Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики

Работа допущена к защите
Заведующий кафедрой
«Прикладная математика»
_____ М.Е. Фролов
«___»_____2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ОПТИМИЗАЦИЯ РАБОТЫ ШЛЮЗА

по направлению 01.04.02 «Прикладная математика и информатика»
по образовательной программе
01.04.02_02 ”Системное программирование”

Выполнил
студент гр.23641/2 И.М. Мясоедов

Руководитель
к. ф.-м. н., доцент И.Е. Ануфриев

Санкт-Петербург
2018

Реферат

72 стр., 21 рисунков, 2 таблицы

ПРОВОДКА, ШЛЮЗ, МЕТОДЫ ОПТИМИЗАЦИИ, ЗАДАЧА СМЕШАННОГО ПРОГРАММИРОВАНИЯ, МЕТОД ВЕТВЕЙ И ГРАНИЦ

В данной работе рассмотрен метод шлюзования для вычисления оптимального вектора проводок кораблей. Сделан анализ работы алгоритма и предложен способ его ускорения.

Реализован алгоритм с помощью языка математического моделирования MATLAB. Написаны скрипты для тестирования алгоритма на скорость работы. Сделана визуализация работы алгоритма на примере нескольких кораблей, прибывающих на шлюз.

The Abstract

72 pages, 21 pictures, 2 tables

LOCKAGE, LOCK, OPTIMIZATION METHODS, MIXED INTEGER LINEAR PROBLEM, BRANCH AND BOUNDARY METHOD

In this paper we propose a method for solving a lock scheduling problem. It has been made an analysis of the algorithm. We have proposed the method for its acceleration.

We have developed the algorithm written in MATLAB. It has been made the visualization of the algorithm on the example of five ships approaching the lock.

Оглавление

1. Введение

Судоходные пути и русла очень сильно загружены из-за растущего речного трафика. Поэтому потребность в способах управления трафиком очень высока. Шлюз – один из основных компонентов инфраструктуры портов, позволяющий судам переправляться по каналам с разным уровнем воды. Так как время ожидания своей очереди у судов при шлюзовании может достигать довольно значительного времени (от нескольких часов до нескольких дней), то задача оптимального алгоритма шлюзования является первостепенной. Управление шлюза представляет собой сложную оптимизационную задачу. После применения алгоритма мы должны получить количество проводок судов, по которым можно составить расписание. Неправильное расписание для проводок судов через шлюз приводит к неэффективной трате ресурсов: время пребывания судов в водных узлах увеличивается. Речной транспорт, так же как и морской, нуждается в сокращении времени ожидания на шлюзах. Это необходимо для увеличения общей пропускной способности шлюзов.

Цель настоящей дипломной работы заключается в нахождении оптимального алгоритма шлюзования судов и его реализации. Оптимальность здесь имеет тот смысл, что алгоритм должен находить глобальный минимум целевой функции, заданной на множестве ограничений задачи.

2 Постановка задачи управления шлюза

2.1. Целевая функция

Постановка задачи похожа на ту, которая указана в диссертации нидерландского ученого [2], причем указанная постановка была переработана и в результате получилась задача с другим набором ограничений. Причины такого шага будут даны в конце пункта 2.2.

В работе мы часто будем использовать понятие “шлюз”. Под шлюзом мы понимаем сооружение на реке или канале для пропуска судов при разном уровне воды на пути их следования. Проводка – это процесс провода кораблей с одного уровня на другой.

Мы рассматриваем один шлюз. Корабли прибывают на шлюз сверху и снизу (Рисунок - 1). При этом сами корабли неразличимы, а также не учитывается геометрия расположения судов в пределах проводки.

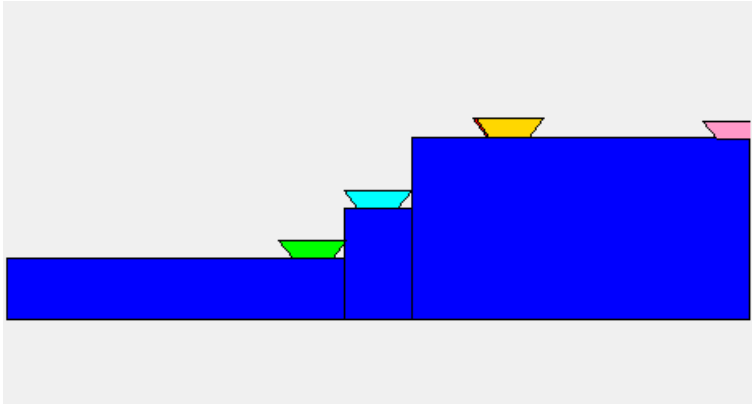

Рисунок 1 – На шлюз прибывают корабли сверху и снизу.

Обозначим через N количество кораблей, прибывающих на шлюз, в течение некоторого промежутка времени t, M – максимальное количество проводок кораблей (без потери общности будем считать, что оно равно $2N$). $\boldsymbol{z_k}$ – булева переменная, равная 0, когда проводка пустая, и 1 в противном случае. Под пустой проводкой мы понимаем проводку, не содержащую кораблей. $\boldsymbol{c_i}$ – время завершения проводки для i–го корабля. $\boldsymbol{T_{max}}$ – максимальное время переправы корабля в процессе шлюзования.

Задача заключается в минимизации целевой функции (с весовыми коэффициентами):

$$min\ \lambda_1 \sum_k \boldsymbol{z_k} + \lambda_2 \sum_i wct_i \boldsymbol{c_i} + \lambda_3 \boldsymbol{T_{max}},\ k = \overline{1,M}, i = \overline{1,N} \qquad (2.1)$$

на некотором ограниченном множестве. Нахождение этого множества будет описано ниже.

Целевая функция (2.1) представляет собой линейную комбинацию нескольких критериев. Если в качестве целевой функции мы возьмем первый член $\sum_k \boldsymbol{z_k}$, то минимизация происходит по количеству проводок. Функция только с вторым членом $\sum_i wct_i \boldsymbol{c_i}$ минимизирует время завершения проводки судов. Функция только с третьим членом $\boldsymbol{T_{max}}$ минимизирует максимальное время пребывания судна при шлюзовании. При необходимости можно смешивать критерии.

2.2. Ограничения для задачи

Каждая проводка корабля должна быть выполнена. Это ограничение можно записать в виде ($\boldsymbol{f_{ik}}$ – булева переменная, принимающая значение 1, когда i-ый корабль находится в k-ой проводке, иначе 0):

$$\boldsymbol{f_{ik}} \le \boldsymbol{z_k},\ \forall i \in \overline{1,N}, k = \overline{1,M} \qquad (2.2)$$

Обозначим через $\boldsymbol{C_k}$ – время завершения k-ой проводки, $\boldsymbol{c_i}$ – время завершения проводки i-ым кораблем. Тогда время завершения проводки для i-го корабля равно времени завершения проводки, в которой он участвует:

$$\boldsymbol{c_i} \ge C_{max}(\boldsymbol{f_{ik}} - 1) + \boldsymbol{C_k},\quad \forall i = \overline{1,N},\ k = \overline{1,M} \qquad (2.3)$$

$$\boldsymbol{c_i} \le C_{max}(1 - \boldsymbol{f_{ik}}) + \boldsymbol{C_k},\quad \forall i = \overline{1,N},\ k = \overline{1,M} \qquad (2.4)$$

Проводка выполняется только в том случае, когда она содержит по крайней мере один корабль:

$$\boldsymbol{z_k} \le \sum_{i \in \overline{1,N}} \boldsymbol{f_{ik}},\ \forall k = \overline{1,M} \qquad (2.5)$$

Максимальное время перемещения всех кораблей в пределах шлюза определяется следующим неравенством:

$$\boldsymbol{T_{max}} \geq \boldsymbol{c_i} - r_i,\ \forall i == \overline{1,N}. \quad (2.6)$$

Ограничения, описывающие объем проводки, задаются неравенствами:

$$\sum_{i \in N} \boldsymbol{f_{ik}} \leq V;\ \forall k = \overline{1,M} \quad (2.7)$$

Для решения задачи нужно ограничить множество решений. Для этого введем искусственно два ограничения для переменных $\boldsymbol{T_{max}}$, $\boldsymbol{C_k}$, неограниченных сверху. Играть роль этого ограничения будет константа B:

$$\boldsymbol{T_{max}} \leq B \quad (2.8)$$

$$\boldsymbol{C_k} \leq B \quad (2.9)$$

Если через r_i обозначить время прибытия i-го корабля на шлюз, то ограничение, описывающее границу снизу для времени завершения проводки C_k, можно записать в виде неравенства:

$$\boldsymbol{C_k} \geq \boldsymbol{f_{ik}} r_i,\ \forall i \in \overline{1,N},\ k = \overline{1,M} \quad (2.10)$$

Введем константу p_t, характеризующую время подъема или спуска шлюза. Ограничение, гарантирующее временную последовательность времен завершения проводок, описывается системой:

$$\boldsymbol{C_{k+1}} \geq \boldsymbol{C_k} + p_t;\ \boldsymbol{C_1} \geq p_t \quad (2.11)$$

Непустая проводка может начаться только в том случае, когда шлюз будет находиться на том же уровне, что и сам корабль:

$$\boldsymbol{f_{ik}} \leq 1 - u_i(1 - s_k) - (1 - u_i)s_k \quad (2.12)$$

Каждый корабль перемещается ровно в одной проводке:

$$\sum_{k \in M} \boldsymbol{f_{ik}} = 1, \forall i = \overline{1,N}, \quad (2.13)$$

Из анализа ограничений (2.2) – (2.13) можно сделать вывод, что мы получили задачу смешанного целочисленного линейного программирования (2.1), (2.2) – (2.13), которую в западной литературе называют MILP (mixed integer linear program).

Сделаем некоторые замечания по поводу ограничений (2.2) – (2.13).

Ограничения (2.2) – (2.6), (2.13) взяты из работы [2]. Оставшиеся ограничения были добавлены в течение реализации алгоритма шлюзования.

В процессе написания программы появились трудности в реализации алгоритма из работы [2]. А именно, реализованный алгоритм некорректно находил оптимальное решение, в связи с чем был сделан вывод, что либо не все ограничения были представлены в работе [2], либо в ограничениях автором были допущены ошибки. Поэтому из постановки задачи была убрана часть ограничений и добавлены новые ограничения.

3. Задача смешанного целочисленного линейного программирования

Задача смешанного целочисленного программирования – это задача, обладающая следующими свойствами:

1) Целевая функция имеет вид $f^T \boldsymbol{x}$, где f – это вектор констант, а $\boldsymbol{x}$ – вектор неизвестных.
2) Ограничения в задаче являются только линейными.
3) Некоторые из компонент вектора $\boldsymbol{x}$ могут принимать только целочисленные значения.

Дадим более строгое определение. Пусть даны векторы f, lb, ub, матрицы A и A_{eq}, соответствующие им векторы b и b_{eq}, и множество индексов S. Нужно найти вектор x, являющийся оптимальным решением задачи:

$$\min_x f^T x \text{ на множестве} \begin{cases} x(S) \text{ целые числа} \\ A \cdot x \leq b \\ A_{eq} \cdot x = b_{eq} \\ lb \leq x \leq ub, \end{cases}$$

будем называть эту задачу MILP (Mixed-Integer Linear Programming).

Решить такую задачу можно с помощью популярных программ для математических расчетов, например MATLAB. Функция для расчета задачи – intlinprog.

4. Общий алгоритм решения задачи MILP

Алгоритм состоит из следующих частей [12] :

1) Сокращение размера задачи, используя линейную предобработку.
2) Решение исходной задачи с ослабленными ограничениями (нецелочисленная задача), LP - релаксация, используя методы линейного программирования.
3) Использование смешанной целочисленной предобработки для сжимания LP релаксации для MILP.
4) Использование метода ветвей и границ для поиска оптимального решения. Это алгоритм решает LP релаксации в ограниченной области для целочисленных переменных. Он пытается сгенерировать последовательность обновленных границ для нахождения оптимального значения целевой функции.

5. Линейная предобработка

Пусть даны матрицы A и A_{eq} и соответствующие векторы b и b_{eq}, которые определяют множество линейных неравенств и равенств

$$A \cdot x \leq b \qquad (5.1)$$
$$A_{eq} \cdot x = b_{eq}$$

Эти линейные ограничения задают некоторый вектор x.

Часто есть возможность сократить количество переменных в задаче (число компонент вектора x) и уменьшить количество линейных ограничений. Так как сокращение числа ограничений может занять некоторое время, то этот процесс может повлиять на общее время нахождения решения задачи в целом. Производительность алгоритма может как уменьшится, так и увеличиться. С помощью линейной предобработки иногда также можно определить, что задача не имеет допустимых решений.

5.1. Обозначения

Перепишем (5.1) в следующей форме:

$\min c^T x$, при условии

$$Ax = b, \qquad (5.2)$$
$$l \leq x \leq u,$$

где $x,\ c,\ l,\ u \in \mathbf{R}^n$ и $A \in \mathbf{R}^{m \times n}$. Некоторые из ограничений l_j или u_j могут принимать значение $-\infty$ или $+\infty$ соответственно. Определим $L = \{j: l_j > -\infty\}$ и $U = \{j | u_j < +\infty\}$; тогда ограничения для (5.2) могут быть записаны так:

$$\begin{cases} Ax^*=b, \\ A^T y^*+z^* = c \\ \left(x_j^*-l_j\right)z_j^*=0, \forall j\in L, \\ \left(u_j-x_j^*\right)z_j^*=0, \forall j\in U, \\ z_j^*\geq 0, \forall j\in\left\{j\in L: x_j^*=l_j \wedge l_j<u_j\right\} \\ z_j^*\leq 0, \forall j\in\left\{j\in U: x_j^*=u_j \wedge l_j<u_j\right\} \\ z_j^*=0, \forall j\notin L\cup U, \\ l\leq x^*\leq u, \end{cases} \tag{5.3}$$

где $y \in \mathbf{R}^m$ и $z \in \mathbf{R}^n$.y и z – лагранжевы множители, соответствующие линейным ограничениям в (5.2). Заметим, что символ "*" означает оптимальную величину. В Таблице 1 мы перечислили возможные случаи, которые могут быть получены из условий (5.3). Например, если считать, что $z_j^* > 0$, то тогда x_j равна нижней грани, если она конечная; иначе задача не ограничена.

$\boldsymbol{l_j}$	$\boldsymbol{u_j}$	$\boldsymbol{z_j^*}$	$\boldsymbol{x_j^*}$	Статус задачи
$> -\infty$	$+\infty$	> 0	l_j	–
$-\infty$	$< +\infty$	< 0	u_j	–
$-\infty$	?	> 0	–	Неограничена
?	$+\infty$	< 0	–	Неограничена
?	$< l_j$	?	–	Недостижима

Таблица 1. Заключение, которое можно сделать из оптимальных условий. Здесь "?" означает любую величину; "-" означает неизвестное значение.

5.2. Процедура предобработки

Вычислительная сложность LP алгоритмов, основанных на симплекс методе или методе внутренней точки, зависит от числа ограничений и переменных в (5.2). На практике наиболее важным оказывается определить, насколько разрежена матрица A, потому что только ненулевые величины сохраняются и работа алгоритма зависит от числа ненулевых переменных. Поэтому процедура предобработки должна сократить размер матрицы A без создания новых ненулевых ограничений в A. Преимущество такого подхода в том, что затраты на хранение не могут быть увеличены процедурой предобработки. Недостаток – в не разрешении обычных преобразований над матрицей A, несмотря на то, что это может дать некоторый выигрыш при определенных условиях. Однако мы разрешаем процедуре предобработки модифицировать саму целевую функцию c, правую часть b и простые границы l и u.

Далее мы будем считать, что матрица A разреженная, то есть содержит менее чем, 30% ненулевых элементов. Процедура предобработки может не дать преимуществ, если это предположение не выполнено. Этот факт установлен эмпирически в работе [1]

Идея процедуры – сделать несколько проходов через матрицу A и в каждом проходе определить некоторые избыточности и их удалить. В следующих разделах мы обсудим эту процедуру в деталях.

5.3. Простые методы предобработки

Здесь мы представим некоторые (простые) методы предобработки:

1) **Пустая строка:**

$$\exists i:\ a_{ij} = 0,\ \forall j \tag{5.4}$$

i-ое ограничение либо избыточное, либо недостижимо.

2) **Пустой столбец:**

$$\exists j:\ a_{ij} = 0,\ \forall i \tag{5.5}$$

В зависимости от ограничений для переменной x_j и коэффициента целевой функции c_j, переменная x_j находится на одном из ее граничных значениях или вся задача неограниченная.

3) Недостижимая переменная:

$$\exists j:\ l_j > u_j. \tag{5.6}$$

Задача недостижима.

4) Неподвижная переменная:

$$\exists j:\ l_j = u_j. \tag{5.7}$$

Переменную можно убрать из задачи.

5) Единственный ненулевой элемент в строке (единичная строка):

$$\exists (j,k):\ a_{ij} = 0,\ \forall j \neq k,\ a_{ik} \neq 0. \tag{5.8}$$

Из i-ого ограничения можно сделать вывод, что $x_j = b_i/a_{ik}$.

Заметим, что преобразование матрицы может привести к дальнейшим упрощениям. Например, если A – перестановочная нижняя треугольная матрица, то повторение пункта (5.6) решает задачу.

