

с/к “Эффективные алгоритмы”

Лекция 7: Линейное программирование

А. Куликов

Computer Science клуб при ПОМИ
<http://logic.pdmi.ras.ru/~infclub/>

План лекции

- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

План лекции

- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

Что такое линейное программирование?

Что такое линейное программирование?

- Многие задачи, для которых нам нужны алгоритмы, являются **оптимизационными**: **кратчайший** путь, **минимальное** остовное дерево, **длиннейшая** возрастающая подпоследовательность.
- В таких задачах мы ищем решение, которое
 - ▶ удовлетворяет некоторым ограничениям (к примеру, путь должен идти по ребрам графа из вершины s в t , дерево должно содержать все вершины графа, подпоследовательность должна быть возрастающей),
 - ▶ является оптимальным (относительно некоторой заданной функции) среди всех решений, удовлетворяющих данным ограничениям.
- **Линейное программирование** (linear programming) описывает широкий класс оптимизационных задач, в которых и ограничения, и целевая функция являются **линейными функциями**.

Что такое линейное программирование?

Что такое линейное программирование?

Итак, в задаче линейного программирования нам дано множество переменных и мы хотим найти набор вещественных значений для них, так чтобы выполнить множество линейных равенств и/или неравенств от данных переменных и максимизировать или минимизировать данную целевую функцию.

Пример: максимизация прибыли

Максимизация прибыли

- Фирма выпускает два типа товаров — C_1 и C_2 .
- Какое количество товаров каждого из типов ей нужно производить, чтобы максимизировать прибыль?
- Допустим, в день она производит x_1 единиц товара типа C_1 по цене 1 руб. каждый и x_2 единиц товара типа C_2 по цене 6 руб. каждый.
- Пусть также известно, что потребность в товарах типов C_1 и C_2 в день ограничена сверху 200 и 300 единицами, соответственно.
- Более того, фирма не может произвести за день более 400 единиц товара.

Соответствующая задача линейного программирования

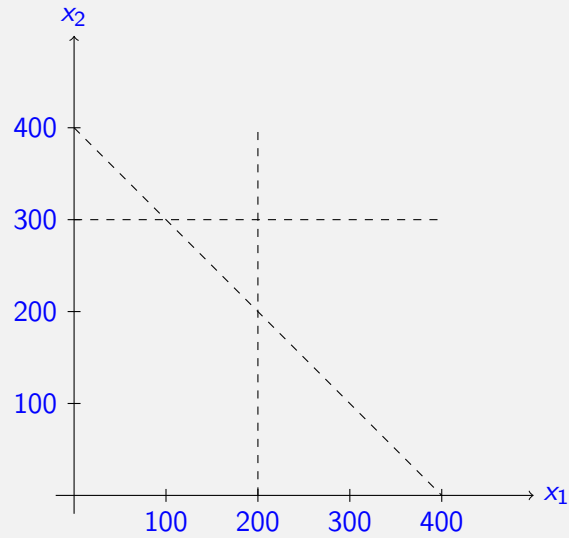
Соответствующая задача линейного программирования

целевая функция	\max	$x_1 + 6x_2$
ограничения	x_1	≤ 200
	x_2	≤ 300
	$x_1 + x_2$	≤ 400
	x_1, x_2	≥ 0

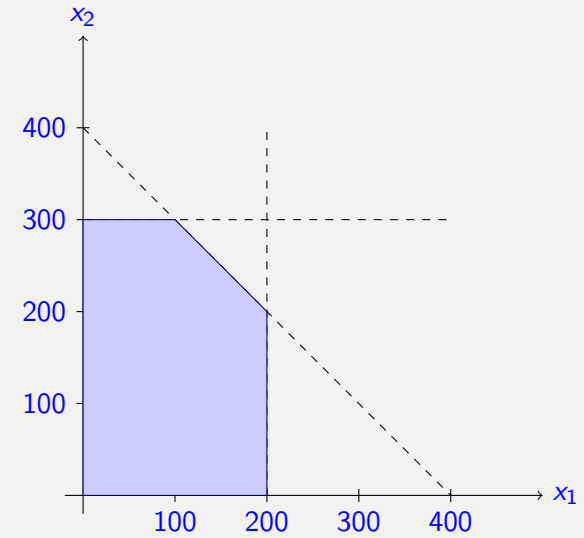
Множество допустимых решений

- Линейное уравнение от переменных x_1 и x_2 задает прямую на плоскости, а линейное неравенство — полуплоскость.
- Значит, множество всех **допустимых решений** (feasible solutions), то есть точек (x_1, x_2) , удовлетворяющих всем ограничениям, является пересечением пяти полуплоскостей.
- Это пересечение является выпуклым многоугольником.

