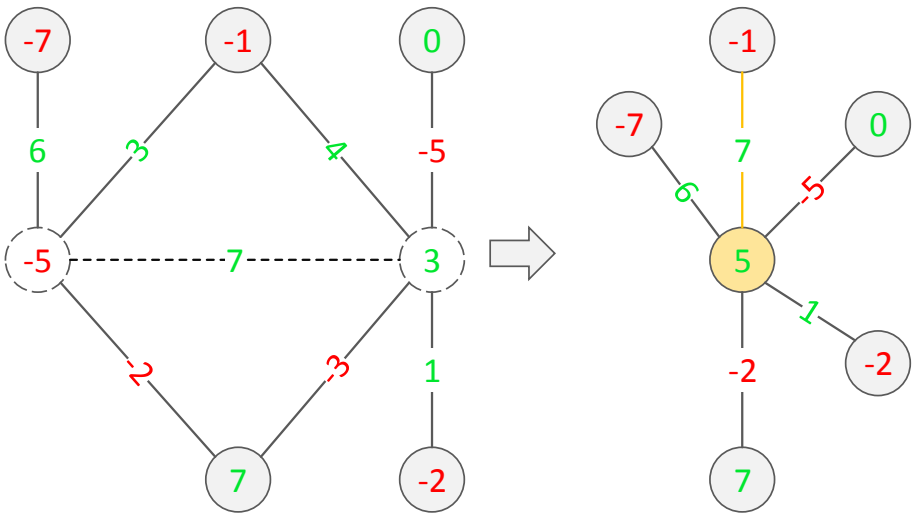

Рисунок 3 – Пример применения первого правила препроцессинга, здесь отмеченные пунктиром вершины и ребро стягиваются в одну вершину, при этом образовавшиеся положительные парные ребра объединились, а из отрицательных парных ребер осталось наибольшее по весу, возникшие элементы отмечены желтым

Во-вторых, мы объединяем цепи вершин и ребер отрицательного веса (Рис. 4). Пусть v — это вершина с $deg(v) = 2$ с соответствующими инцидентными ребрами $e_1 = (u, v)$ и $e_2 = (v, w)$. Если все три веса $\omega(v)$, $\omega(e_1)$ и $\omega(e_2)$ неположительны, тогда v, e_1 и e_2 могут быть заменены на одно ребро $e = (u, w)$ с весом $\omega(e) = \omega(v) + \omega(e_1) + \omega(e_2)$. Объединение отрицательных цепей может быть совершено в один проход, итеративно пытаясь применить правило ко всем вершинам.

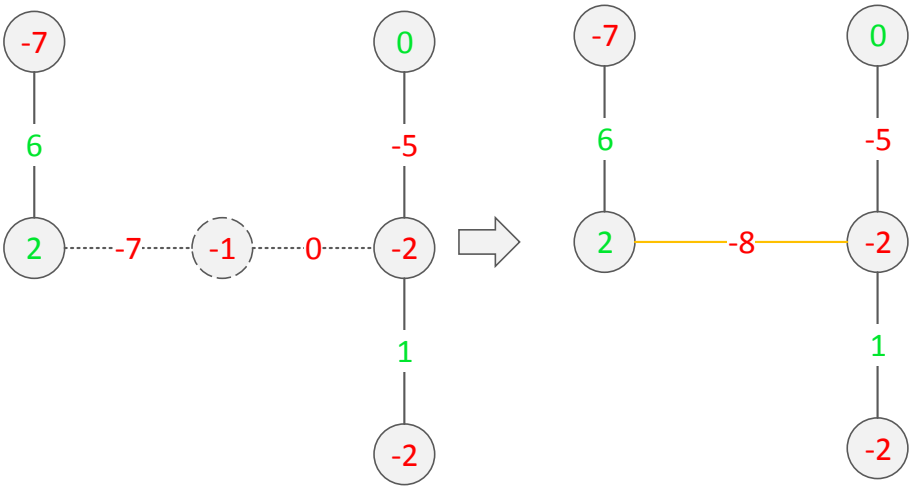

Рисунок 4 – Пример применения второго правила препроцессинга, здесь два ребра и вершина отрицательного веса, отмеченные пунктиром, были заменены одним ребром (желтого цвета)

2.2. Декомпозиция по точкам сочленения

В этом параграфе мы обсудим как экземпляр обобщенной задачи может быть разделен на три подзадачи меньшего размера. Декомпозиция основана на идее, что двусвязные компоненты могут быть рассмотрены раздельно [7].