5.4. Единичные столбцы

Определим, как можно использовать единичные столбцы. Единичный столбец, это такой столбец матрицы, удовлетворяющий условию:

$$\exists (j,k):\ a_{ij} = 0,\ \forall i \neq k,\ a_{kj} \neq 0. \tag{5.9}$$

Если одно из ограничений на единичный столбец бесконечно, то единичный столбец налагает ограничение на один из оптимальных лагранжевых множителей. Все эти границы перечислены в Таблице 2. Они будут полезны позже.

6) Открытый единичный столбец:

$$\exists (j,k):\ \left(a_{ij} = 0,\ \forall i \neq k,\ a_{jk} \neq 0\right) \wedge l_j = -\infty\ \wedge u_j = +\infty. \tag{5.10}$$

Можно сделать замену

$$x_j = \frac{b_i - \sum_{p \neq j} a_{ip} x_p}{d_{ik}}$$

l_j	u_j	a_{kj}	z_j^*	y_k^*
$> -\infty$	$+\infty$	> 0	l_j	$-$
$-\infty$	$< +\infty$	< 0	u_j	$-$
$-\infty$	?	> 0	$-$	Неограничена
?	$+\infty$	< 0	$-$	Неограничена
?	$< l_j$	?	$-$	Недостижима

Таблица 2. Ограничения на оптимальные лагранжевы множители y^*, налагаемые единичным столбцом a_{kj}

Сокращение с использованием открытого единичного столбца очень важно, потому что одно ограничение и одна переменная удаляются из задачи без генерации временных переменных в матрице A, хотя целевая функция и подвергается изменениям. Далее мы обсудим два метода для генерации свободных единичных столбцов.

7) Двойное равенство вместе с единичным столбцом [1]

$$\exists i, j, k\colon\ a_{ij}x_j + a_{ik}x_k = b_i,\ j \neq k, a_{ij} \neq 0, a_{ik} \neq 0. \qquad (5.11)$$

Если переменная x_k – единичный столбец, тогда границы для переменной x_j можно изменять так, что достижимая область останется неизменной даже если ограничения для переменной x_k удалить. Впоследствии переменную x_k можно исключить из задачи.

Также возможно, что устанавливать ограничения на единичный столбец – это излишнее действие. Например,

$$x_j - \sum_{k \neq j} x_k = 0. x_j, x_k \geq 0, \qquad (5.12)$$

где x_j – это единичный столбец. Очевидно, что простые ограничения на переменную x_j могут быть удалены без изменения достижимой области, при этом создав открытый единичный столбец. Этот пример может быть обобщен далее.

Каждый раз, когда метод предобработки определяет единичный столбец x_j, этот метод пытается установить, является ли столбец свободным или нет, используя следующую технику. Для каждого $a_{ij} \neq 0$ следующие ограничения на переменную x_j могут вычислены так:

$$u'_{ij} = \begin{cases} \frac{(b_i - \sum_{k \in P_{ij}} a_{ik} l_k - \sum_{k \in M_{ij}} a_{ik} u_k)}{a_{ij}}, & a_{ij} > 0, \\ \frac{(b_i - \sum_{k \in P_{ij}} a_{ik} u_k - \sum_{k \in M_{ij}} a_{ik} l_k)}{a_{ij}}, & a_{ij} < 0, \end{cases} \quad (5.13)$$

и

$$l'_{ij} = \begin{cases} \frac{(b_i - \sum_{k \in P_{ij}} a_{ik} l_k - \sum_{k \in M_{ij}} a_{ik} u_k)}{a_{ij}}, & a_{ij} < 0, \\ \frac{(b_i - \sum_{k \in P_{ij}} a_{ik} u_k - \sum_{k \in M_{ij}} a_{ik} l_k)}{a_{ij}}, & a_{ij} > 0, \end{cases} \quad (5.14)$$

где M_{ij}=$\{k: a_{ik} < 0, j \neq k\}$ и $P_{ij} = \{k: a_{ik} > 0, j \neq k\}$. Можно проверить, что неравенство $l'_{ij} \leq x_j \leq u'_{ij}$ выполняется для любого достижимого решения x в задаче (5.1). Если эти новые ограничения такие же строгие как первоначальные, то переменная называется предполагаемо открытой (implied free)

8) Предполагаемо открытый единичный столбец

$$\exists j, k: \left(a_{ij} = 0, \forall i \neq k, a_{kj} \neq 0\right) \wedge l_j \leq l'_{kj} \leq u'_{kj} \leq u_j. \quad (5.15)$$

Полагается, что l'_{kj} и u'_{kj} вычисляются согласно (5.13) и (5.14). Переменную x_j можно рассматривать как открытый единичный столбец. (см (5.9)).

5.5. Принудительные и мажорируемые ограничения

В этом пункте ограничения на исходные переменные используются для определения принудительных и мажорируемых ограничений.

Чтобы установить, данное ограничение принудительного или мажорируемого типа, мы вычисляем

$$g_i = \sum_{j \in P_i} a_{ij} l_j + \sum_{j \in M_i} a_{ij} u_j \;\text{ и }\; h_j = \sum_{j \in M_i} a_{ij} l_j + \sum_{j \in P_i} a_{ij} u_j, \quad (5.16)$$

где $P_i = \{j: a_{ij} > 0\}$ и $M_i = \{j: a_{ij} < 0\}$. Очевидно, что неравенство

$$g_i \leq \sum_j a_{ij} x_j \leq h_i \quad (5.17)$$

выполняется для любого решения x, которое удовлетворяет простым ограничениям. Возможны следующие варианты:

9) Недостижимое ограничение

$\exists i: h_i < b_i \vee b_i < g_i.$ (5.18)

В этом случае задача является недостижимой.

10) Принудительное ограничение

$\exists i: g_i = b_i \vee h_i = b_i.$ (5.19)

Если $g_i = b_i$, то из-за линейности только одно допустимое значение переменой x_j равно l_j (u_j) при $a_{ij} > 0$ $(a_{ij} < 0)$. Поэтому мы можем проинициализировать все переменные в i-ом ограничении. Подобное утверждение можно сделать для случая, когда $h_i = b_i$.

В предыдущем пункте были указаны (5.13) и (5.14) для вычисления предполагаемых ограничений переменной x_j. Эти ограничения вместе с первоначальными ограничениями могут быть использованы для определения большего количества открытых единичных столбцов.

Чтобы уменьшить вычислительную сложность мажорируемой (dominated constraint procedure) процедуры, мы вычисляем только l'_{ij} и u'_{ij} для каждого $a_{ij} \neq 0$, если либо g_i, либо h_i являются конечными. Более того, g_i и h_i вычисляются в течение проверки принудительного ограничения и они могут быть переиспользованы для вычисления предполагаемых ограничений (implied bounds) без больших вычислительных затрат, используя следующие формулы:

$$u'_{ij} = \begin{cases} \frac{(b_i - g_i)}{a_{ij}} + l_j, & a_{ij} > 0, \\ \frac{(b_i - h_i)}{a_{ij}} + l_j, & a_{ij} < 0, \end{cases} \quad (5.20)$$

и

$$l'_{ij} = \begin{cases} \frac{(b_i - h_i)}{a_{ij}} + u_j, & a_{ij} > 0, \\ \frac{(b_i - g_i)}{a_{ij}} + u_j, & a_{ij} < 0, \end{cases} \qquad (5.21)$$

Заметим, что предполагаемые ограничения (implied bounds) используются для определения большего числа свободных единичных столбцов. В зависимости от контекста, в котором мажорируемые ограничения используются, предполагаемые ограничения могут, конечно, быть использованы для усиления или ослабления ограничений на исходные (primal) переменные. Например, ограничения должны быть усилены, если конкретная задача является задачей смешанного целочисленного программирования. С другой стороны, они могут быть также использоваться для ослабления простых ограничений.

5.6. Принудительные и мажорируемые столбцы.

Мажорируемая процедура (dominated constraint procedure), которая обсуждалась в предыдущем пункте, может быть использована для дуальной задачи.

Первый шаг в такой процедуре заключается в нахождении всех единичных столбцов и использовании их для вычисления предполагаемых (implied) нижней и верхней границ на оптимальные лагранжевы множители y^*. Как это может быть сделано, обсуждалось в пункте 5.3. Пусть $\bar{y}_i$ и $\hat{y}_i$ являются максимальными и минимальными из всех таких ограничений соответственно, тогда мы имеем

$$\bar{y}_i \le y_i^* \le \hat{y}_i, \ \forall i \qquad (5.22)$$

Эти простые ограничения на лагранжевы множители можно использовать для изменения некоторых из начальных (primary) переменных. Пусть

$$e_j = \sum_{i \in P_j} a_{ij}\bar{y}_i + \sum_{i \in M_j} a_{ij}\hat{y}_i \ \text{ и } \ d_j = \sum_{i \in P_j} a_{ij}\hat{y}_i + \sum_{i \in M_j} a_{ij}\bar{y}_i, \qquad (5.23)$$

где $P_j = \{i: a_{ij} > 0\}$ и $M_j = \{i: a_{ij} < 0\}$. Очевидно, что

$$e_j \le \sum_i a_{ij} y_i^* \le d_j. \quad (5.24)$$

Из оптимальности условий (5.2) и (5.24) можно заключить, что

$$c_j - d_j \le z_j^* = c_j - \sum_i a_{ij} y_i^* \le c_j - e_j. \quad (5.25)$$

11) Мажорируемый столбец (dominated):

$$\exists j: c_j - d_j > 0 \vee c_j - e_j < 0 \quad (5.26)$$

Если $c_j - d_j > 0$, то из (5.25) $z_j^* > 0$ и согласно Таблице 1 переменную x_j можно принять равной ее нижней границе. Однако, если $l_j = -\infty$, то задача не ограничена. Аналогичное заключение может быть сделано для случая, если $c_j - e_j < 0$.

12) Слабо мажорируемый столбец (weakly dominated):

$$\exists j \notin S: \ c_j - d_j = 0 \wedge l_j > -\infty, \quad (5.27)$$

$$\exists j \notin S: \ c_j - e_j = 0 \wedge u_j < +\infty, \quad (5.28)$$

где $S = \{j: a_{ij} - \text{единичный столбец при некотором } i\}$. Заметим, что единичные столбцы используются для генерации ограничений d_j и e_j и поэтому их нельзя удалить в этом тесте. В следующем предложении мы докажем, что в случаях (5.27) и (5.28) переменной x_j может быть присвоена нижняя или верхняя граница соответственно.

Предложение 3.1. *Пусть (5.1) имеет оптимальное решение. Если (5.27) или (5.28) выполняются, то тогда существует оптимальное решение, такое, что* $x_j^* = l_j$ *или* $x_j^* = u_j$*, соответственно.*

Доказательство. Пусть утверждение (5.27) выполняется. Более того, положим без потери общности, что $l = 0, u = \infty$ и $j = n$. Тогда двойственная задача для (5.1) такая:

$$\max b^T y$$
$$\text{т. ч } A^T y + s = c, \quad (5.29)$$
$$s \ge 0.$$

Из нашего предположения следует, что (5.29) имеет оптимальное решение. Поэтому существует двойственное оптимальное решение $\bar{x} \in \mathbb{R}^{n-1}$ для задачи (5.24) без последнего ограничения, которое утверждает, что

$$A(\bar{x}, 0) = b \text{ и } \bar{x} \geq 0. \tag{5.30}$$

Очевидно, что $x = (\bar{x}, 0)$ является оптимальным решением для задачи (5.1), доказывающим, что существует оптимальное решение, такое, что $x_j^* = 0 = l_j$ в этом случае. Утверждение для (5.28) аналогичное. ■

12) Принудительный столбец:

$$\exists j \notin S\colon \left(c_j - e_j = 0 \wedge u_j = \infty\right) \vee \left(c_j - d_j = 0 \wedge l_j = -\infty\right). \tag{5.31}$$

Пусть $c_j - e_j = 0$ и $u_j = \infty$, откуда следует, что $0 \leq z_{jj}^* \leq c_j - e_j = 0$, Из построения следует

$$y_k^* = \begin{cases} \bar{y}_k, & a_{kj} > 0. \\ \hat{y}_k, & a_{kj} < 0. \end{cases} \tag{5.32}$$

Если $-y_k^*$ раз k-ое ограничение будет вычтено из целевой функции, когда $a_{kj} \neq 0$, то k-ое ограничение может быть удалено из задачи. Мы не указали этот метод, потому что этот случай происходит крайне редко.

В конечном счете ограничения e_j и d_j используются для вычисления новых ограничений на оптимальные Лагранжевы множители y^* следующим образом. Положим, что $l_j > \infty$ и $u_j = +\infty$. Из оптимальности условий (5.2) следует, что

$$0 \leq z_j^* = c_j - \textstyle\sum_i a_{ij} y_i^*. \tag{5.33}$$

Если $a_{kj} \neq 0$, то мы имеем

$$y_k^* \leq \frac{c_j - e_j}{a_{kj}} + \bar{y}_k, \; a_{kj} > 0, \tag{5.34}$$

или

$$y_k^* \geq \frac{c_j - e_j}{a_{kj}} + \hat{y}_k, \; a_{kj} < 0, \tag{5.35}$$

Подобным образом, если $l_j = -\infty$, мы получаем

$$y_k^* \geq \frac{c_j - d_j}{a_{kj}} + \hat{y}_k, \; a_{kj} > 0 \tag{5.36}$$

и

$$y_k^* \le \frac{c_j - d_j}{a_{kj}} + \bar{y}_k,\ a_{kj} > 0 \qquad (5.37)$$

В случае (6.34) мы имеем $c_j - e_j \ge 0$. Иначе (5.26) имеет место. Замечая, что $(c_j - e_j)/a_{kj} \ge 0$, получаем

$$\bar{y}_k \le y_k^* \le \frac{c_j - e_j}{a_{kj}} + \bar{y}_k, \qquad (5.38)$$

так что новые ограничения на Лагранжевы множители не может быть несовместным. Подобным образом, все ограничения (5.35), (5.37) и (5.36) должны быть совместными.

Новые ограничения на Лагранжевы множители y^*, сгенерированные с помощью (5.34)-(5.37) используются для повторного вычисления e_j и d_j, которые в свою очередь используются для определения большего количества мажорируемых (dominated) столбцов.

5.7. Строки-дубликаты.

Линейно-зависимые строки в матрице A нежелательны, потому что в алгоритме внутренней точки определенная линейная система должна обязательно быть решена в течение каждой итерации. Эта линейная система имеет сингулярную матрицу, если матрица A содержит линейно-зависимые строки.

Одна из простых форм линейно зависимых строк – это когда две строки полностью идентичны, кроме скалярного множителя. Две такие строки называются дубликатами. Общее определение такое:

$$\exists i, k:\ \ a_{ij} = \vartheta a_{kj},\ i \ne k, \forall j \notin S, \qquad (5.39)$$

где $S = \{j: a_{ij} - \text{елиничный столбец для некоторого } i\}$ и ϑ – это скалярный множитель.

Пусть $i-$ая и $k-$ая строки являются строками-дубликатами; по определению мы имеем

$$\begin{cases} \sum_{j\notin S} a_{ij}x_j + \sum_{j\in S} a_{ij}x_j = b_i\,, \\ \sum_{j\notin S} a_{kj}x_j + \sum_{j\in S} a_{kj}x_j = b_k, \\ \quad a_{ij} = \vartheta a_{kj},\ \forall j \notin S. \end{cases} \tag{5.40}$$

Если i-ая строку добавить $-\vartheta$ раз к k-ой, мы получим

$$\begin{cases} \quad \sum_{j\notin S} a_{ij}x_j + \sum_{j\in S} a_{ij}x_j = b_i, \\ \sum_{j\in S} a_{kj}x_j - \vartheta\left(\sum_{j\in S} a_{ij}x_j\right) = b_k - \vartheta b_i. \end{cases} \tag{5.41}$$

Возможны следующие варианты.

14) (a) k-ая строка пустая, (b) k-ая строка содержит предполагаемо открытую переменную (implied free variable) или (c) k-ая строка – это двойное равенство (doubleton equation), содержащее единичный столбец.

Все эти случаи уже были описаны выше.

15) Ограничение k содержит только единичные столбцы:

$$a_{ij} = 0, \forall j \in S. \tag{5.42}$$

В этом случае мы имеем дело с двумя независимыми задачами, которые могут быть решены независимо.

Со столбцами дубликатами рассуждения аналогичны.

6. Линейное программирование.

Исходная задача с ослабленными ограничениями – это задача линейного программирования с той же самой целевой функцией и ограничениями, как и в определении задачи целочисленного программирования, но только без целочисленных ограничений. Пусть x_{LP} является решением для задачи с ослабленными ограничениями, и x – решение начальной задачи с целочисленными ограничениями. Очевидно, что имеет место неравенство:

$$f^T x_{LP} \leq f^T x,$$

потому что x_{LP} минимизирует ту же самую функцию, но с более слабыми ограничениями.

Эта исходная задача с ослабленными ограничениями и все релаксации в течении алгоритма ветвей и границ решаются с помощью обычных методов линейного программирования.

7. Предобработка методами смешанного целочисленного программирования (MILP)

В течение MILP предобработки идет анализ линейных неравенств $A^*x \leq b$ и ограничений, чтобы определить:

· Является ли задача ограниченной

· Могут ли ограничения усилены

· Являются ли неравенства избыточными, поэтому удалены из задачи

· Можно ли целочисленные переменные зафиксировать

Основная цель предобработки MILP заключается в упрощении вычислений алгоритма ветвей и границ. Предобработка включает в себя анализ и устранение подзадач без решений. В противном случае эти подзадачи обрабатывались бы алгоритмом.