Многоугольник решений



Многоугольник решений

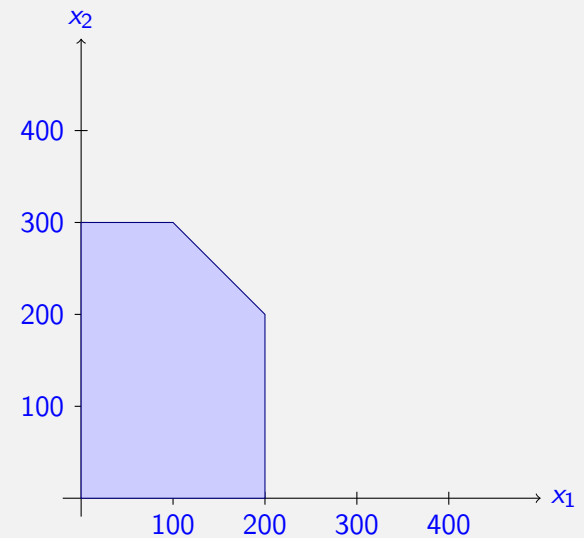


Целевая функция

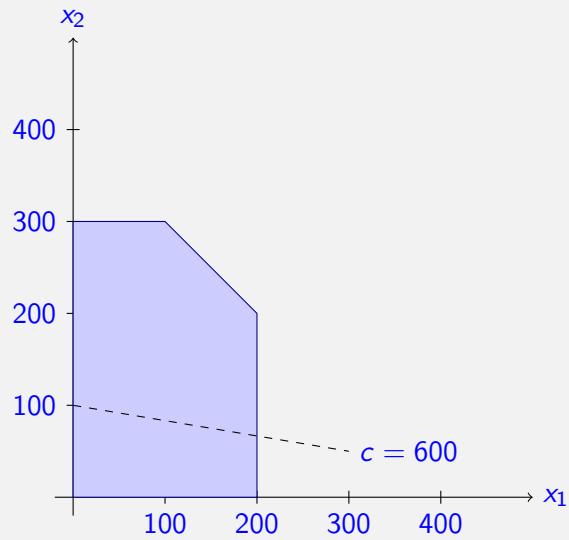
Целевая функция

- Вспомним, что нам нужно максимизировать прибыль, то есть функцию $x_1 + 6x_2$.
- Точки (x_1, x_2) , соответствующие прибыли c , лежат на прямой $x_1 + 6x_2 = c$.
- Чем больше c , тем выше эта прямая.
- Таким образом, для максимизации целевой функции нужно поднимать эту прямую до тех пор, пока она пересекает многоугольник допустимых решений.