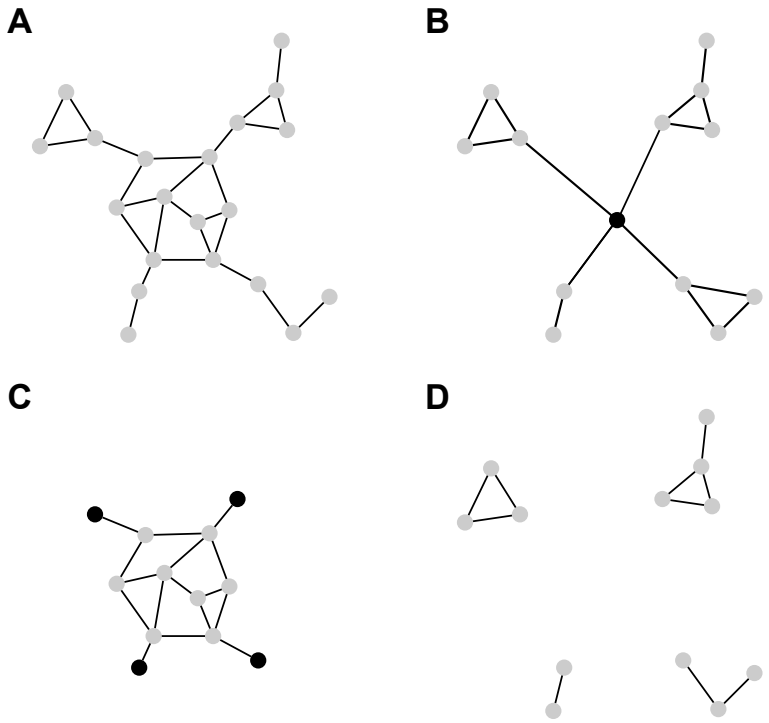

Рисунок 5 – Исходный граф и экземпляры, порожденные декомпозицией

Кратко, пусть имеется экземпляр обобщенной задачи на входе (Рис. 5A). Сперва, мы объединяем наибольшую двусвязную компоненту в одну вершину с нулевым весом и решаем корневую задачу для модифицированного таким образом графа с корнем в созданной вершине (Рис. 5B). Затем, мы заменяем каждую из компонент, ветвящуюся из наибольшей двусвязной компоненты на одну вершину с весом, равным весу соответствующего подграфа в решении корневой задачи с предыдущего шага (Рис. 5C). Напоследок, мы пытаемся найти подграф с наибольшем весом, который лежит полностью в одной из этих ветвящихся компонент (Рис. 5D).

Формально, пусть B — двусвязная компонента графа G с максимальным числом вершин. Пусть C — набор точек сочленения графа G, содержащихся в B. Пусть B_c — компонента связности, содержащая c в графе $G \setminus (B \setminus C)$.

Лемма 6. Пусть подграф $\widetilde{G}$ графа G — оптимальное решение обобщенной задачи для графа G и $\widetilde{G}_c$, $\forall c \in C$ — оптимальные решения для графов B_c с

корнем в вершине c. В этом случае если $\widetilde{G}$ содержит вершину $c \in C$, тогда мы можем сконструировать оптимальное решение $\widetilde{G}'$ такое что:

1) $\widetilde{G}' \cap B = \widetilde{G} \cap B$ и 2) $\widetilde{G}' \cap B_c = \widetilde{G}_c$.

Доказательство. Пусть $\widetilde{B}_c = \widetilde{G} \cap B_c$. Мы покажем, что он может быть заменен $\widetilde{G}_c$ без потери связности и оптимальности. Во-первых, $\widetilde{B}_c$ должен быть связен. Пусть он несвязен. Тогда не существует пути между c и какой-то вершиной v. Поскольку $\widetilde{G}$ связен, то существует простой путь vc в G. Однако, по определению точки сочленения путь vc не может содержать вершин из $G \backslash B_c$ и, следовательно, он полностью лежит в B_c, противоречие. Поскольку $\widetilde{B}_c$ связен и содержит c, то он не может иметь вес больший, чем $\widetilde{G}_c$ по построению $\widetilde{G}_c$.