7.1 Основные методы предобработки и апробации

Главная идея основных методов улучшения представления смешанной целочисленной задачи $\min\{cx + hy: Ax + Gy \leq b, x \in \{0,1\}^n, y \in RR^m\}$ заключается в анализе каждого неравенства системы неравенств для определения достижимых областей по очереди, чтобы установить: является ли вся задача достижимой; или какие-либо ограничения избыточны; или можно ли какое-нибудь из неравенств использовать для усиления ограничений на переменных; может ли неравенство усилено, изменяя его коэффициенты; следует ли из неравенств. что какие-либо из переменных все время принимают какое-то константное значение, 0 или 1.

Пусть неравенство имеет вид

$$\sum_{j\in B^+} a_j x_j - \sum_{j\in B^-} a_j x_j + \sum_{j\in C^+} g_j y_j - \sum_{j\in C^-} g_j y_j \leq b,$$

где $B = B^+ \cup B^-$ - множество двоичных переменных, $C = C^+ \cup C^-$ множество целых и вещественных переменных, и $a_j > 0$ для $j \in B$ и $g_j > 0$

для $j \in C$. Заметим, что мы не различаем целочисленные и вещественные переменные. Обозначим верхние и нижние ограничения для целых и вещественных чисел как l_j и u_j.

Итак, пусть $Ax + Gy \leq b, x \in \{0,1\}^n, y \in RR^m$ – данное представление задачи смешанного программирования, пусть $a^i x + g^i y \leq b_i$ произвольное неравенство системы, а $A^i x + G^i y \leq b^i$ обозначим системы неравенств, полученную из системы $Ax + Gy \leq b$ удалением строки $a^i x + g^i y \leq b_i$.

7.1.1. Основные методы предобработки

Определение недостижимости

Рассмотрим следующую задачу смешанного целочисленного программирования

$$z = \min \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

при условии

$$A^i x + G^i y \leq b^i$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Если $z > b_i$, то очевидно, что достижимая область пустая. К сожалению, указанную выше задачу смешанного целочисленного программирования тяжело решить в первоначальной формулировке, но очевидно, что каждая нижняя граница для z удовлетворяет неравенству $(z \leq z_{LB} \leq b_i)$. Простейшая, но также самая слабая нижняя граница может быть получена удалением системы $A^i x + G^i y \leq b^i$. В этом случае мы можем заключить, что задача неограниченная, если

$$-\sum_{j \in B^-} a_j^i + \sum_{j \in C^+} g_j^i l_j - \sum_{j \in C^-} g_j^i u_j > b_j.$$

Определение избыточности

Рассмотрим следующую задачу смешанного программирования

$$z = \min \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

при условии

$$A^i x + G^i y \leq b^i$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Если $z < b_i$, то очевидно, что неравенство $a^i x + g^i y \leq b_i$ избыточное. Конечно, указанную выше задачу смешанного программирования тяжело решить в первоначальном виде, но очевидно, что каждая нижняя граница для z удовлетворяет неравенству $(z \leq z_{LB} \leq b_i)$. Простейшая, но также самая слабая нижняя граница может быть получена удалением системы $A^i x + G^i y \leq b^i$. В этом случае мы можем заключить, что неравенство избыточно, если

$$\sum_{j \in B^+} a_j^i + \sum_{j \in C^+} g_j^i u_j - \sum_{j \in C^-} g_j^i l_j \leq b_j.$$

Улучшающие ограничения

Рассмотрим переменную y_k, $k \in C^+$ и следующую целочисленную задачу

$$z_k = min \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+ \setminus \{k\}} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

при условии

$$A^i x + G^i y \leq b^i$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Очевидно, что $y_k \leq (b_i - z_k)/g_k^i$. Более того, верхнее ограничение u_k на переменную y_k, $k \in C^+$ может быть улучшено, если $\frac{(b_i - z_k)}{g_k^i} < u_k$. Когда мы удалим систему $A^i x + G^i y \leq b^i$, то мы можем заключить, что верхняя граница на переменную y_k, $k \in C^+$может быть улучшена, если

$$\frac{\left(b_i + \sum_{j\in B^-} a_j^i - \sum_{j\in C^+\setminus\{k\}} g_j^i\, l_j + \sum_{j\in C^-} g_j^i\, u_j\right)}{g_k^i} < u_k$$

Далее, рассмотрим переменную $y_k, k \in C^-$ и следующую целочисленную задачу

$$z_k = min \sum_{j\epsilon B^+} a_j^i x_j - \sum_{j\in B^-} a_j^i\, x_j + \sum_{j\in C^+} g_j^i\, y_j - \sum_{j\in C^-\setminus\{k\}} g_j^i\, y_j$$

при условии

$$A^i x + G^i y \le b^i$$

$$l \le y \le u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Очевидно, что $y_k \ge (z_k - b_i)/g_k^i$. Поэтому нижняя граница l_k переменной y_k, $k \in C^-$ может быть улучшена, если $\frac{(z_k - b_i)}{g_k^i} > l_k$. Когда мы удалим подсистему $A^i x + G^i y \le b^i$, мы можем заключить, что нижняя граница для переменной $y_k, k \in C^-$ может быть улучшена, если

$$\frac{\left(-\sum_{j\in B^-} a_j^i + \sum_{j\in C^+} g_j^i\, l_j - \sum_{j\in C^-\setminus\{k\}} g_j^i\, u_j - b_i\right)}{g_k^i} > l_k$$

7.1.2 Основные методы апробации

Методы апробации основаны на исследовании логических заключений. Например, предварительно установив бинарной величине значения 0 или 1, мы анализируем результаты. В схеме, представленной выше, это равносильно добавлению равенств $x_k = 0$ или $x_k = 1$ к множеству ограничений и применения методов предобработки к этой расширенной формулировке. Очевидно, что результаты должны интерпретироваться по-разному.

Фиксированные переменные

Рассмотрим бинарную переменную $x_k, k \in B^+$ и следующую задачу смешанного целочисленного программирования:

$$z_k = min \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

при условии

$$A^i x + G^i y \leq b^i$$

$$x_k = 1$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

В случае $z_k > b_i$, очевидно, что достижимая область расширенной формулировки пустая. Поэтому $x_k \neq 1$ для любого допустимого решения исходной формулировки и x_k может быть зафиксирована на значении 0. Когда мы удалим систему $A^i x + G^i y \leq b^i$, мы можем установить переменной x_k значение 0, если

$$a_k^i - \sum_{j \in B^-} a_j^i + \sum_{j \in C^+} g_j^i l_j - \sum_{j \in C^+} g_j^i u_j > b_i.$$

Далее, рассмотрим бинарную переменную $x_k, k \in B^-$ и следующую задачу смешанного целочисленного программирования:

$$z_k = min \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

при условии

$$A^i x + G^i y \leq b^i$$

$$x_k = 0$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

В случае $z_k > b_i$, очевидно, что достижимая область расширенной формулировки пустая. Поэтому $x_k \neq 0$ для любого допустимого решения исходной формулировки и x_k может быть зафиксирована на значении 1. Когда мы удалим систему $A^i x + G^i y \leq b^i$, мы можем присвоить переменной x_k значение 1, если

$$-\sum_{j\in B^-\setminus\{k\}} a_j^i + \sum_{j\in C^+} g_j^i\, l_j - \sum_{j\in C^-} g_j^i\, u_j > b_i.$$

Улучшающие коэффициенты

Рассмотрим бинарную переменную $x_k, k \in B^+$ и следующую расширенную задачу смешанного целочисленного программирования

$$z_k = max \sum_{j\in B^+} a_j^i\, x_j - \sum_{j\in B^-} a_j^i\, x_j + \sum_{j\in C^+} g_j^i\, y_j - \sum_{j\in C^-} g_j^i\, y_j$$

при условии

$$A^i x + G^i y \le b^i$$

$$x_k = 0$$

$$l \le y \le u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Если $z_k \le b_i$, то очевидно, что неравенство $a^i x + g^i y \le b_i$ избыточное в расширенной формулировке. Аналогично, при предположении, что $x_k = 0$, множество допустимых значений не будет затронуто, если из b_i и a_k^i вычесть $\delta = b_i - z_k$. Более того, также при предположении, что $x_k = 1$, множество допустимых значений не будет затронуто, если из b_i и a_k^i вычесть $\delta = b_i - z_k$. Чтобы это увидеть, перепишем неравенство $a^i x + g^i y \le b_i$ как

$$\sum_{j\in B^+\setminus\{k\}} a_j^i\, x_j - \sum_{j\in B^-} a_j^i\, x_j + \sum_{j\in C^+} g_j^i\, y_j - \sum_{j\in C^-} g_j^i\, y_j \le b_i - a_k^i \Rightarrow$$

$$\sum_{j\in B^+\setminus\{k\}} a_j^i\, x_j - \sum_{j\in B^-} a_j^i\, x_j + \sum_{j\in C^+} g_j^i\, y_j - \sum_{j\in C^-} g_j^i\, y_j \le (b_i - \delta) - \left(a_k^i - \delta\right); \delta$$

$$\in RR_+ \Rightarrow$$

$$\sum_{j\in B^+\setminus\{k\}} a_j^i\, x_j + \left(a_k^i - \delta\right)x_k - \sum_{j\in B^-} a_j^i\, x_j + \sum_{j\in C^+} g_j^i\, y_j - \sum_{j\in C^-} g_j^i\, y_j \le b_i - \delta; \delta$$

$$\in RR_+$$

Поэтому a_k и b_i могут быть уменьшены на величину $b_i - z_k$ без изменения множества допустимых решений. Когда мы удалим систему $A^i x + G^i y \leq b^i$, мы можем уменьшить a_k и b_i, если

$$\sum_{j \in B^+ \setminus \{k\}} a_j^i + \sum_{j \in C^+} g_j^i u_j - \sum_{j \in C^-} g_j^i l_j < b_i.$$

Далее, рассмотрим бинарную переменную $x_k, k \in B^+$ и следующую расширенную задачу смешанного целочисленного программирования

$$z_k = max \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

при условии

$$A^i x + G^i y \leq b^i$$

$$x_k = 1$$

$$l \leq y \leq u$$

$$x \in \{0,1\}^n, y \in RR^m.$$

Если $z_k \leq b_i$, то очевидно, что неравенство $a^i x + g^i y \leq b_i$ является избыточным в расширенной формулировке. Далее, при предположении, что $x_k = 1$, множество допустимых решений не изменится, если a_k^i уменьшить на величину $\delta = b_i - z_k$. Также при предположении, что $x_k = 0$, множество допустимых решений не изменится, если a_k^i уменьшить на величину $\delta = b_i - z_k$.

Поэтому a_k можно уменьшить на величину $b_i - z_k$ без изменения множества допустимых решений. Когда мы удалим систему $A^i x + G^i y \leq b^i$, мы можем уменьшить a_k, если

$$\sum_{j \in B^+} a_j^i - a_k^i + \sum_{j \in C^+} g_j^i u_j - \sum_{j \in C^-} g_j^i l_j < b_i.$$

7.1.3. Реализация

Основные метода предобработки и апробации могут быть реализованы очень эффективно. Пусть

$$L^i_{max} = \sum_{j \in B^+} a^i_j + \sum_{j \in C^+} g^i_j u_j - \sum_{j \in C^-} g^i_j l_j$$

и

$$L^i_{min} = -\sum_{j \in B^-} a^i_j + \sum_{j \in C^+} g^i_j l_j - \sum_{j \in C^-} g^i_j u_j,$$

то есть рассматриваются максимальная и минимальная величины суммы на левой стороне неравенства. Не трудно увидеть, что основные методы предобработки и апробации требуют следующих тестов

- Определение достижимости:

$$L^i_{min} > b_i$$

- Определение избыточности:

$$L^i_{max} \leq b_i$$

- Улучшение ограничений:

$$\frac{\left(b_i - \left(L^i_{min} - g^i_k l_k\right)\right)}{g^i_k} < u_k, \qquad y_k, k \in C^+$$

$$\frac{\left(\left(L^i_{min} + g^i_k u_k\right) - b_i\right)}{g^i_k} > l_k, \qquad y_k, k \in C^-$$

- Фиксация переменных:

$$L^i_{min} + a^i_k > b_i, \qquad x_k, k \in B^+ \cup B^-$$

- Улучшение коэффициентов:

$$L^i_{max} - a^i_k < b_i, \qquad x_k, k \in B^+ \cup B^-$$

Очевидно, что все эти тесты могут быть проведены за константное время, когда значения L^i_{min} и L^i_{max} вычислены. Как результат, применение основных методов предобработки и апробации ко всем неравенствам системы $Ax +$

$Gy \leq b$ требует $O(n)$ времени, где n – общее число всех ненулевых элементов в системе.

8. Построение сечений

Сечения – это дополнительные линейные ограничения в виде неравенств, которые добавляются в задачу. Цель этих неравенств заключается в ограничении области допустимых решений для релаксации линейной задачи таким образом, что решения будут ближе к целым числам.

8.1 Определения

Далее мы будет следовать определениям, указанным ниже. Задачи смешанного программирования, это задачи в форме

$$\max cx + hy$$
$$Ax + Gy \leq b$$
$$x \geq 0 \text{ целое}$$
$$y \geq 0,$$

где данные представляют собой строчные векторы с $\in \mathbb{Q}^n, h \in \mathbb{Q}^p$, матрицы $A \in \mathbb{Q}^{m \times n}$,$G \in \mathbb{Q}^{m \times p}$ и столбцевой вектор $b \in \mathbb{Q}^m$; переменные – столбцевые векторы $x \in \mathbb{R}^n$ и $y \in \mathbb{R}^p$. Множество S допустимых решений для задачи смешанного программирования называется смешанным линейным множеством. Если $P \coloneqq \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Ax + Gy \leq b\}$, то $S \coloneqq P \cap (\mathbb{Z}^n_+ \times \mathbb{R}^p_+)$. Можно показать, что $conv(S)$ – это полиэдр. Поэтому чтобы решать задачу смешанного программирования, то для этого достаточно решить линейную задачу

$$\max cx + hy$$

$$(x, y) \in conv(S),$$

так как крайние точки множества $conv(S)$ также являются крайними и для множества S. Это полиэдральный подход**.** Основная сложность, конечно, заключается в построении неравенств, определяющих множество $conv(S)$, получаемое из множества P. Далее будет представлены подходы для построения улучшенных аппроксимаций множества $conv(S)$ рекурсивно. Говорят, что неравенство удовлетворяет множеству, если оно удовлетворяется в каждой точке множества. Сечение относительно точки $(\bar{x}, \bar{y}) \notin conv(S)$ – это неравенство, удовлетворяющее множеству $conv(S)$, которое нарушается в точке $(\bar{x}, \bar{y})$. Полиэдральный подход к решению задач смешанного программирования ведет естественным образом к подходу сечения плоскости: Решить линейную релаксацию, игнорирующую требования целостности для вектора x; если эта релаксация не ограничена или недостижима, то тогда мы останавливаемся: смешанная задача недостижима или не ограничена; иначе пусть $(\bar{x}, \bar{y})$ оптимальная крайняя точка решения; если $(\bar{x}, \bar{y}) \in S$, то останавливаемся: $(\bar{x}, \bar{y})$ – оптимальное решение для смешанной задачи; иначе генерируем новое сечение к точке $(\bar{x}, \bar{y})$, и добавляем его в предыдущую линейную релаксацию, решаем улучшенную релаксацию и повторяем алгоритм снова.

8.2. Полиэдр

Полиэдр в $\mathbb{R}^n$ – множество в форме $P \coloneqq \{x \in \mathbb{R}^n : Ax \leq b\}$, где A – вещественная матрица, а b является вещественным вектором. Если A и b состоят из рациональных элементов, P называется рациональным полиэдром**.** Полиэдр $\{x \in \mathbb{R}^n : Ax \leq 0\}$ называется полиэдральным конусом. Заметим, что полиэдральный конус все время непустой, так как он содержит нулевой вектор.

Для $S \subseteq \mathbb{R}^n$, выпуклая оболочка множества S – множество $conv(S) \coloneqq \{x \in \mathbb{R}^n : x = \sum_{i=1}^{k} \lambda_i x^i$ где $k \geq 1, \lambda \in \mathbb{R}_+^k, \sum_{i=1}^{k} \lambda_i = 1$ и $x^1, \dots, x^k \in S\}$. Это наименьшее выпуклое множество, содержащее множество S. Иногда бывает удобно работать с $\overline{conv}(S)$, замыкание множества $conv(S)$, которое является наименьшим замкнутым выпуклым множеством, содержащим S. Коническая оболочка непустого множества $S \subseteq \mathbb{R}^n$ - $cone(S) \coloneqq \{x \in \mathbb{R}^n : x = \sum_{i=1}^{k} \lambda_i x^i$ где $k \geq 1, \lambda \in \mathbb{R}_+^k$ и $x^1, \dots, x^k \in S\}$.

Выпуклая оболочка конечного множества точек множества $\mathbb{R}^n$ называется политопом.