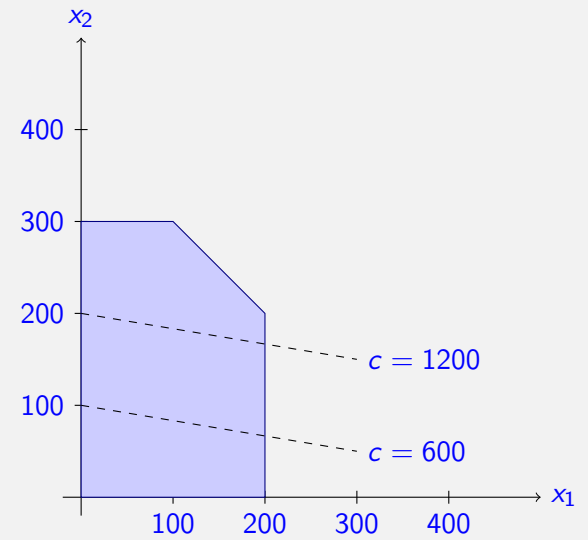
Оптимальное значение



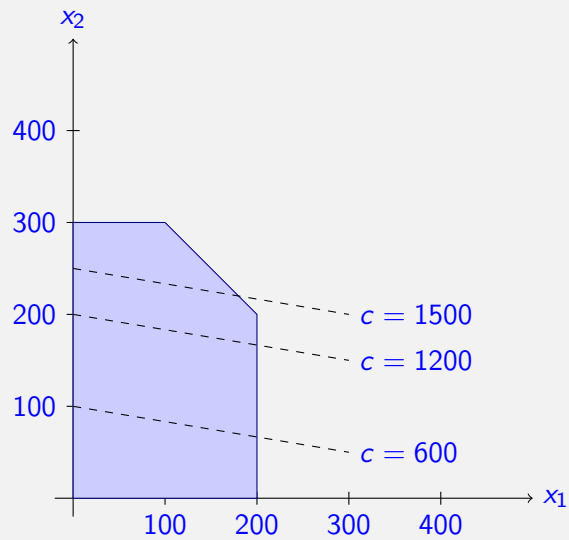
Оптимальное значение



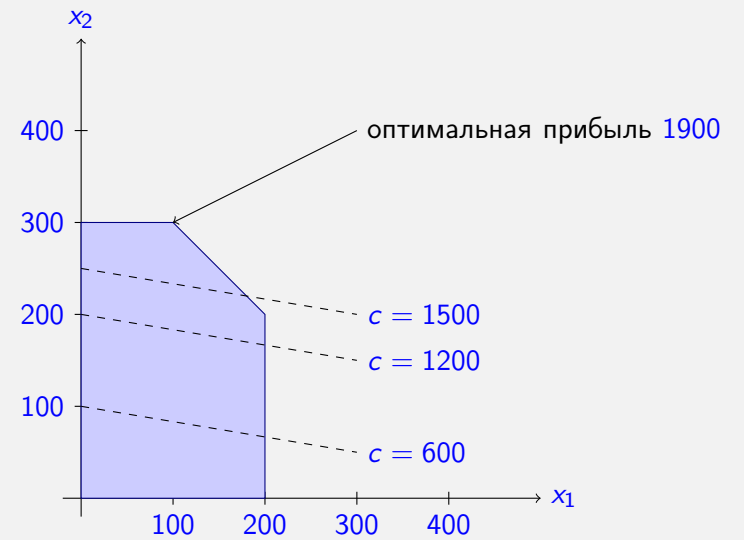
Оптимальное значение



Оптимальное значение



Оптимальное значение



Недопустимые и неограниченные программы

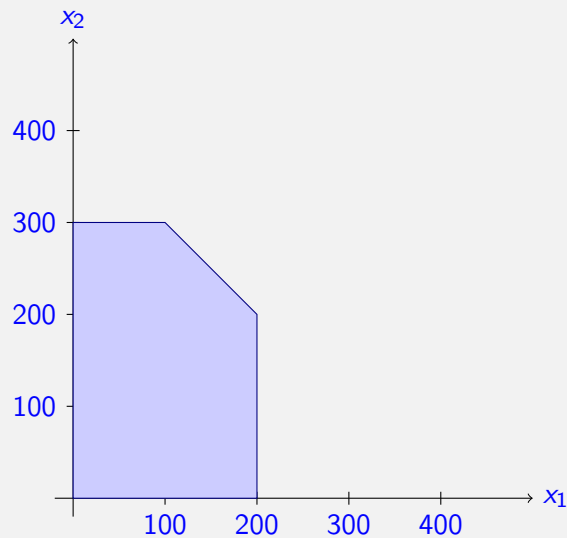
- В нашем примере оптимальное значение достигается в вершине многоугольника. Ясно, что в другой ситуации множество оптимальных точек могло бы совпасть с ребром многоугольника.
- Бывают и ситуации, когда решения нет вообще.
 - ▶ Линейная программа **недопустима** (infeasible), то есть ограничения противоречат друг другу. К примеру,
$$x \leq 1, x \geq 2.$$
 - ▶ Ограничения настолько слабы, что множество допустимых решений **неограничено** (unbounded). К примеру,
$$\max x_1 + x_2$$
$$x_1, x_2 \geq 0$$