Сейчас мы покажем, что замена сохраняет связность графа. Это можно сделать, повторив рассуждения с предыдущего шага доказательства и получив, что $\widetilde{G} \cap B$ должен быть связен. Таким образом, $\widetilde{G}_c$ связен, $\widetilde{G} \cap B$ связен и оба этих подграфа содержат вершину c. Поэтому и $\widetilde{G}'$ тоже связен.

□

Эта лемма позволяет нам рассматривать только оптимальные решения, которые либо включают вершину из B и в подграфах B_c идентичны соответствующему экземпляру корневой задачи, либо полностью лежат подграфах B_c.

Во-первых, для каждой $c \in C$ мы хотим знать лучшее решение задачи для графа B_c, содержащее вершину c. Это в точности экземпляр корневой задачи. По практическим причинам, лучше порождать один экземпляр на этом шаге вместо $|C|$ экземпляров. Пусть $G^* = \bigcup_{\forall c \in C} B_c$. Тогда мы объединяем все вершины из C, содержащиеся в G^*, в одну вершину r с весом $\omega(r) = 0$ и решаем корневую задачу для этого графа. Пусть S — это решение такого экземпляра. Чтобы получить решение для графа B_c мы заменяем обратно r на c в S и удаляем все вершины, которые не содержатся в B_c.

Во-вторых, мы ищем подграф максимального веса, который не лежит полностью в одном из B_c. Пусть $\widetilde{G}_c$ — решение корневой задачи для графа B_c с корнем c, полученное на предыдущем шаге работы алгоритма. Мы получаем новый экземпляр обобщенной задачи, рассматривая компоненту B и для всех $c \in C$ устанавливаем вес $\omega(v) = \Omega(\widetilde{G}_c)$. Мы решаем получившийся экземпляр, а затем восстанавливаем ответ на исходную задачу.

И напоследок, мы ищем потенциальные решения, которые лежат целиком в B_c для всех $c \in C$. Для этого мы порождаем еще один экземпляр для графа $G^* = \bigcup_{\forall c \in C} B_c$. Очевидно, что если решение задачи для графа G лежит полностью в одном из B_c, то мы найдем его на этом шаге работы алгоритма.

2.3. Сведение к задаче смешанного целочисленного программирования

В этом параграфе мы рассмотрим сведение задачи к задаче смешанного целочисленного программирования. Мы воспользуемся нелинейной формулировкой, описанной в разделе 1.3.1.2, покажем то, как лучше избавиться от нелинейности и как можно генерировать дополнительные сечения для корневой обобщенной задачи. Кроме того, мы введем так называемые ограничения нарушения симметрии, которые для уменьшения пространства будут требовать определенный вид дерева обхода.

2.3.1. Линеаризация

Применение техники переформулировки-линеаризации позволило получить линейные ограничения из нелинейных (раздел 1.3.1.3), однако получившаяся формулировка оказалась достаточно громоздкой и сложной для понимания. Кроме того, были введены новые наборы переменных. В этом параграфе мы покажем то, как можно получить компактную линейную формулировку из нелинейной, не вводя дополнительных переменных.

Итак, нелинейные равенства (7) и (8) могут быть заменены неравенствами следующего вида:

$$d_v + nr_v \leqslant n, \qquad \forall v \in V; \tag{27}$$

$$n + d_u - d_v \geqslant (n+1)x_{vu}, \qquad \forall (v,u) \in A; \tag{28}$$

$$n + d_v - d_u + \geqslant (n-1)x_{vu}, \qquad \forall (v,u) \in A. \tag{29}$$

Лемма 7. Любое возможное решение, удовлетворяющее (1)-(8), также удовлетворяет (1)-(6), (27)-(29) и наоборот.

Доказательство. Мы будем говорить, что две системы неравенств эквивалентны, если они ограничивают одни и те же наборы решений. В данном случае система из одного неравенства отождествляется самому неравенству. Сперва мы покажем, что неравенство (27) эквивалентно (7). Поскольку r_v — бинарная переменная, мы можем рассмотреть два случая. Предположим, что

$r_v = 1$, тогда (7) примет вид $d_v = 1$, когда (27) примет форму $d_v \leqslant 1$, вместе с (4) мы получаем $d_v = 1$. Теперь предположим, что $r_v = 0$. Неравенство (7) примет вид $0 = 0$, что означает, что дополнительных ограничений не накладывается. (27) примет форму $d_v \leqslant n$, однако система уже содержит такое неравенство, а значит, как и в первом случае, множество решений дополнительно не ограничивается.