Важная теорема Минковского-Вейля заключает, что каждый полиэдр P можно представить в виде суммы политопа Q и полиэдрального конуса C. Здесь $Q + C \coloneqq \{x \in \mathbb{R}^n : x = y + z$ для некоторых $y \in Q$ и $z \in C\}$. Заметим, что $P = \emptyset$ тогда и только тогда, когда $Q = \emptyset$. Если $P \coloneqq \{x \in \mathbb{R}^n : Ax \leq b\}$ непустой, то тогда $C = \{x \in \mathbb{R}^n : Ax \leq 0\}$ единственный. Конус $\{x \in \mathbb{R}^n : Ax \leq 0\}$ называется рецессивным конусом политопа P.

Ещё один важный результат леммы Фаркаша : для непустого полиэдра $P \coloneqq \{x : Ax \leq b\}$ неравенство $\alpha x \leq \beta$ удовлетворяет полиэдру P тогда и только тогда, когда существует вектор $v \geq 0$, такой, что $vA = \alpha$ и $vb \leq \beta$. [Это также результат двойственности линейного программирования $\beta \geq max\{\alpha x : Ax \leq b\} = \min\{vb : vA = \alpha, v \geq 0\}$.] Еще одна форма леммы Фаркаша : $Ax \leq b$ имеет решение тогда и только тогда, когда $vb \geq 0$ для всех $v \geq 0$, таких, что $vA = 0$. [Эквивалентно, по двойственности линейного программирования, $max\{0 : Ax \leq b\} = min\{vb : vA = 0, v \geq 0\}$.]

Доказательства этих утверждений можно найти в учебниках, таких, как в [3] и [4].

8.2.1. Некоторые свойства полиэдров и выпуклых множеств

Эта секция содержит две леммы, которые будут использованы в будущем.

Лемма 1. Пусть $P \coloneqq \{x \in \mathbb{R}^n : Ax \leq b\}$ полиэдр и $\Pi \coloneqq P \cap \{x : \pi x \leq \pi_0\}$. Если $\Pi \neq \emptyset$ и $\alpha x \leq \beta$ удовлетворяет множеству Π, тогда существует скалярное значение $\lambda \in \mathbb{R}_+$, такое, что неравенство

$$\alpha x - \lambda(\pi x - \pi_0) \leq \beta$$

удовлетворяет множеству P.

Доказательство. По лемма Фаркаша, так как $\Pi \neq \emptyset$, то существуют $u \geq 0$, $\lambda \geq 0$, такие, что

$$\alpha = uA + \lambda\pi$$

$$\text{и } \beta \geq ub + \lambda\pi_0.$$

Так как $uAx \leq ub$ удовлетворяет множеству P, то $uAx \leq \beta - \lambda\pi_0$. Так как $uAx = \alpha x - \lambda\pi x$, то неравенство $\alpha x - \lambda(\pi x - \pi_0) \leq \beta$ удовлетворяет множеству P. ■

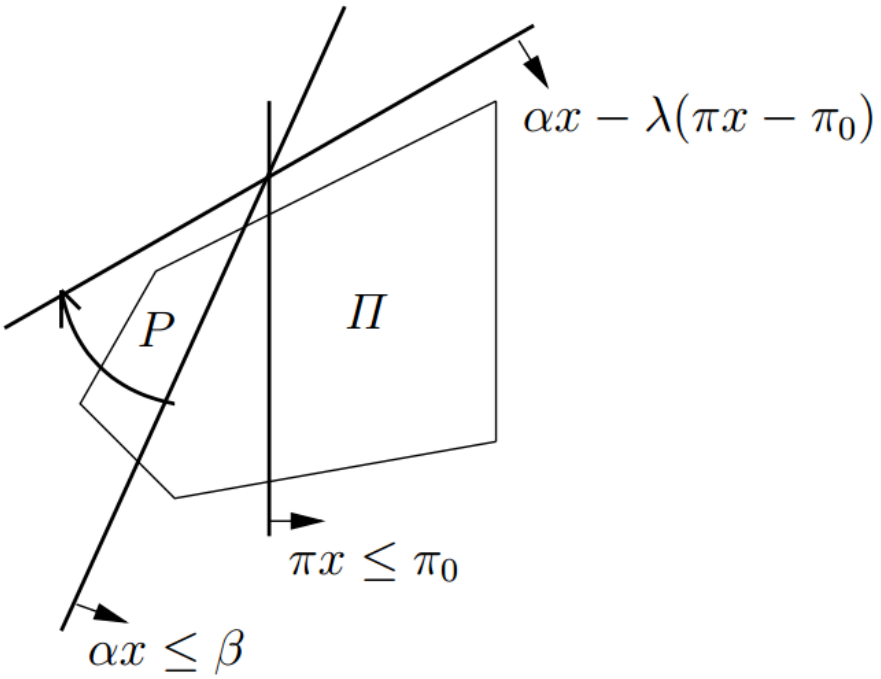

Рисунок 2. Иллюстрация леммы 1

Замечание 1. Из леммы 1 следует два возможных результата: Когда гиперплоскость $\pi x = \pi_0$ параллельна к $\alpha x = \beta$, то, выбрав значение λ, такое, что $\alpha = \lambda\pi$, мы получаем тривиальное неравенство $\lambda\pi_0 \leq \beta$, которое

выполняется во всех точках множества $\mathbb{R}^n$ (например, лемма выполняется, когда $P = \mathbb{R}^n$); в более интересном случае, когда $\pi x = \pi_0$ не параллельна к $\alpha x = \beta$, результат леммы – замкнутое полу-пространство, содержащее P (см. Рисунок 2).

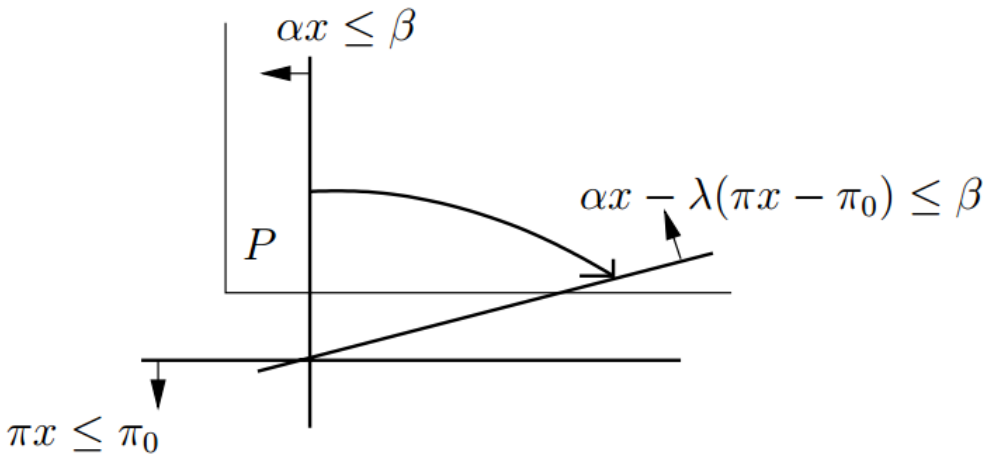

Рисунок 3. Лемма 1 не выполняется, когда $\Pi = \emptyset$

Замечание 2. Условие $\Pi = \emptyset$ необходимо для леммы 1, как показано в следующем примере (см. Рисунок 3) : $P := \{x \in \mathbb{R}^2 : x_1 \geq 0, x_2 \geq 0\}$ и $\Pi := P \cap \{x : x_2 \leq -1\}$. Поэтому множество Π пустое. Неравенство $x_1 \leq 1$ удовлетворяет Π, но не такого скалярного значения λ, такого, что неравенство $x_1 - \lambda(x_2 + 1) \leq 1$ удовлетворяет множеству P.

Пусть $H \subseteq \mathbb{R}^n$ является гиперплоскостью и $S \subseteq \mathbb{R}^n$. В общем $conv(S) \cap H \neq conv(S \cap H)$, как показано в следующем примере: S состоит из двух точек, не содержащихся в множестве H, но отрезок линии, их соединяющей, пересекает H. Следующая лемма показывает, что равенство выполняется, когда S расположено полностью в одном из замкнутых полупространств, определяемых гиперплоскостью H (см. Рисунок 4).

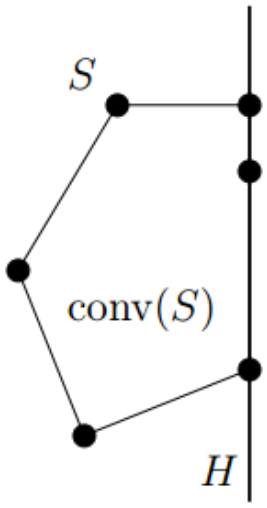

Рисунок 4. Иллюстрация леммы 2

Лемма 2. Пусть $H \coloneqq \{x \in \mathbb{R}^n : ax = b\}$ – гиперплоскость и $S \subseteq \{x : ax \leq b\}$. Тогда $conv(S) \cap H = conv(S \cap H)$.

Доказательство. Сначала докажем, что $conv(S \cap H) \subseteq conv(S) \cap H$ и затем $conv(S) \cap H \subseteq conv(S \cap H)$.

Очевидно, что $conv(S \cap H) \subseteq conv(S)$ и $conv(S \cap H) \subseteq H$, поэтому первое включение очевидно.

Чтобы доказать второе включение, предположим, что некоторый вектор $x \in conv(S) \cap H$. Это значит, что $ax = b$ и $x = \sum_{i=1}^{k} \lambda_i x^i$, где $x^1,\ldots,x^k \in S$, $\lambda \geq 0$ и $\sum_{i=1}^{k} \lambda_i = 1$.

$$b = ax = \sum_{i=1}^{k} \lambda_i a x^i \leq \sum_{i=1}^{k} \lambda_i b = b \qquad (8.1)$$

где неравенство следует из $ax^i \leq b$ и $\lambda_i \geq 0$. Отношение (8.1) означает, что эти неравенства на самом деле являются равенствами, то есть $ax^i = b$ для $i = 1, \ldots k$. Поэтому $x^i \in S \cap H$. А значит $x \in conv(S \cup H)$. ■

8.2.2. Фасеты

Пусть $P \coloneqq \{x \in \mathbb{R}^n : Ax \leq b\}$ – полиэдр. Грань множества P – это множество в виде

$$F \coloneqq P \cap \{x \in \mathbb{R}^n : \alpha x = \beta\}$$

где $\alpha x \leq \beta$ удовлетворяет множеству P (говорят, что неравенство определяет грань F). Сама грань является полиэдром, так как она есть пересечение полиэдра P с другим полиэдром (гиперплоскость $\alpha x = \beta$). Размерность множества – это размерность наименьшего афинного пространства, которое его содержит. Грани размерности 0 называются вершинами множества P; размерности 1 называются ребрами**,** а грани размерности $\dim(P) - 1$ называются фасетами. Доказательство следующей теоремы можно найти в [3].

Теорема 1. Пусть $P \subseteq \mathbb{R}^n$ является непустым полиэдром.

1) Для каждой фасеты F множества P , по крайней мере, одно из неравенств, определяющих F, необходимо для любого описания $Ax \leq b$ множества P.
2) Неравенства, определяющие грани с размерностями, меньшими, чем $\dim(P) - 1$, не нужны для описания множества P и могут быть удалены.

Из результата следует, что если полиэдр в множестве $\mathbb{R}^n$ имеет m фасет, то любое представление в виде линейных неравенств в $\mathbb{R}^n$ содержит, по крайней мере, m неравенств. В целочисленном программировании мы часто рассматриваем полиэдр в виде множества, заданного неявно как $conv(S)$. Для таких полиэдров необычно иметь множество фасет, число которых экспоненциально зависит от размера входных значений. Поэтому их представление в виде линейных неравенств в множестве $\mathbb{R}^n$ трудоемко. В некоторых случаях можно обойти это затруднение: полиэдр с большим количеством фасет иногда можно получить как проекцию полиэдра с

меньшим числом фасет. По этой причине проекции приобретают важную роль.

8.2.3. Проекции

Пусть $P \subseteq \mathbb{R}^{n+p}$, где $(x, y) \in P$ интерпретируется как $x \in \mathbb{R}^n$ и $q \in \mathbb{R}^p$. **Проекция** множества P на x-пространство $\mathbb{R}^n$ есть

$$proj_x(P) \coloneqq \{x \in \mathbb{R}^n : \exists y \in \mathbb{R}^p, \text{такой что } (x, y) \in P\}.$$

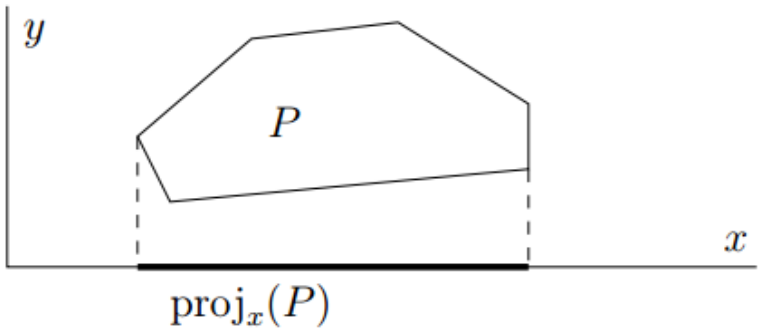

Рисунок 5. Проекция

Теорема 2. Пусть $P \coloneqq \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : Ax + Gy \leq b\}$. Тогда $proj_x(P) = \{x \in \mathbb{R}^n : v^t(b - Ax) \geq 0 \text{ для всех } t \in T\}$, где $\{v^t\}_{t \in T}$ является множеством крайних лучей для $Q \coloneqq \{v \in \mathbb{R}^m : vG = 0, v \geq 0\}$.

Доказательство. Пусть $x \in \mathbb{R}^n$. По лемме Фаркаша $Gy \leq b - Ax$ имеет решение y тогда и только тогда, когда $v^t(b - Ax) \geq 0$ для всех крайних лучей v^t множества $vG = 0, v \geq 0$. ■

Перечисление крайних лучей множества Q может быть непростой задачей в приложениях. Еще один способ получения проекции для P заключается в удалении переменных y_i: удаление одной переменной за один проход процедуры (процедура удаления Фурье-Моцкина):

Рассмотрим полиэдр $P \subseteq \mathbb{R}^{n+1}$, определяемый системой неравенств:

$$\sum_{j=1}^{n} a_{ij}x_j + g_j z \leq b_i \text{ для } i \in I. \qquad (8.2)$$

Пусть $I^0 = \{i \in I: g_i = 0\}, \quad I^+ = \{i \in I: g_i > 0\}, \quad I^- = \{i \in I: g_i < 0\}.$ Процедура Фурье-Моцкина удаляет переменную z следующим образом: она хранит неравенства из (8.2) в I^0, и объединяет неравенства $i \in I^+$ и $l \in I^-$ для удаления z.

Теорема 3. Система $|I^0| + |I^+||I^-|$ неравенств, полученных с помощью процедуры Фурлье-Моцкина является проекцией $proj_x(P)$ для P в x-пространстве $\mathbb{R}^n$.

Доказательство. Из (2) следует

$$z \leq \frac{1}{g_i}\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right) \forall i \in I^+$$

$$z \geq \frac{1}{g_l}\left(b_l - \sum_{j=1}^{n} a_{lj}x_j\right) \forall l \in I^-.$$

Поэтому каждый элемент (x, y) в (2) удовлетворяет неравенству

$$\frac{1}{g_l}\left(b_l - \sum_{j=1}^{n} a_{lj}x_j\right) \leq \frac{1}{g_i}\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right) \tag{8.3}$$

Обратно, если x удовлетворяем неравенствам (2) для $i \in I^0$и системе (8.3) для каждого $i \in I^+, l \in I^-$, тогда

$$max_{l \in I^-}\frac{1}{g_l}\left(b_l - \sum_{j=1}^{n} a_{lj}x_j\right) \leq min_{i \in I^+}\frac{1}{g_i}\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right).$$

Пусть *x* является любым значением из интервала. Тогда (x, z) удовлетворяет неравенству (8.2). ■

8.2.4. Объединение полиэдров

В этой секции мы докажем результат Баласа [6], [7] об объединении полиэдров. Рассмотрим k полиэдров $P_i := \{x \in \mathbb{R}^n: A_i x \leq b^i\}$. Мы покажем,

что наименьшее замкнутое выпуклое множество, содержащее $\bigcup_{i=1}^{k} P_i$, является полиэдром. Это множество обозначается как $\overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$.

Замкнутость необходима, это показано в следующем примере: P_1 состоит из одной точки, P_2 является прямой, которая не содержит точку P_1

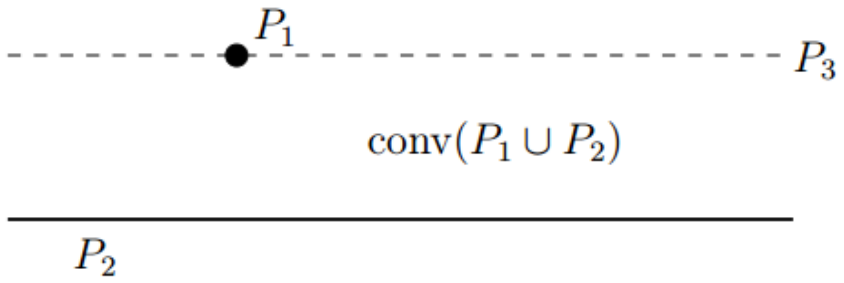

Рисунок 6. $\overline{conv}\left(\bigcup_{i=1}^{k} P_i\right) \neq conv\left(\bigcup_{i=1}^{k} P_i\right)$

(Рисунок 6). Пусть P_3 – это прямая, идущая через точку P_1 параллельно к P_2. Легко проверить, что $\overline{conv}(P_1 \cup P_2) = conv(P_2 \cup P_3)$ и что $conv(P_1 \cup P_2) = conv(P_2 \cup P_3)\backslash(P_3\backslash P_1)$ (на самом деле точка x^* в $P_3\backslash P_1$ не принадлежит множеству $conv(P_1 \cup P_2)$, но имеется последовательность точек $x^k \in conv(P_1 \cup P_2)$, стремящихся к x^*). Здесь $conv(P_1 \cup P_2)$ не является замкнутым множеством, поэтому не является полиэдром.

Следующий пример покажет, что $\overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$ может иметь экспоненциальное число фасет относительно размера входных данных $P_i, i = 1, \dots, k$.

Пусть e^i – i-ый единичный вектор в $\mathbb{R}^n$. Пусть P_i является точкой e^i для $i = 1, \dots, n$, и P_{n+i} – точка $-e^i$ для $i = 1, \dots, n$. Тогда $\overline{conv}\left(\bigcup_{i=1}^{2n} P_i\right)$ имеет 2^n фасет:

$$\overline{conv}\left(\bigcup_{i=1}^{2n} P_i\right) = \left\{x \in \mathbb{R}^n : \sum_{i=1}^{n} \epsilon_i x_i \leq 1 \text{ для всех } \epsilon \in \{-1, +1\}^n\right\}.$$

Такой полиэдр называется октаэдром**.**

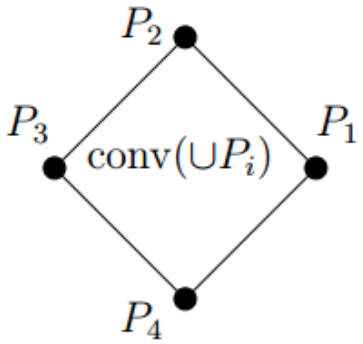

Рисунок 7. Октаэдр

Хотя $\overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$ может иметь экспоненциальное число фасет, Балас [6], [7] доказал, что это множество является проекцией полиэдра Y более высокой размерности с представлением полиномиального размера:

$$Y \coloneqq \begin{cases} A_i x^i \leq b^i y_i \\ \sum x^i = x \\ \sum y_i = 1 \\ y_i \geq 0 \text{ для } i = 1, \dots, k \end{cases} \tag{8.4}$$

В этой формулировке x^i является вектором в $\mathbb{R}^n$ и y_i скаляр, для $i = 1, \dots, k$. Вектор $x \in \mathbb{R}^n$ соответствует изначальному пространству, в которое Y проектируется. Потому полиэдр Y определен в $\mathbb{R}^{kn+n+k}$. Говорят, что формулировка с полиномиальным числом переменных и ограничений сжатая**.** Смысл результата Баласа заключается в том, что $\overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$ обладает сжатым представлением. Доказательство дано ниже.

Пусть $P_i \coloneqq \left\{x \in \mathbb{R}^n : A_i x \leq b^i\right\}$ полиэдр и $C_i \coloneqq \{x \in \mathbb{R}^n : A_i x \leq 0\}$ его конус. По теореме Минковского-Вейля существует политоп, такой, что $P_i = Q_i + C_i$.

Теорема 4. Рассмотрим k полиэдров $P_i \coloneqq \left\{x \in \mathbb{R}^n : A_i x \leq b^i\right\}$ и $C_i \coloneqq \{x \in \mathbb{R}^n : A_i x \leq 0\}$. Пусть $P_i := Q_i + C_i$, где Q_i является политопом. Если $\bigcup P_i \neq \emptyset$, положим, что $C_j \subseteq cone \cup_{i:P_i \neq \emptyset} C_i$ для всех j, таких, что $P_j = \emptyset$. Тогда следующие множества равны.

(i) $\overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$

(ii) $conv\left(\bigcup_{i=1}^{k} Q_i\right) + cone\left(\bigcup_{i=1}^{k} C_i\right)$

(iii) $proj_x Y$, где Y опеределено в (9.4).

Доказательство. Определим $Q \coloneqq conv\left(\bigcup_{i=1}^{k} Q_i\right)$ и $C \coloneqq cone\left(\bigcup_{i=1}^{k} C_i\right)$. Мы покажем, что $conv\left(\bigcup_{i=1}^{k} P_i\right) \subseteq proj_x Y \subseteq Q + C \subseteq \overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$.

Так как $proj_x Y$ замкнутое множество, то первое включение дает $\overline{conv}\left(\bigcup_{i=1}^{k} P_i\right) \subseteq proj_x Y$ и теорема доказана.

(a) $conv\left(\bigcup_{i=1}^{k} P_i\right) \subseteq proj_x Y$

Результат выполняется, если $\bigcup_{i=1}^{k} P_i = \emptyset$, поэтому мы будем считать, что $\bigcup_{i=1}^{k} P_i \neq \emptyset$. Без потери общности, $P_1, \dots, P_h$ непустые и $P_{h+1}, \dots, P_k$ пустые, где $1 \leq h \leq k$.

Пусть $x \in conv\left(\bigcup_{i=1}^{k} P_i\right)$. Тогда x является выпуклой комбинацией конечного числа точек в $\bigcup_{i=1}^{h} P_i$. Так как P_i выпуклое, мы можем записать x в виде выпуклой комбинации точек $z^i \in P_i$, скажем $x = \sum_{i=1}^{h} y_i z^i$, где $y_i \geq 0$ для $i = 1, \dots, h$ и $\sum_{i=1}^{h} y_i = 1$. Пусть $x^i = y_i z^i$ для $i = 1, \dots, k$ и $x = \sum_{i=1}^{k} x^i$. Это показывает, что $x \in proj_x Y$ и поэтому (a) выполняется.

(b) $proj_x Y \subseteq Q + C$

Утверждение выполняется, если $Y = \emptyset$, поэтому будем считать, что $Y \neq \emptyset$.

Пусть $x \in proj_x Y$. По определению проекции, существуют $x^1, \dots, x^k, y$, такие, что $x = \sum_{i=1}^{k} x^i$, где $Ax^i \leq b^i y_i, \sum y_i = 1, y \geq 0$. Пусть $I \coloneqq \{i : y_i > 0\}$.

Для любого $i \in I$ обозначим $z^i \coloneqq \frac{x^i}{y^i}$. Тогда $z^i \in P_i$. Так как $P_i = Q_i + C_i$, то мы можем записать $z^i = w^i + x^i$, где $w^i \in Q_i$ и $x^i \in C_i$.

$$x = \sum_{i \in I} y_i z^i + \sum_{i \notin I} x^i = \underbrace{\sum_{i \in I} y_i w^i}_{\in conv(\cup Q_i)} + \underbrace{\sum_{i=1}^{k} \lambda_i x^i}_{\in cone(\cup C_i)}, \text{ где } \lambda_i \geq 0 \ \in Q + C$$

(c) $Q + C \subseteq \overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$

Утверждение выполняется, когда $Q = \emptyset$. Без потери общности $Q_1, \dots, Q_h$ непустые и $Q_{h+1}, \dots, Q_k$ пустые, где $1 \leq h \leq k$.

Пусть $x \in Q + C$. Тогда $x = \sum_{i=1}^{h} y_i w^i + \sum_{i=1}^{k} x^i$, где $w^i \in Q_i, y_i \geq 0$ для $i = 1, \dots, h$, $\sum_{i=1}^{h} y_i = 1$ и $x^i \in C_i$ для $i = 1, \dots, k$. По предположению теоремы, мы имеем для $j = h+1, \dots, k$, $x^j = \sum_{i=1}^{h} \mu_i^j x^{ij}$, где $\mu_i^j \geq 0$ и $x^{ij} \in C_i$ для $i = 1, \dots, h$.

Используя вектор $y \in \mathbb{R}_+^h$, введем $I := \{i: y_i > 0\}$ и рассмотрим точку

$$x^* := \sum_{i \in I} \left(y_i - \frac{h}{|I|} \epsilon \right) w^i + \sum_{i=1}^{h} \epsilon \left(w^i + \frac{1}{\epsilon} \left(x^i + \sum_{j=h+1}^{k} \mu_i^j x^{ij} \right) \right)$$

для $\epsilon > 0$, достаточно малого для того, чтобы $y_i - \frac{h}{|I|} \epsilon \geq 0$ для всех $i \in I$.

$x^\epsilon \in conv\left(\bigcup_{i=1}^{k} P_i\right)$, так как $\sum_{i \in I} \left(y_i - \frac{h}{|I|} \epsilon \right) + \sum_{i=1}^{h} \epsilon = 1$ и $w^i + \frac{1}{\epsilon}\left(x^i + \sum_{j=h+1}^{k} \mu_i^j x^{ij} \right) \in P_i$.

Более того, $x^\epsilon \to x$ при $\epsilon \to 0$.

Поэтому $x \in \overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$. ■

Замечание 4. Утверждение теоремы необходимо, как показано в следующем примере (см. Рисунок 8): $P_1 := \{x \in \mathbb{R}^2: 0 \leq x \leq 1\}$ и $P_2 := \{x \in \mathbb{R}^2: x_1 \leq 0, x_1 \geq 1\}$. Заметим, что $P_2 = \emptyset$ и $C_2 = \{x \in \mathbb{R}^2: x_1 = 0\}$. Теорема не выполняется в этом случае, так как $proj_x Y = P_1 + C_2 = \{x \in \mathbb{R}^2: 0 \leq x_1 \leq 1\}$, что отличается от $\overline{conv}(P_1 \cup P_2) = P_1$

Замечание 5. Утверждение теоремы 4 автоматически выполняется, если

(i) $C_i = \{0\}$ каждый раз, когда $P_i = \emptyset$, или

(ii) все полиэдры P_i имеют тот же самый конус.

Например, (i) выполняется, когда все P_i непустые, или когда $C_i = \{0\}$ для всех i.

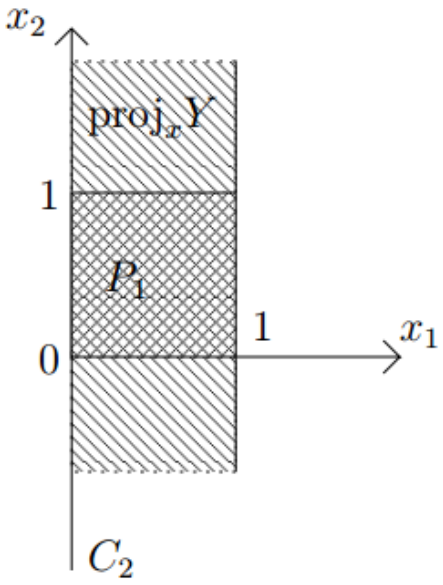

Рисунок 8. $\overline{conv}(P_1 \cup P_2) \neq proj_x Y$

Замечание 6. Если все полиэдры P_i имеют тот же самый конус $C = C_i$, $i = 1, \dots, k$, то $conv\left(\bigcup_{i=1}^{k} P_i\right) = \overline{conv}\left(\bigcup_{i=1}^{k} P_i\right)$. Действительно, в этом случае часть (c) в доказательстве выше упрощается и показывает, что $Q + C \subseteq conv\left(\bigcup_{i=1}^{k} P_i\right)$. Множество $\bigcup P_i$ - x −проекция для

$$Y_i := \begin{cases} A_i x^i \leq b^i y_i \\ \sum x^i = x \\ \sum y_i = 1 \\ y_i \in \{0,1\}, i = 1, \dots, k \end{cases}$$

Чтобы доказать последнее утверждение, заметим, что если $x \in \bigcup P_i$, то $x \in P_j$ для $j = 1, \dots, k$. Установив $x^j = x, y_j = 1$ и $x^i = 0, y_i = 0$ для $i \neq j$ мы имеем решение $(x, x^1, \dots, x^k, y) \in Y_I$. Обратно, пусть $(x, x^1, \dots, x^k, y) \in Y_I$, и j – это индекс, при котором $y_j = 1$. Мы имеем $A_j x^j \leq b^j$ и $A_i x^i \leq 0$ для $i \neq j$, а именно, что $x^j \in P_j$ и x^i принадлежит конусу полиэдра P_j, точка $x = \sum_{i=1}^{k} x^i$ принадлежит P_j.

Замечание 7. Когда полиэдры P_j имеют различные рецессивные конусы, x-проекция для Y_I обычно отличается от $\bigcup P_i$, как показано в следующем примере (см. Рисунок 6).

Пусть $P_1 \subseteq \mathbb{R}^2$ состоит из единственной точки $(0,1)$, и пусть $P_2 \subseteq \mathbb{R}^2$ является прямой $x_2 = 0$. Тогда мы можем утверждать, что x-проекция для Y_I – это $P_2 \cup P_3$, где P_3 обозначает прямую $x_2 = 1$. Действительно, когда $y_1 = 1$ в Y_I, $x = x^1 + x^2$ с $x^1 \in P_1$ и x^2 принадлежит рецессивному конусу полиэдра P_2. Так что x-проекция для Y_I, ограниченная прямой $y_1 = 1$, сама является прямой P_3. x-проекция для Y_I, ограниченная прямой $y_2 = 1$, сама является прямой P_2.

По сути из предложения теоремы 4 мы имеем

$$proj_x Y_I = \bigcup_{i=1}^{k} Q_i + cone\left(\bigcup_{i=1}^{k} C_i\right)$$

8.3. Подъем и проекция (Lift-and-Project)

В этой секции мы рассмотрим смешанные 0,1 линейные задачи. Это смешанные целочисленные задачи, где целым числам разрешено принимать только два возможных значения 0 или 1. Будет удобно писать 0,1 линейные задачи в форме

$$\min cx$$
$$Ax \geq b$$
$$x_j \in \{0,1\} \text{ для } j = 1, \dots, n$$
$$x_j \geq 0 \text{ для } j = n+1, \dots, n+p,$$

где матрица $A \in Q^{m\times(n+p)}$, строчный вектор $c \in Q^{n+p}$ и столбцевой вектор $b \in Q^m$ – данные, и $x \in \mathbb{R}^{n+p}$ – столбцевой вектор переменных.

Рассмотрим полиэдр $P \coloneqq \{x \in \mathbb{R}_+^{n+p} : Ax \geq b\}$ и смешанное 0,1 линейное множество $S \coloneqq \{x \in \{0,1\}^n \times \mathbb{R}_+^p : Ax \geq b\}$. Множество $conv(S)$ является

полиэдром и, было бы лучшим вариантом, если бы имели его линейное описание в форме $conv(S) = Dx \geq d$. Такое описание $Dx \geq d$ сократило бы решение смешанной 0,1 линейной задачи к тому, которое можно получить, решая обычную линейную задачу. Но эта цель слишком амбициозна в общем случае, так как число неравенств, необходимых для описания $Dx \geq d$, обычно большое. Это неудивительно, так как смешанная 0,1 линейная задача NP-трудная. Более разумная цель заключается в получении промежуточного множества между P и $conv(S)$, в котором мы можем найти оптимальное решение за полиномиальное время, а дальше использовать рекурсию для получения более близких аппроксимаций множества $conv(S)$.

Шерали и Адамс [8], Ловаз и Шривер [9] и Балас, Цериа и Корнужолс [10] предлагают подход, согласно которому конструируются промежуточные множества Q между P и $conv(S)$ как проекции множеств большей размерности, имеющих полиномиальное описание. Полиэдр $P \subseteq \mathbb{R}^{n+p}$ сначала смещается в пространство более высокой размерности $\mathbb{R}^{n+p+q}$, где приобретает новую формулировку. Затем этот полиэдр проецируется обратно в изначальное пространство $\mathbb{R}^{n+p}$, определяя Q. В этом процессе ограничения в формулировке более высокой размерности определяются явно, в то время как ограничения для Q только известны неявно через проекцию, поэтому разрешая множеству Q иметь неполиномиальное число ограничений. Этот подход называется **подъемом и проекцией (Lift and Project).** Далее будем этот метод обозначать двумя буквами ПП.

8.3.1. ПП релаксация

Рассмотрим полиэдр $P \coloneqq \{x \in \mathbb{R}_+^{n+p} : Ax \geq b\}$ и смешанное 0,1 множество $S \coloneqq \{x \in \{0,1\}^n \times \mathbb{R}_+^p : Ax \geq b\}$. Без потери общности, мы можем считать, что ограничения $Ax \geq b$ включают $x_j \geq 0$ для $j = 1, \dots, n+p$ и $x_j \leq$

1 для $j = 1, \dots, n$. Балас, Церия и Корнужолс [10] рассматривают следующую ЛП процедуру:

Шаг 0: Выбрать $j \in \{1, \dots, n\}$.

Шаг 1: Сконструировать нелинейную систему $x_j(Ax - b) \geq 0$, $\big(1 - x_j\big)(Ax - b) \geq 0$

Шаг 2: Линеаризовать систему с помощью замены y_i на $x_i x_j, i \neq j$, и x_j для x_j^2. Обозначим полученное множество через M_j.

Шаг 3: Проекция множества M_j в x-пространство. Пусть P_j – полиэдр, который получится в результате.

$S \subseteq P_j$ следует из того факта, что для любого $x \in S$ мы имеем $(x, y) \in M_j$, выбрав $y_i = x_i x_j$ для $i \neq j$, так как $x_j^2 = x_j$ (это справедливо, так как x_j – 0,1 переменная). Мы также имеем $P_j \subseteq P$, так как $Ax \geq b$ получено добавлением ограничений, определяющих M_j. Насколько сильна релаксация P_j множества S в сравнении с начальной релаксацией P? Следующая теорема показывает, что эта релаксация самая сильная из всех релаксаций, игнорирующих полноту все переменных x_i для $i \neq j$.