Во второй части доказательства мы будем использовать тот же подход. Здесь мы покажем, что (8) может быть представлен в виде линейных неравенств (28) и (29).

1. Пусть $x_{vu} = 1$. После подстановки в (8) мы имеем $d_u = d_v + 1$. Тогда мы подставляем x_{vu} в (28) и (29)

$$n + d_u - d_v \geqslant n + 1$$
$$n + d_v - d_u \geqslant n - 1$$

или

$$d_u \geqslant d_v + 1$$
$$d_v + 1 \geqslant d_u$$

или $d_u = d_v + 1$.

2. Пусть $x_{vu} = 0$. Нелинейное уравнение примет вид $0 = 0$. Нам следует показать, что (28) и (29) тоже не добавляют дополнительных ограничений на переменные. После подстановки неравенства примут вид:

$$n + d_u - d_v \geqslant 0$$
$$n + d_v - d_u \geqslant 0$$

или $|d_v - d_u| \leqslant n$. Очевидно, что если выполняются (4), то и данное неравенство тоже. Поэтому дополнительных ограничений не вводится $\square$

2.3.2. Ограничения нарушения симметрии

В таких задачах принято уменьшать число возможных решений задачи смешанного целочисленного программирования посредством ограничения числа различных, но логически эквивалентных решений. Такие решения называются симметричными. В нашей формулировке ограничения (1)-(6), (27)-(29)

разрешают любому дереву обхода доказывать связность графа. В связи с этим в этом параграфе мы показываем, как уменьшить число возможных деревьев обхода и, таким образом, уменьшить пространство поиска.

2.3.2.1. Правило выбора корня

Прежде всего, для некорневой обобщенной задачи мы требуем, чтобы корнем была вершина с максимальным весом в подграфе. Соответствующие ограничения добавляются в формулировку задачи смешанного программирования.

$$\sum_{v \prec u} r_v \leqslant 1 - y_u, \qquad \forall u \in V, \tag{30}$$

где $v \prec u$, если $\omega(v) < \omega(u)$. В случае совпадения весов мы используем какой-нибудь линейный порядок на вершинах.

Для корневой задачи мы устанавливаем корень дерева обхода таким же, как и корень экземпляра корневой задачи.

2.3.2.2. Ограничение вида дерева обхода

Более того, связный подграф может быть обойден из одной вершины различными способами. В этом параграфе мы покажем как запретить такие решения, которые не могут быть достигнуты, используя поиск в ширину [12].

Для достижения такой формы деревьев мы добавляем ограничения:

$$d_v - d_u \leqslant n - (n-1)w_e, \qquad \forall e = (v,u) \in E; \tag{31}$$

$$d_u - d_v \leqslant n - (n-1)w_e, \qquad \forall e = (v,u) \in E. \tag{32}$$

Эти ограничения требуют соблюдения правила: если подграф содержит ребро e, то дистанции инцидентных вершин отличаются не более, чем на единицу.

Лемма 8. Для любого связного подграфа G_s графа G существует решение $(\overline{r}, \overline{y}, \overline{w}, \overline{x}, \overline{d})$, которое кодирует подграф G_s и удовлетворяет: (1)-(6), (27)-(29) и (30)-(32).

Доказательство. Во-первых, для любого подграфа G_s мы можем выбрать любую его вершину как корень, в том числе и ту, у которой максимальный вес. Сделаем из нее обход в ширину. Как было показано выше, для любого

связного подграфа G_s и любого его дерева обхода существует соответствующее кодирование $(\overline{r}, \overline{y}, \overline{w}, \overline{x}, \overline{d})$ которое удовлетворяет ограничениям (1)-(6) и (27)-(29). По причине выбора вершины с максимальным весом в качестве дерева обхода, (30) выполняется, ограничения (31)-(32) также выполняются, что напрямую следует из порядка обхода в ширину. □

Эта лемма говорит о том, что добавление ограничений нарушения симметрий не сужает набор описываемых системой подграфов.

2.3.3. Дополнительные сечения

Как было упомянуто в обзорной части работы, в формулировку могут быть добавлены дополнительные сечения «на лету». Мы используем сечения, которые похожи на предложенные Álvarez-Miranda и др. в [6]. Однако мы модифицировали их для применения в реберно-взвешенной корневой задаче.

Таким образом, мы используем следующие ограничения:

$$y_v \leqslant \sum_{e \in C} w_e, \qquad \forall v \in V, C \in C_v^*, \tag{33}$$

где C_v^* — это набор всех r-v разрезов графа. Чтобы найти нарушенные ограничения мы ассоциируем с каждым ребром e значение переменной w_e решения линейного приближения задачи. Затем для каждой вершины v мы пытаемся найти нарушенные ограничения с помощью поиска такого r-v разреза C, что $y_v > \sum_{e \in C} w_e$. Поскольку слева стоит константа, то лучшим кандидатом нарушенного ограничения будет то, что соответствует минимальному разрезу. Для поиска минимального разреза используется алгоритм Эдмондса-Карпа. Однако мы останавливаем работу алгоритма, если на очередном шаге мы имеем разрез с суммарным весом, большим y_v.

Выводы по главе 2

В этой главе были описаны все элементы разработанного нами солвера. Было показано, как модифицировать препроцессинг из вершинно-взвешенной задачи поиска связного подграфа максимального веса, как сделать декомпозицию, не порождая большого количества экземпляров, а также была представлена новая линеаризация нелинейной формулировки задачи и введены новые ограничения нарушения симметрии, которые ограничивают виды деревьев обхода. Кроме того, мы показали как в корневой задаче можно добавлять дополнительные сечения.

ГЛАВА 3. ПРАКТИЧЕСКОЕ ИССЛЕДОВАНИЕ

3.1. Описание экспериментов

В качестве датасета мы выбрали 101 экземпляр, сгенерированный Shiny GAM, a web-service for integrated transcriptional and metabolic network analysis [5]. В этом датасете 38 экземпляров вершинно-взвешенной задачи поиска связного подграфа максимального веса и 63 для обобщенного варианта. Медианы числа вершин и ребер в вершинно-взвешенных экземплярах составляют 2217 и 2470 соответственно, причем наименьший экземпляр содержит 48 вершин и 49 ребер, а наибольший — 2669 вершин и 3489 ребер. Медианы числа вершин и ребер в реберно-взвешенных экземплярах составляют 711 и 873 соответственно, причем наименьший экземпляр содержит 558 вершин и 668 ребер, а наибольший — 1194 вершины и 1580 ребер. Количество вершин и ребер подсчитывалось только для наибольшей по размеру компоненте связности экземпляра. Архив доступен по адресу

`http://genome.ifmo.ru/files/papers_files/WABI2016/gmwcs/instances.tar.gz`.

Для сравнения мы выбрали два других солвера: *Heinz* версии 1.68 [4] и *Heinz2* версии 2.1 [7]. Первый, *Heinz*, был изначально разработан для вершинно-взвешенной задачи, позже добавилась возможность устанавливать ребрам веса. Однако с этом случае солвер находит только ациклические решения. Второй, *Heinz2*, вовсе не принимает веса ребер, но работает быстрее *Heinz* на вершинно-взвешенных экземплярах.

Мы запустили каждый из солверов на каждом экземпляре 10 раз с ограничением по времени, равным 1000 секунд. *Heinz2* и наш солвер были запущены в четыре потока, в то время как *Heinz* не поддерживает многопоточное исполнение. Тестирование было произведено на процессоре AMD Opteron 6380 2.5GHz. Таблица результатов доступна по адресу `http://genome.ifmo.ru/files/papers_files/WABI2016/gmwcs/results.final.tsv`.

3.2. Результаты на вершинно-взвешенных экземплярах задачи

Эксперименты показали, что на вершинно-взвешенных экземплярах задачи наш солвер схож по производительности с *Heinz2*. 24 экземпляра (63%) были решены нашим солвером медленнее аналога, однако 32 экземпляра (84%) были решены нашим солвером менее чем за 30 секунд, в то время как *Heinz2*

смог решить только 27 (71%). Более того 4 экземпляра не были решены с помощью *Heinz2* в отведенные 1000 секунд в сравнении с 1 экземпляром для нашего солвера.