Теорема 5. $P_j = conv\{\big(Ax \geq b, x_j = 0\big) \cup \big(Ax \geq b, x_j = 1\big)\}$

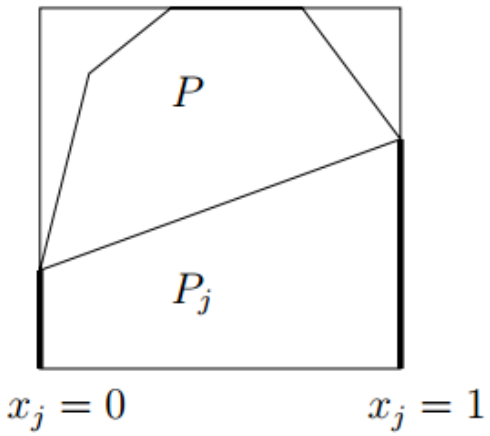

Рисунок 9. Иллюстрация к теореме 5.

Доказательство. Обозначим через P^* множество $conv\{\big(Ax \geq b, x_j = 0\big) \cup \big(Ax \geq b, x_j = 1\big)\}$.

Сначала докажем, что $P_j \subseteq P^*$.

Мы будем считать, что $P \neq \emptyset$, так как в противном случае результат тривиальный.

Если $P \cap \{x_j = 0\} = \emptyset$, то $P^* = P \cap \{x_j = 1\}$. Мы уже знаем, что $P_j \subseteq P$. Поэтому чтобы показать, что $P_j \subseteq P^*$, достаточно показать, что $P_j \subseteq \{x_j = 1\}$, то есть $x_j \geq 1$ справедливо для P_j. Пусть $\epsilon := min\{x_j : x \in P\}$. Так как $P \cap \{x_j = 0\} = \emptyset$, то $\epsilon > 0$. Неравенство $x_j \geq \epsilon$ справедливо для P и, по лемме Фаркаша, выражается в виде комбинаций строк $Ax \geq b$. Поэтому неравенство $(1 - x_j)x_j \geq (1 - x_j)\epsilon$ справедливо для нелинейной системы из Шага 1. Шаг 2 заменяет x_j^2 на x_j. Отсюда $x_j \geq 1$ справедливо для P_j.

Подобным образом, если $P \cap \{x_j = 1\} = \emptyset$, то $P_j \subseteq P^*$.

Теперь положим, что $P \cap \{x_j = 0\} \neq \emptyset$ и $P \cap \{x_j = 1\} \neq \emptyset$. Возьмем неравенство $\alpha x \geq \beta$, выполняющееся в P^*. Так как оно выполняется в $H := P \cap \{x : x_j \leq 0\}$, мы можем найти λ, такой, что $\alpha x + \lambda x_j \geq \beta$ выполняется в P (Лемма 1). Подобным образом, мы можем найти μ, такой, что $\alpha x + \lambda x_j \geq \beta$ выполняется в P.

Поэтому $(1 - x_j)(\alpha x + \lambda x_j - \beta) \geq 0$ и $x_j(\alpha x + \mu(1 - x_j) - \beta) \geq 0$ выполняются в нелинейной системе из Шага 1, и их сумма также

$$\alpha x + (\lambda + \mu)(x_j - x_j^2) - \beta \geq 0.$$

Шаг 2 заменяет x_j^2 на x_j. Откуда получаем $\alpha x \geq \beta$, справедливое для M_j, и поэтому для P_j. Это завершает доказательство того, что $P_j \subseteq P^*$.

Теперь докажем, что $P^* \subseteq P_j$. Будем считать, что $P^* \neq \emptyset$, так как иначе результат тривиальный. Пусть $\bar{x}$ является точкой в $P \cap \{x_j = 0\}$ или в $P \cap \{x_j = 1\}$. Обозначим $\bar{y}_i = \bar{x}_i \bar{x}_j$ для $i \neq j$. Тогда $(\bar{x}, \bar{y}) \in M_j$, так как $\bar{x}_j^2 = \bar{x}_j$. Из выпуклости P_j следует, что $P^* \subseteq P_j$. ■

Множество $\bigcap_{j=1}^{n} P_j$ называется **ПП замыканием.** Это более лучшая аппроксимация множества $conv(S)$, чем P:

$$conv(S) \subseteq \bigcap_{j=1}^{n} P_j \subseteq P.$$

8.3.2 ПП сечения

Оптимизация линейной функции во множестве P_j ведет к решению задачи линейного программирования. Поэтому задача заключается в генерации неравенства, удовлетворяющего P_j и отсекающего данную точку $\bar{x}$, или в том, чтобы показать, что такого сечения не существует. Шаг 2 процедуры ПП из секции 9.3.3 конструирует следующий полиэдр M_j:

$$M_j \coloneqq \left\{x \in \mathbb{R}_+^{n+p}, y \in \mathbb{R}_+^{n+p-1} : Ay - bx_j \geq 0, Ax + bx_j - Ay \geq b, y_j = x_j\right\}.$$

Первые два ограничения получаются из линеаризации неравенств Шага 1. По существу, переменная y_j не вводится явно в Шаге 2. Поэтому в действительности M_j является полиэдром в $\mathbb{R}_+^{n+p} \times \mathbb{R}_+^{n+p-1}$, полученным определением переменных $y_j = x_j$ в вышеуказанном множестве. Пусть A_j является $m \times (n + p - 1)$ матрица, полученная из матрицы A удалением j-ый столбец матрицы a^j. Мы имеем

$$M_j \coloneqq \left\{x \in \mathbb{R}_+^{n+p}, y \in \mathbb{R}_+^{n+p-1} : A_j y + \left(a^j - b\right)x_j \geq 0, Ax + \left(b - a^j\right)x_j - Ay \geq b\right\}.$$

Переименовав матричные коэффициенты перед x, мы получим

$$M_j = \left\{x \in \mathbb{R}_+^{n+p}, y \in \mathbb{R}_+^{n+p-1} : \tilde{B}_j x + A_j y \geq 0, \tilde{A}_j x - A_j y \geq b\right\}$$

Мы хотим спроецировать переменные y. Это может быть сделано с помощью Теоремы 2. Соответствующий конус

$$Q \coloneqq \left\{(u, v) : uA_j - vA_j = 0, u \geq 0, v \geq 0\right\}.$$

То есть множество P_j может быть записано в виде:

$$P_j = \left\{x \in \mathbb{R}_+^{n+p} : \left(u\tilde{B}_j + v\tilde{A}_j\right)x \geq vb \text{ для всех } (u, v) \in Q\right\}.$$

Пусть дано решение $\bar{x}$, мы теперь можем показать, что неравенство $\alpha x \geq \beta$ выполняется в P_j и отсекает $\bar{x}$. Действительно, $\alpha = u\tilde{B}_j + v\tilde{A}_j$ и $\beta = vb$ для $(u, v) \in Q$ означает, что $\bar{x}$ удовлетворяет P_j, а неравенство $\alpha\bar{x} < \beta$ говорит, что это является сечением. Чтобы получить более глубокое сечение, мы должны решить следующую линейную задачу, так называемую ЛП, генерирующую сечения (cut generating LP - CGLP):

$$\max vb - \left(u\tilde{B}_j + v\tilde{A}_j\right)\bar{x}$$

$$uA_j - vA_j = 0$$

$$u \geq 0, v \geq 0.$$

Эти ограничения вместе с ограничением на нормализацию для отсечения конуса будут генерировать сечения. Поэтому CGLP имеет $2m$ переменных и $n + p$ ограничений, отличных от условий не отрицательности. Это очень большая задача для того, чтобы просто сконструировать одно сечение. Множество способов используется для ускорения решения, таких как работа в подпространстве переменных, где $\bar{x}_j > 0$ для $j = 1, \ldots, n + p$ и $\bar{x}_j < 1$ для $j = 1, \ldots, n$, [10]. Возможно ли конструирование ПП сечений без явной формулировки и решения CGLP? Балас и Перегаард [11] дают положительный ответ. Мы это объясним ниже.

Вместо того, чтобы выразить P_j как проекцию M_j, есть возможность представить P_j, используя теоремы 5 и 4 и затем проецировать в x-пространство. По теореме 5 множество P_j является выпуклой оболочкой объединения двух полиэдров:

$$Ax \geq b$$

$$x \geq 0$$

$$-x_j \geq 0$$

и

$$Ax \geq b$$

$$x \geq 0$$

$$x_j \geq 1$$

Далее, мы предположим, что неравенства $Ax \geq b$ содержат $-x_j \geq -1$ для $j = 1, \dots, n$, но не выполняется $x \geq 0$. По теореме 4,

$$P_j = proj_x \begin{cases} Ax^0 \geq by_0 \\ -x_j^0 \geq 0 \\ Ax^1 \geq by_1 \\ x_j^1 \geq y_1 \\ x^0 + x^1 = x \\ y_0 + y_1 = 1 \\ x, x^0, x^1, y_0, y_1 \geq 0. \end{cases}$$

Пусть e_j означает j-ый единичный вектор. Используя теоремы проекций (Теорема 2), мы получим, что P_j определяется неравенствами $\alpha x \geq \beta$, такими, что

$$\alpha - uA + u_0 e_j \geq 0$$

$$\alpha - vA - v_0 e_j \geq 0$$

$$\beta - ub \leq 0$$

$$\beta - vb - v_0 \leq 0$$

$$u, u_0, v, v_0 \geq 0$$

Добавив ограничение для нормализации, мы получим CGLP:

$$\min \alpha \bar{x} - \beta$$

$$\alpha - uA + u_0 e_j \geq 0$$

$$\alpha - vA - v_0 e_j \geq 0$$

$$\beta - ub \leq 0$$

$$\beta - vb - v_0 \leq 0$$

$$\sum_{i=1}^{m} u_i + u_0 + \sum_{i=1}^{m} v_i + v_0 = 1$$

$$u, u_0, v, v_0 \geq 0$$

9. Метод ветвей и границ

9.1. Разделяй и властвуй

Рассмотрим задачу:

$$z = max\{cx: x \in S\}$$

Как мы можем разделить задачу на более мелкие, так, чтобы решить каждую из них по отдельности, а потом соединить информацию вместе для решения исходной задачи?

Предложение 10.1. Пусть $S = S_1 \cup \ldots \cup S_K$ разделение множества S на более мелкие множества, и пусть $z^k = max\{cx: x \in S_k\}$ для $k = 1, \ldots, K$. Тогда $z = max_k z^k$

Простейший способ представить подход разделяй и властвуй - через дерево перечисления. Например, если $S \subseteq \{0,1\}^3$, то мы могли бы сконструировать дерево перечисления, показанное на рисунке 10.

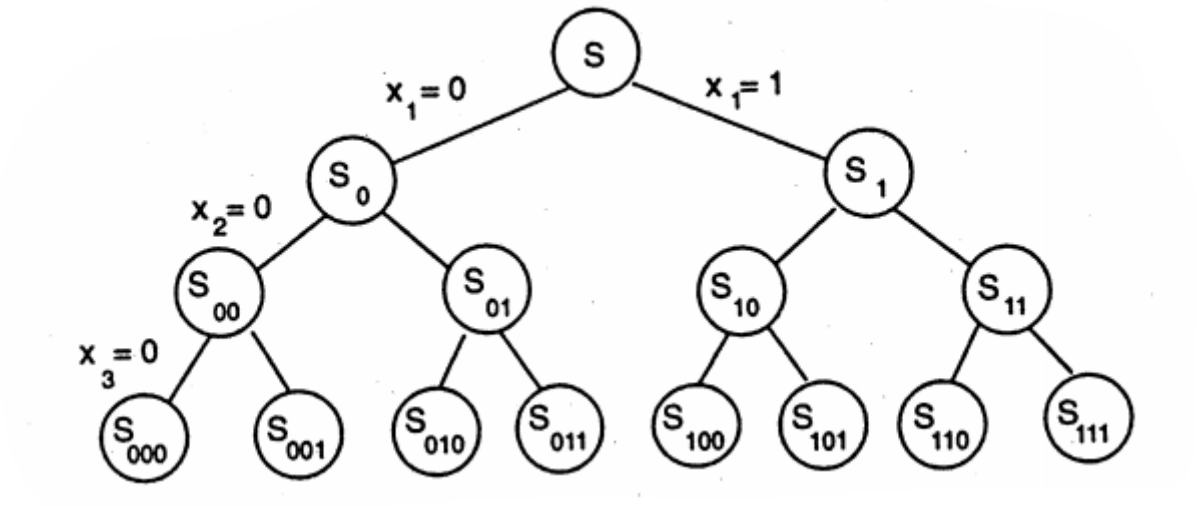

Рисунок 10. Бинарное дерево перечисления

Здесь мы сначала делим S на $S_0 = \{x \in S: x_1 = 0\}$ и $S_1 = \{x \in S: x_1 = 1\}$, затем $S_{00} = \{x \in S_0, x_2 = 0\} = \{x \in S: x_1 = x_2 = 0\}$, $S_{01} = \{x \in S_0, x_2 = 1\}$ и так далее. Заметим, что лист дерева $S_{i_1 i_2 i_3}$ непустой тогда и только тогда, если $x = (i_1, i_2, i_3)$ принадлежит S. Поэтому листья дерева соответствуют в точности точкам бинарного дерева B^3, по которому можно произвести

перечисление. Заметим, что для удобства мы считаем, что корень дерева находится вверху.

Еще один пример перечисления – задача перечисления всех путей коммивояжёра. Сначала мы делим множество S всех путей в 4 города на $S_{(12)}$, $S_{(13)}$, $S_{(14)}$, где $S_{(ij)}$ – множество всех путей, содержащих дугу (ij). Затем $S_{(12)}$ делится снова на $S_{(12)(23)}$ и $S_{(12)(24)}$ и так далее. Процесс показан на рисунке 11.

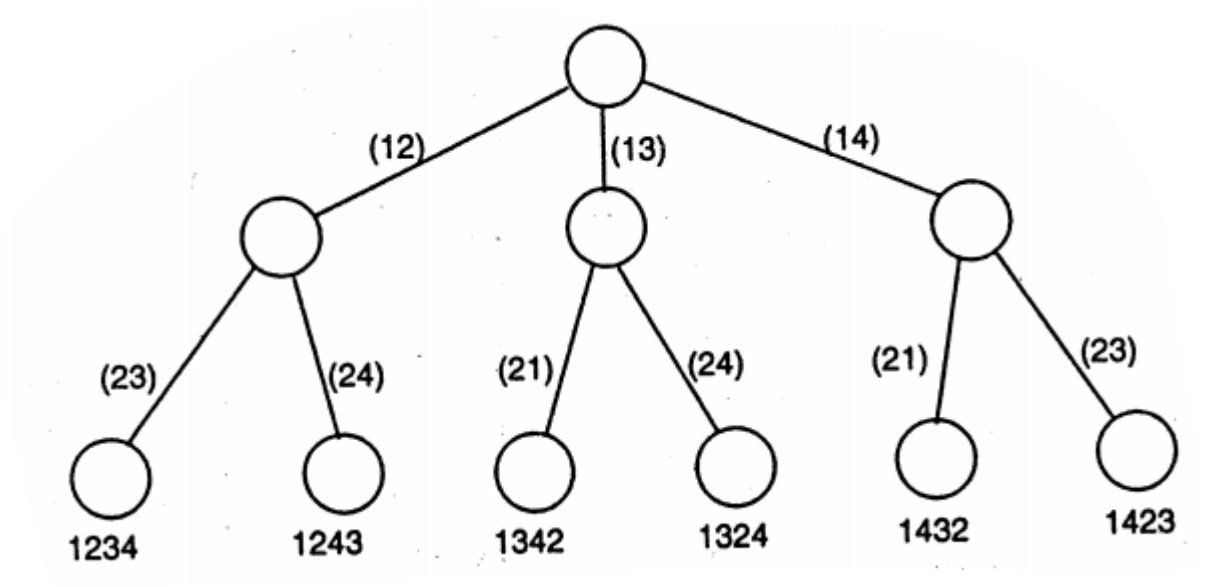

Рисунок 11. Дерево перечисления

Здесь шесть листов дерева соответствуют $(n+1)!$ Показанных путей, где $i_1 i_2 i_3 i_4$ означает, что города посещались в порядке i_1, i_2, i_3, i_4, i_1 соответственно. Заметим, что это пример многоходового ветвления, а не бинарного, где множество может делиться на более чем две части.

9.2. Неявное перечисление

Мы видели в пункте 10.1, что полное перечисление совершенно невозможно для большинства задач, так как число переменных в целочисленной задаче, или вершин в графе практически всегда превосходит число 20 или 30. Поэтому нам нужно сделать больше, чем просто делить неограниченное

число раз. Как мы можем использовать ограничения на значения $\{z^k\}$ с умом? Прежде всего, как мы можем соединить вместе информацию об ограничениях?

Предложение 10.2. Пусть $S = S_1 \cup \ldots \cup S_K$ является разделением множества S на более мелкие множества, и $z^k = max\{cx: x \in S_k\}$ для $k = 1, \ldots, K, \bar{z}^k$ – верхняя граница для z^k и $\underline{z}^k$ является нижней границей для z^k. Тогда $\overline{z} = max_k \overline{z}^k$ верхняя граница для z, а $\underline{z} = max_k \underline{z}^k$ является нижней границей для z.

Теперь мы рассмотрим три примера о том, как ограниченная или частичная информация о подзадаче может быть использована. Какой вывод можно сделать о нижней и верхней гранях оптимального значения z и какие множества нужны для дальнейшего исследования, чтобы получить оптимальное значение.

Пример 1. На рисунке 12 мы видим декомпозицию(разделение) множества S на два множества S_1 и S_2 и нижних и верхних границ соответствующих задач.

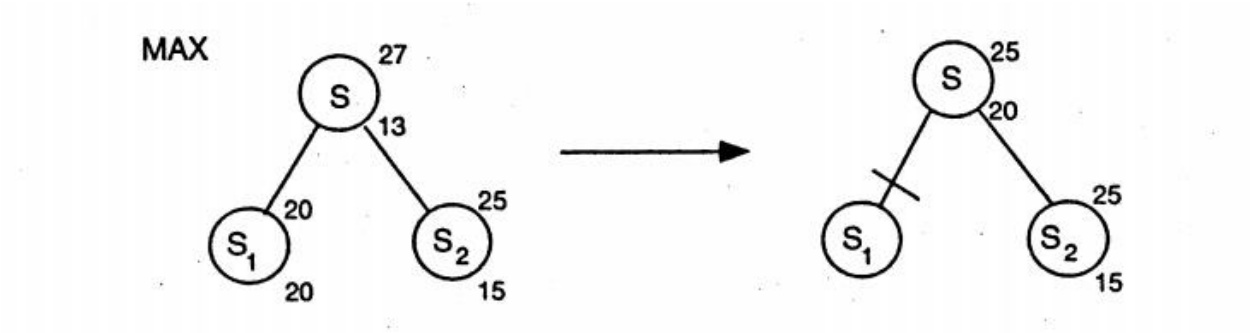

Рисунок 12. Удаление из-за оптимальности

Мы заметим сначала, что $\overline{z} = max_k \overline{z}^k = max\{20{,}25\} = 25$ и $\underline{z} = max_k \underline{z}^k = max\{20{,}15\} = 20$.

Затем, мы видим, что нижняя и верхняя грани для z_1 равны, $z_1 = 20$, и больше нет необходимости рассматривать множество S_1. Поэтому ветвь S_1 дерева перечисления может быть удалена **из-за оптимальности**. ■

Пример 2. На рисунке 13 мы снова делим множество S на два множества S_1 и S_2 и находим верхние и нижние множества соответствующих задач.

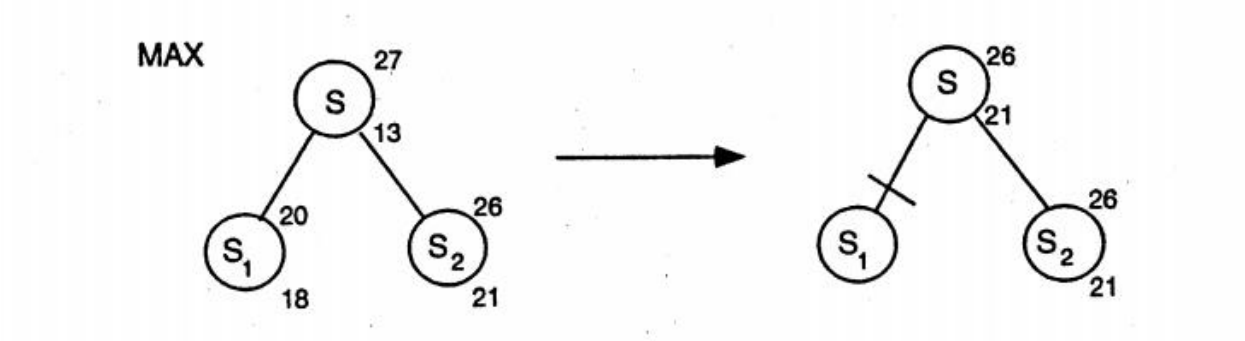

Рисунок 13. Удаление из-за ограниченности

Заметим сначала, что $\overline{z} = max_k \overline{z}^k = max\{20,26\} = 26$ и $\underline{z} = max_k \underline{z}^k = max\{18,21\} = 21$.

Далее, мы видим, что так как оптимальная величина имеет значение не меньше 21, и верхняя граница $\overline{z}^1 = 20$, в множестве S_1 не может быть оптимального решения. Поэтому ветка S_1 дерева перечисления может быть удалена **из-за ограниченности**. ■

Пример 3. На рисунке 14 мы снова делим множество S на два подмножества S_1 и S_2 с различными верхними и нижними границами.

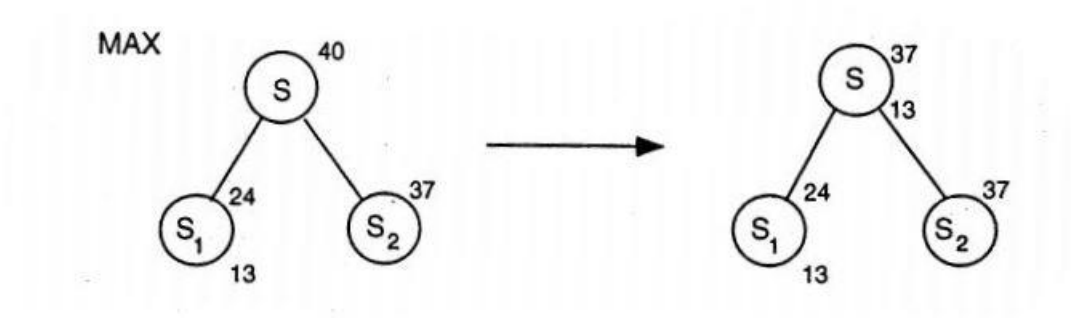

Рисунок 14. Ветки не могут быть удалены

Заметим сначала, что $\overline{z} = max_k \overline{z}^k = max\{24{,}37\} = 37$ и $\underline{z} = max_k \underline{z}^k = max\{13, -\} = 13$. Здесь мы не можем сделать определенный вывод, поэтому необходимо далее исследовать множества S_1 и S_2.

Основываясь на трех примерах, мы можем перечислить по меньшей мере три причины, которые позволяют подрезать дерево и поэтому перечислять большое число решений неявно.

(i) Удаление из-за оптимальности: $z^t = \{\max cx: x \in S_t\}$ решена

(ii) Удаление из-за ограничений: $\overline{z}^t \leq \underline{z}$.

(iii) Удаление из-за неограниченности: $S_t = \emptyset$.

9.3 Метод ветвей и границ: пример

Наиболее общий случай решения целочисленных задач заключается в использовании неявного перечисления, или метода ветвей и границ, в котором линейные релаксации обеспечивают ограничения.

$$z = \max 4x_1 - x_2$$
$$7x_1 - 2x_2 \leq 14$$
$$x_2 \leq 3$$
$$2x_1 - 2x_2 \leq 3$$
$$x \in Z_+^2.$$

Ограничения. Чтобы получить первую верхнюю границу, мы добавим фиктивные переменные x_3, x_4, x_5 и решим линейную релаксацию, в которой ограничения на целочисленность переменных удалены. Полученное оптимальное представление:

$$\overline{z} = \max \frac{59}{7} - \frac{4}{3}x_3 - \frac{1}{7}x_4$$
$$x_1 + \frac{1}{7}x_3 + \frac{2}{7}x_4 = \frac{20}{7}$$
$$x_2 + x_4 = 3$$

$$-\frac{2}{7}x_3 + \frac{10}{7}x_4 + x_5 = \frac{23}{7}$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

Поэтому мы получаем верхнюю границу $\overline{z} = \frac{59}{7}$, и нецелочисленное решение $(\overline{x}_1, \overline{x}_2) = \left(\frac{20}{7}, 3\right)$. Есть ли какой-нибудь прямой способ найти достижимое решение? Очевидно, что нет. Будем считать, что если нет достижимого решения, то мы считаем, что нижняя граница $\underline{z} = -\infty$.

Ветки. Теперь так как $\underline{z} < \overline{z}$, то нам нужно делить множество допустимых решений. Но как мы будем это делать? Одна простая идея заключается в том, что мы выбираем целое и дробное значения в линейном решении и делим задачу на две относительно дробного значения. Если $x_j = \overline{x}_j \notin Z^1$, мы получим:

$$S_1 = S \cap \{x: x_j \leq \lfloor \overline{x}_j \rfloor\}$$

$$S_2 = S \cap \{x: x_j \geq \lceil \overline{x}_j \rceil\}$$

Очевидно, что $S = S_1 \cup S_2$ и $S_1 \cap S_2 = \emptyset$. Еще одна причина для таково выбора в том, что решение $\overline{x}$ для $LP(S)$ недостижимо либо в $LP(S_1)$, либо в $LP(S_2)$. Это означает, что если нет вырождения (например, несколько оптимальных решений линейной задачи), то $max\{\overline{z}_1, \overline{z}_2\} < \overline{z}$, так что верхняя граница будет строго уменьшаться.

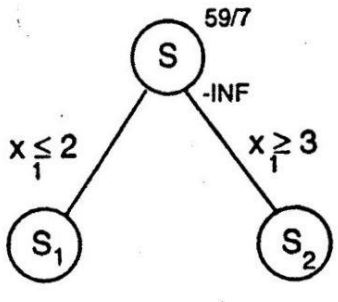

Рисунок 15. Частичный метод ветвей и границ, дерево 1

Так как $\overline{x}_1 = 20/7 \notin Z^1$, мы имеем $S_1 = S \cap \{x: x_1 \leq 2\}$ и $S_2 = S \cap \{x: x_1 \geq 3\}$. Теперь у нас дерево, показанное на рисунке 15. Подзадачи (узлы), которые должны быть проверены, называются **активными.**

Выбор узла. Список активных задач (узлов), который сейчас проверяется, содержит S_1, S_2. Возьмем S_1 (произвольно).

Реоптимизация. Как нам следует решать новую измененную линейную задачу $LP(S_1)$ для $i = 1,2$, не начиная с нуля?

Так как мы только что добавили одно ограничение на верхнюю и нижнюю границы для линейной задачи, то наш предыдущий базис остается двойственно достижимым, и поэтому естественно реоптимизировать из этого базиса, используя двойственный симплекс алгоритм. Обычно несколько (немного) проходов необходимо для нахождения оптимального решения линейной задачи.

Применив это к линейной задаче $LP(S_1)$, мы можем написать новое ограничение $x_1 \geq 2$ как $x_1 + s = 2, s \geq 0$, которое может быть переписано в терминах не базовых переменных как

$$-\frac{1}{7}x_3 - \frac{2}{7}x_4 + s = -\frac{6}{7}.$$

Поэтому получим двойственное достижимое представление:

$$\overline{z}_1 = \max \frac{59}{7} - \frac{4}{7}x_3 - \frac{1}{7}x_4$$

$$x_1 + \frac{1}{7}x_3 + \frac{2}{7}x_4 = \frac{20}{7}$$

$$x_2 + x_4 = 3$$

$$-\frac{2}{7}x_3 + \frac{10}{7}x_4 + x_5 = \frac{23}{7}$$

$$-\frac{1}{7}x_3 - \frac{2}{7}x_4 + s = -\frac{6}{7}$$

$$x_1, x_2, x_3, x_4, x_5, s \geq 0$$

После двух симплекс проходов линейная задача реоптимизирована:

$$\overline{z}_1 = \max \frac{15}{2} - \frac{1}{2}x_5 - 3s$$

$$x_1 + s = 2$$

$$x_2 - \frac{1}{2}x_5 + s = \frac{1}{2}$$

$$x_3 - x_5 - 5s = 1$$

$$x_4 + \frac{1}{2}x_5 + 6s = \frac{5}{2}$$

$$x_1, x_2, x_3, x_4. x_5, s \geq 0$$

с $\overline{z}_1 = \frac{15}{2}$, и $\left(\overline{x}_1^1, \overline{x}_2^1\right) = \left(2, \frac{1}{2}\right)$.

Ветки. S_1 не может быть удалено, поэтому используя то же самое правило ветвления, как и прежде, мы создаем два новых узла $S_{11} = S_1 \cap \{x: x_2 \leq 0\}$ и $S_2 = S_1 \cap \{x: x_2 \geq 1\}$, и добавим их в список узлов. Теперь дерево такое, как показано на рисунке 16.

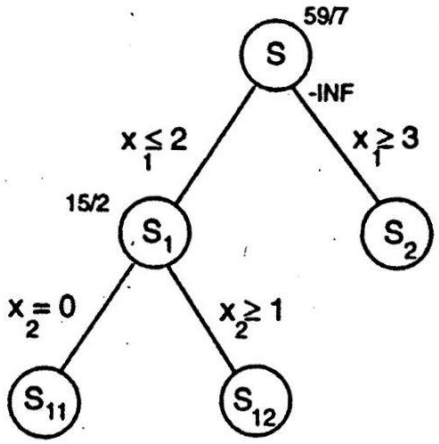

Рисунок 16. Частичный метод ветвей и границ, дерево 2

Выбор узла. Активный список узлов теперь содержит S_2, S_{11}, S_{12}. Произвольно выбрав S_2, мы удалим его из списка узлов и проверим его более детально.

Реоптимизация. Чтобы решить $LP(S_2)$, мы используем двойственный симплекс алгоритм точно таким же способом, как и раньше. Ограничение $x_1 \geq 3$, записанное в виде $x_1 - t = 3, t \geq 0$, выраженное в терминах не базовых переменных, становится:

$$\frac{1}{7}x_3 + \frac{2}{7}x_4 + t = -\frac{1}{7}.$$

Из анализа этого ограничения, мы можем заключить, что линейная задача

$$\overline{z}_2 = \max \frac{59}{7} - \frac{4}{7}x_3 - \frac{1}{7}x_4$$

$$x_1 + \frac{1}{7}x_3 + \frac{2}{7}x_4 = \frac{20}{7}$$

$$x_2 + x_4 = 3$$

$$-\frac{2}{7}x_3 + \frac{10}{7}x_4 + x_5 = \frac{23}{7}$$

$$\frac{1}{7}x_3 + \frac{2}{7}x_4 + t = -\frac{1}{7}$$

$$x_1, x_2, x_3, x_4. x_5, t \geq 0$$

недостижима, $\overline{z}_2 = -\infty$, и следовательно узел S_2 можно удалить **из-за недостижимости**.

Выбор узла. Список узлов теперь содержит S_{11}, S_{12}. Произвольным образом выбрав узел S_{12}, удалим его из списка.

Реоптимизация. $S_{12} = S \cap \{x: x_1 \leq 2, x_2 \geq 1\}$. Полученная в результате линейная задача имеет оптимальное решение $\overline{x}^{12} = (2,1)$ со значением 7. Так как $\overline{x}^{12}$ – целое число, то $z^{12} = 7$.

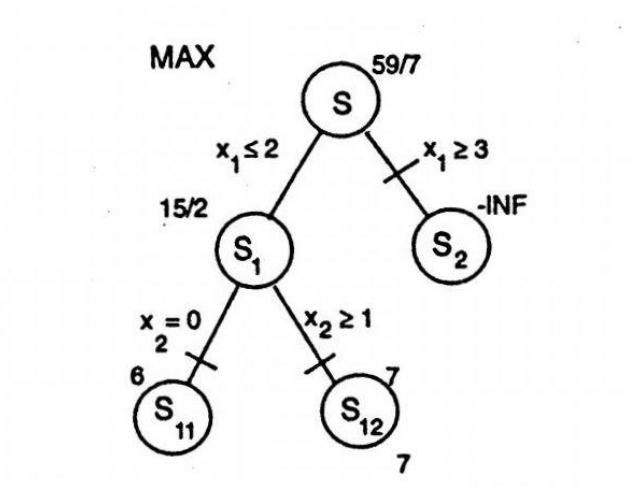

Рисунок 17. Полное дерево, метод ветвей и границ

Обновление текущего оптимального решения. Так как решение задачи $LP(S_{12})$ – целое, мы обновляем значение лучшего допустимого найденного решения $\underline{z} \leftarrow \max\{\underline{z}, 7\}$ и сохраняем соответствующее решение $(2,1)$. S_{12} теперь удалено из-за оптимальности.

Выбор узла. Теперь список узлов содержит только S_{11}.

Реоптимизация. $S_{11} = S \cap \{x: x_1 \leq 2, x_2 \leq 0\}$. Полученная в результате линейная задача имеет оптимальное решение $\overline{x}^{11} = \left(\frac{3}{2}, 0\right)$ со значением 6. Так как $\underline{z} = 7, \overline{z}_{11} = 6$, то узел можно удалить **из-за ограничения.**

Выбор узла. Так как список узлов пустой, то алгоритм заканчивает свою работу. Текущее решение $x = (2,1)$ со значением $z = 7$ оптимальное.

Полное дерево метода ветвей и границ показано на рисунке 17.

10. Ускорение алгоритма

Основным показателем эффективности работы алгоритма является время решения задачи (2.1), (2.2) - (2.13). В среднем, чем больше размерность матрицы, полученной из ограничений (2.2) - (2.13), тем сложность решения будет больше, а значит, будет больше время, затраченное на поиск оптимального решения. Поэтому важно сократить число переменных.