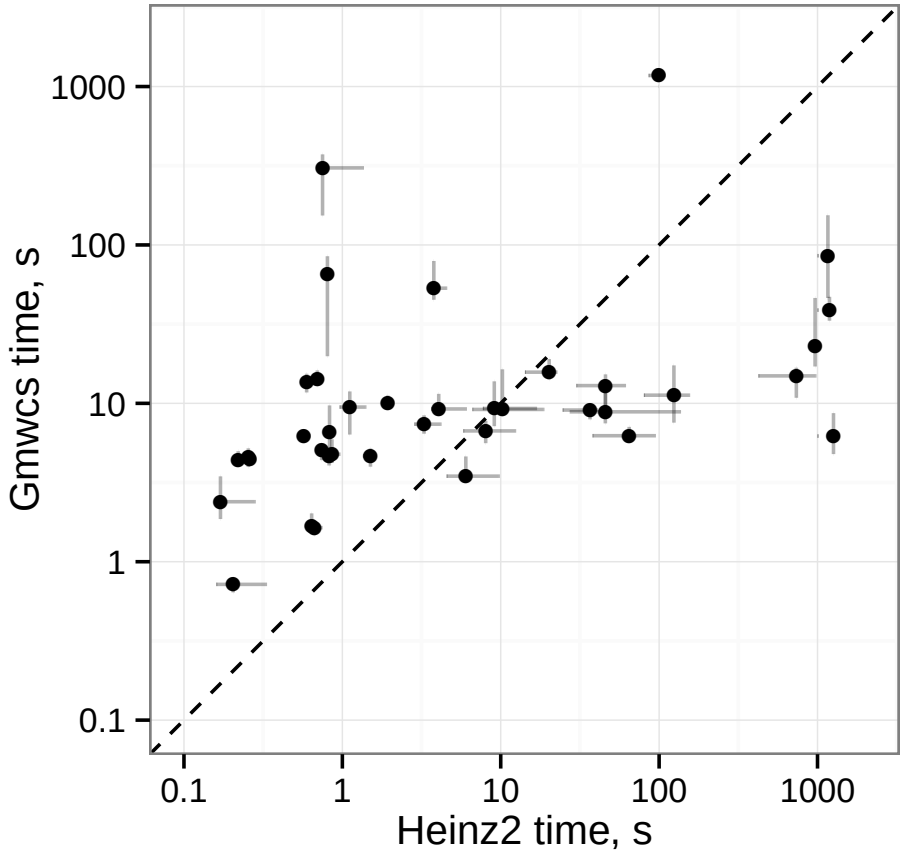

Рисунок 6 – Сравнение нашего солвера с *Heinz2*, точки представляют медианное значение времени работы солвера при 10 запусках на одном экземпляре. Серые линии отображают вторые максимальные и минимальные времена работы.

3.3. Результаты на реберно-взвешенных экземплярах задачи

На реберно-взвешенных экземплярах задачи наш солвер смог найти оптимальное решение в течение 10 секунд для всех экземпляров, кроме двух. В то же время решение с помощью *Heinz* заняло больше 10 секунд на 30 экземплярах (48%). Более того, только 35 экземпляров (56%) имеют ациклические решения, поэтому 28 экземпляров не были решены до оптимальности в смысле обобщенной задачи.

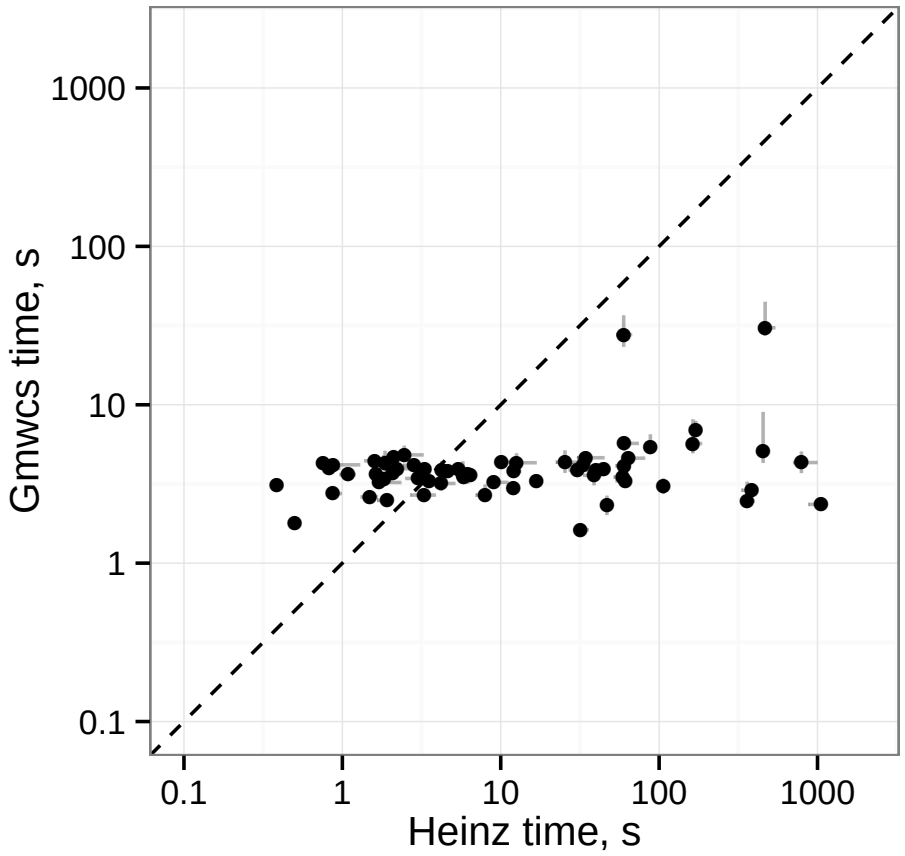

Рисунок 7 – Сравнение нашего солвера с *Heinz* на реберно-взвешенных экземплярах задачи.

Выводы по главе 3

В этой главе были приведены результаты сравнительного тестирования нашего солвера и ближайших аналогов. Эти результаты подтверждают то, что поставленная в начале исследования цель достигнута, а также то, что даже для частных случаев задачи солвер работает за время сравнимое с современными солверами.

ЗАКЛЮЧЕНИЕ

Подходы к анализу сетей активно разрабатываются для анализа биологических данных. С математической точки зрения они обычно соответствуют *NP*-трудным задачам. В этой работе мы описали точный практический солвер для обобщенной задачи поиска связного подграфа максимального веса, которая естественно возникает в метаболических сетях. Мы протестировали метод на реальных данных и показали, что он сравним по производительности с существующими солверами *Heinz2* на вершинно-взвешенных экземплярах и работает лучше и более точно, если сравнивать с *Heinz* на обобщенной задаче. Реализация доступна по адресу `https://github.com/ctlab/gmwcs-solver`.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Chen C.-Y.*, *Grauman K.* Efficient activity detection with max-subgraph search // Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. — IEEE. 2012. — С. 1274—1281.

2 *Vijayanarasimhan S.*, *Grauman K.* Efficient region search for object detection // Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. — IEEE. 2011. — С. 1401—1408.

3 *Dilkina B.*, *Gomes C. P.* Solving connected subgraph problems in wildlife conservation // Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. — Springer, 2010. — С. 102—116.

4 Identifying functional modules in protein-protein interaction networks: an integrated exact approach. / M. T. Dittrich [и др.] // Bioinformatics (Oxford, England). — 2008. — Т. 24, № 13. — С. i223—31. — ISSN 1367-4811. — DOI: 10.1093/bioinformatics/btn161.

5 GAM: a web-service for integrated transcriptional and metabolic network analysis. / A. Sergushichev [и др.] // Nucleic acids research. — 2016.

6 *Álvarez-Miranda E.*, *Ljubić I.*, *Mutzel P.* The maximum weight connected subgraph problem // Facets of Combinatorial Optimization. — Springer, 2013. — С. 245—270.

7 *El-Kebir M.*, *Klau G. W.* Solving the Maximum-Weight Connected Subgraph Problem to Optimality. — 2014. — eprint: 1409.5308.

8 *Haouari M.*, *Maculan N.*, *Mrad M.* Enhanced compact models for the connected subgraph problem and for the shortest path problem in digraphs with negative cycles // Computers & Operations Research. — 2013. — Т. 40, № 10. — С. 2485—2492.

9 *Gomory R. E.* Solving linear programming problems in integers // Combinatorial Analysis. — 1960. — Т. 10. — С. 211—215.

10 *Sherali H. D.*, *Adams W. P.* A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems // SIAM Journal on Discrete Mathematics. — 1990. — Т. 3, № 3. — С. 411—430.

11 *Sherali H. D.*, *Adams W. P.* A hierarchy of relaxations and convex hull characterizations for mixed-integer zeroone programming problems // Discrete Applied Mathematics. — 1994. — Т. 52, № 1. — С. 83—106.

12 Алгоритмы. Построение и анализ / Т. Х. Кормен [и др.]. — Вильямс, 2015.