Из анализа ограничений (2.2) – (2.13) легко сделать вывод, что основной вклад в сложность задачи вносят переменные f_{ik}, так как размер соответствующих матриц растет как $O(N^2)$. Размеры матриц других переменных, входящих в ограничения задачи, растут как $O(N)$. Поэтому важно сократить число переменных матрицы (f_{ik}).

Для начала заметим, что матрица (f_{ik}) является разреженной. Для простоты будем считать, что шлюз может принять только одно судно. Наши рассуждения можно легко распространить на больший размер шлюза (вмещающий большее число кораблей).

Каждая i-ая строчка матрицы (f_{ik}) может содержать только одну единицу, остальные элементы будут нулями. Задача заключается в определении нулевых областей матрицы (f_{ik}), это позволит сократить количество переменных этой матрицы.

Первый шаг ускорения заключается в правильном упорядочении кораблей, приходящих на шлюз. Упорядочим их по времени прибытия на шлюз. Это позволит увидеть распределение нулей и единиц в пределах матрицы (f_{ik}).

Второй шаг заключается в определении гарантированно нулевой области матрицы (f_{ik}).

На рисунке 18 показан вид матрицы (f_{ij}), которая получается после перенумерации кораблей способом, указанным выше. Здесь штриховкой показана область матрицы, где могут быть как нулевые, так и ненулевые

элементы. Незаштрихованная область матрицы – область, состоящая только из нулей.

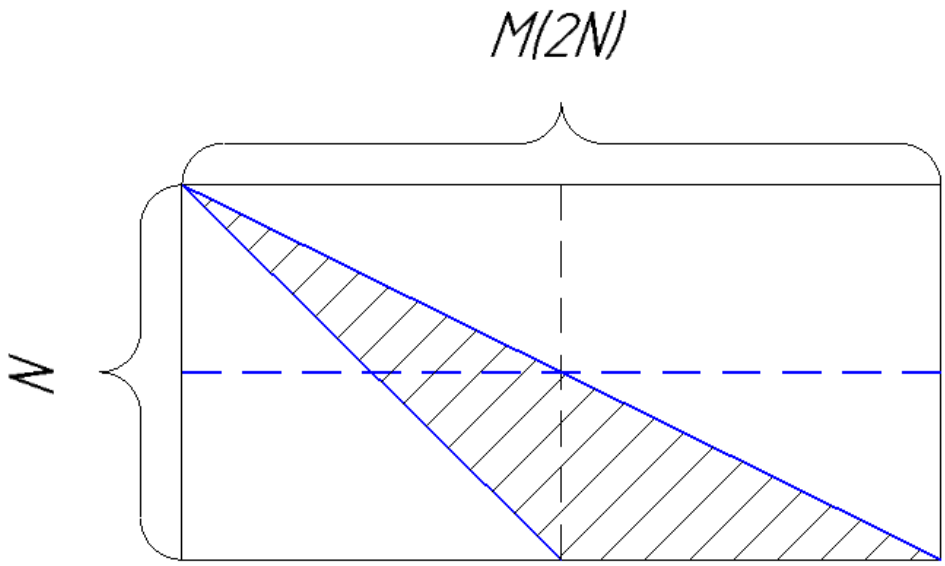

Рис 18. Вид матрицы f_{ik}

Покажем, что матрица f_{ik} приобретает в точности такой вид, как показано на рисунке 18. Для этого проведем три вспомогательные линии. Первые две пусть проходят через оси прямоугольника матрицы. Третья пусть будет осью квадрата, полученного, при разделении прямоугольника на две части (Рисунок 19).

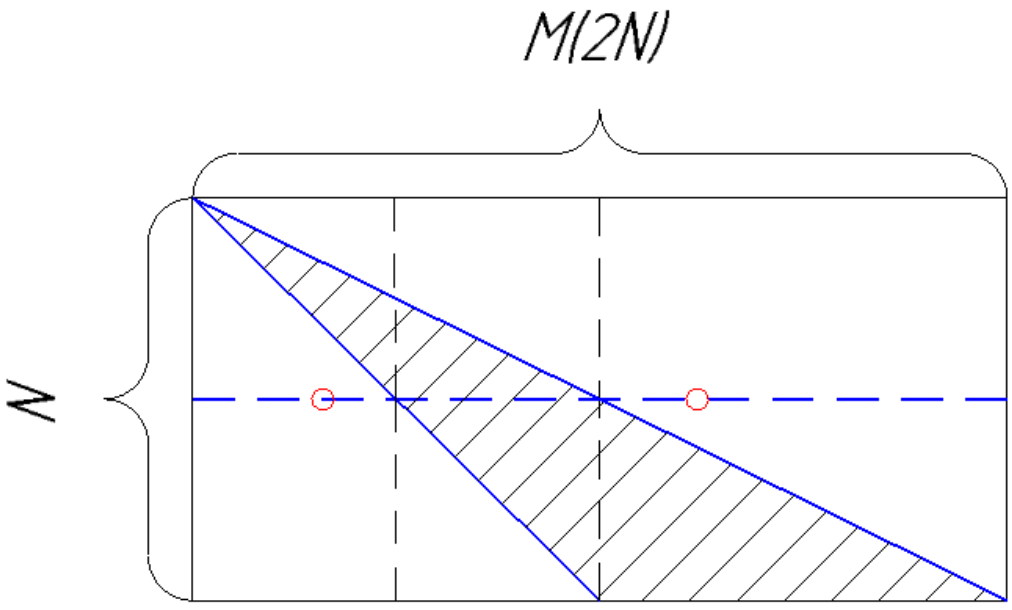

Рисунок 19. Определение нулевой области матрицы f_{ik}

Возьмем первую точку на правой части прямоугольника (Рисунок 19). В ней может быть только нулевой элемент, так как в противном случае $N/2$ кораблей шлюз переправил бы за $N + k$ $(k \geq 1)$ проводок, что невозможно. Потому что в худшем случае для переправы $N/2$ может быть N проводок. Аналогично можно показать это для любой другой точки из области, расположенной выше заштрихованной.

Возьмем вторую точку на левой части прямоугольника (Рисунок 19). В ней может быть только нулевой элемент, так как в противном случае $N/2$ кораблей можно было провести за $N/2 - k$ $(k \geq 1)$ проводок, что невозможно, так как в лучшем случае число кораблей равно число проводок, но никак не меньше.

Так как нам известна нулевая область матрицы (f_{ik}), то мы можем сократить количество переменных этой матрицы. Из рисунка 19 легко сделать вывод, что количество переменных матрицы (f_{ik}) сократилось в 4 раза.

11. Результаты расчетов

Написана программа с помощью среды разработки MATLAB, при ее написании были использованы постановка задачи (2.1), (2.2) - (2.13) и замечания из пункта 12 по ускорению алгоритма работы. Программа обладает графическим интерфейсом, с помощью которого наглядно можно показать работу алгоритма.

Заметим, что тесты алгоритма, показанные ниже, проводились на стационарном компьютере с такими характеристиками: процессор - Intel® Core™ i5-4210U CPU @ 1,70GHz 2,40 GHz, память 12 Gb DDR3, видеокарта – NVIDIA GeForce 840M. На системах с лучшими характеристиками данные могут отличаться от представленных.

В пункте 10 был предложен способ ускорения алгоритма. Проанализируем эффективность улучшенного алгоритма. Для этого построим график зависимости $T(N)$ (времени выполнения алгоритма от числа кораблей, прибывающих на шлюз).

Расчеты значений проводились так. На вход подавались случайные выборки времен прибытия кораблей. Для каждого определенного количества кораблей, прибывающих на вход, формировалось 20 случайных выборок. Алгоритм MILP находил оптимальный вектор проводок для каждой выборки прибытия кораблей и фиксировалось время работы этого алгоритма. Значения времен работы усреднялись (арифметическое среднее). Результаты расчетов можно увидеть на рисунке 20.

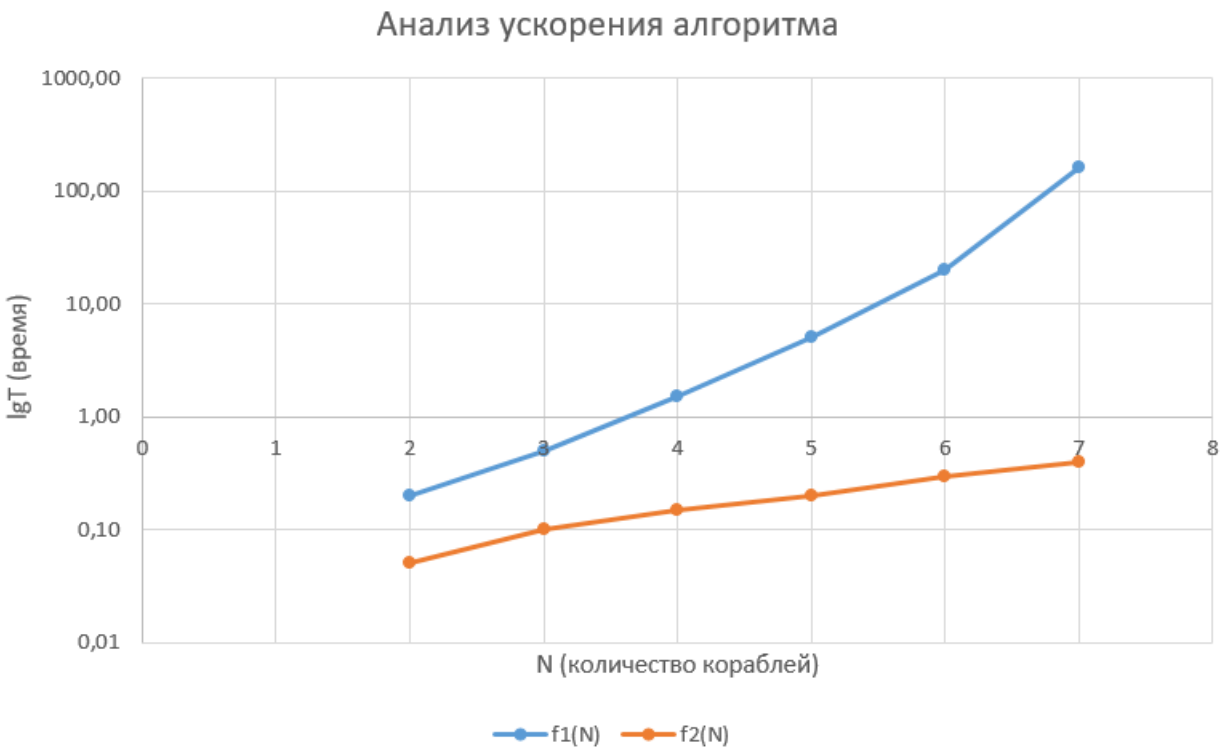

Рисунок 20. Сравнение работы алгоритмов с ускорением и без (ось абсцисс – количество кораблей, ось ординат – это время работы алгоритма в секундах – логарифмическая шкала). Здесь $f1(N)$ – ускоренный алгоритм, а $f2(N)$ – без ускорения.

Зависимость времени работы алгоритма от числа кораблей показана на рисунке 21. Из него можно сделать вывод, что алгоритм пригоден для расчета проводок, когда количество кораблей, прибывающих на шлюз, в пределах 15-и.

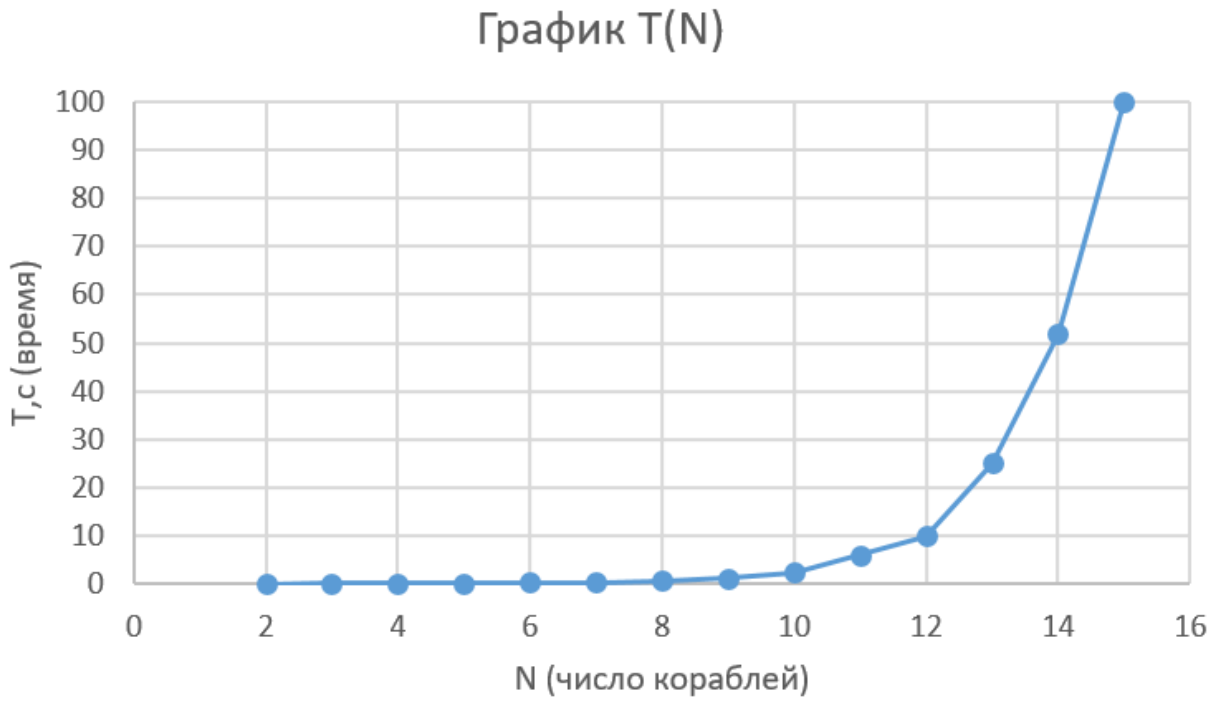

Рисунок 21. Зависимость времени работы алгоритма от числа кораблей (ось абсцисс – число кораблей, ось ординат – время в секундах)

В результате анализа работы алгоритма установлена следующая закономерность: алгоритм рассчитывает проводки долго (больше среднего времени), когда разница между какими-либо двумя кораблями, прибывающими на шлюз меньше времени осуществления проводок. В противном случае время выполнения алгоритма в разы быстрее.

12. Выводы

В результате выполнения работы реализован алгоритм для оптимального шлюзования судов.

В процессе написания программы появились трудности в реализации алгоритма из работы [2]. А именно, реализованный алгоритм некорректно находил оптимальное решение, в связи с чем был сделан вывод, что либо не все ограничения были представлены в работе [2], либо в ограничениях автором были допущены ошибки. Поэтому это побудило переработать постановку задачи, для этого была удалена часть ограничений из исходной задачи и добавлена часть новых ограничений.

После реализации нового алгоритма было установлено, что в среднем при произвольных входных данных алгоритм считает распределение проводок медленно, а именно – уже для 7 кораблей время решения алгоритма занимало около 200 секунд. Поэтому возникла задача ускорения этого алгоритма.

Был найден способ ускорения алгоритма. Этот способ указан в пункте 10 данной работы. Ускорение дает возможность проводить расчет до 15-и кораблей, прибывающих на шлюз (при произвольных входных значениях).

Также после анализа входных данных был найдет эмпирический критерий, позволяющий алгоритму работать еще быстрее. Было установлено, что алгоритм считает проводки медленно (в среднем), если в векторе прибывающих кораблей есть последовательные пары, разница времен прибытия которых меньше, чем время проводки кораблей. Если времена прибытия кораблей немного сместить, то алгоритм будет считать проводки в несколько раз быстрее (2-3 раза).

13. Литература

1. Andersen, E. D., and Andersen, K. D. Presolving in linear programming. Mathematical Programming 71, pp. 221–245, 1995.

2. Verstichel J., De Causmaecker, P. (sup.), Vanden Berghe, G. (sup.) (2013). The Lock Scheduling Problem (Het sluisplanningsprobleem), 160 pp.

3. R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30:1–52, 2008.

4. A. Schrijver, Theory of Linear and Integer Programming, Wiley, New York (1986).

5. G.M. Ziegler, Lectures on Polytopes, Springer, New York (1995)

6. E. Balas, Disjunctive programming: properties of the convex hull of feasible points, GSIA Management Science Research Report MSRR 348, Carnegie Mellon University (1874), published as invited paper in *Discrete Applied Mathematics 89* (1998) 1-44

7. E. Balas, Disjunctive programming and a hierarchy of relaxations for discrete optimization problems, SIAM Journal on Algebraic and Discrete Methods 6 (1985) 466–486

8. H. Sherali and W. Adams, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, *SIAM Journal on Discrete Mathematics 3* (1990) 311–430.

9. L. Lov´ asz and A. Schrijver, Cones of matrices and set-functions and 0-1 optimization, *SIAM Journal of Optimization 1* (1991) 166–190.

10. E. Balas, S. Ceria and G. Cornu´ ejols, A lift-and-project cutting plane algorithm for mixed 0-1 programs, *Mathematical Programming 58* (1993) 295–324.

11. E. Balas and M. Perregaard, A Precise correspondence between lift-and-project cuts, simple disjunctive cuts and mixed integer Gomory cuts for 0-1 programming, *Mathematical Programming B 94* (2003) 221–245.

12.https://www.mathworks.com/help/optim/ug/mixed-integer-linear-programming-algorithms.html