Решение задач линейного программирования

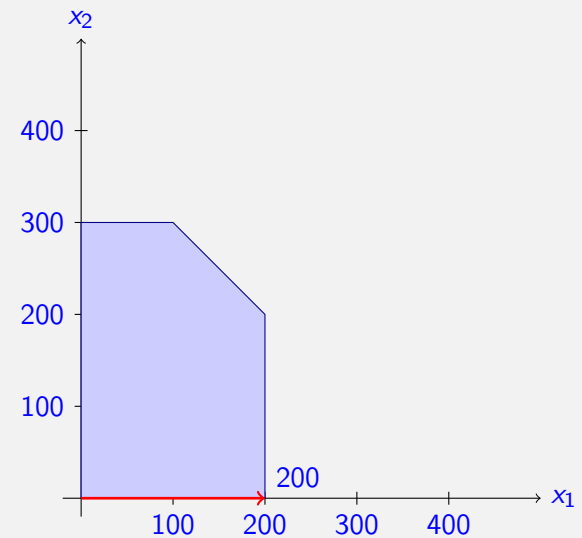
Основные идеи симплекс-метода

- Алгоритм выбирает какую-нибудь вершину многоугольника и последовательно ищет смежные вершины, в которых достигается лучшее значение целевой функции.
- Найдя вершину, у которой нет более оптимального соседа, алгоритм останавливается и выдает значение целевой функции в этой вершине.
- Почему же этот **локальный** оптимум является еще и **глобальным**?
- Из простых геометрических соображений: поскольку все соседи оптимальной вершины многоугольника лежат ниже прямой прибыли, то и весь прямоугольник лежит ниже этой прямой.

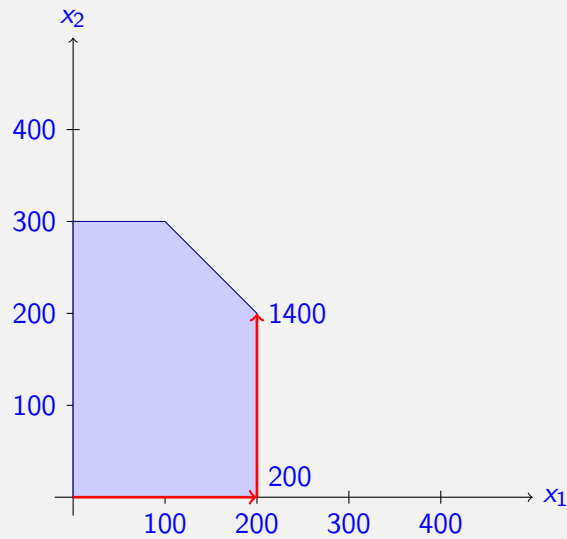
Оптимальное значение



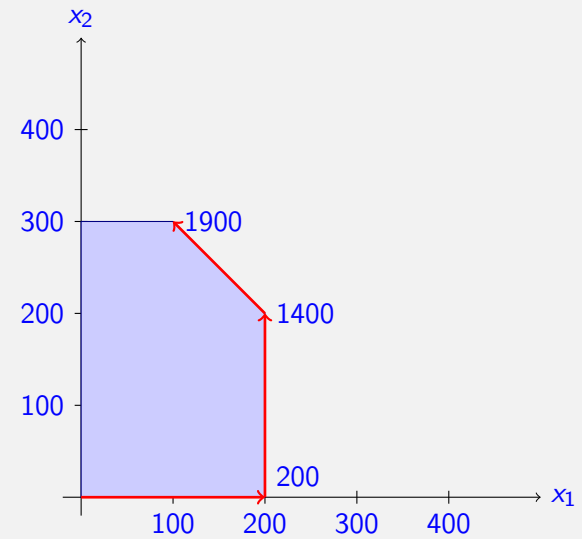
Оптимальное значение



Оптимальное значение



Оптимальное значение



Усложняем пример

- Допустим теперь, что фирма стала производить товары типа C_3 по цене 13 руб. за единицу.
- Как и прежде, фирма не может произвести более 400 единиц товара за день.
- Более того, для товаров типов C_2 и C_3 требуется некоторая процедура упаковки, налагающая ограничение $x_2 + 3x_3 \leq 600$.

Соответствующая задача линейного программирования

Соответствующая задача линейного программирования

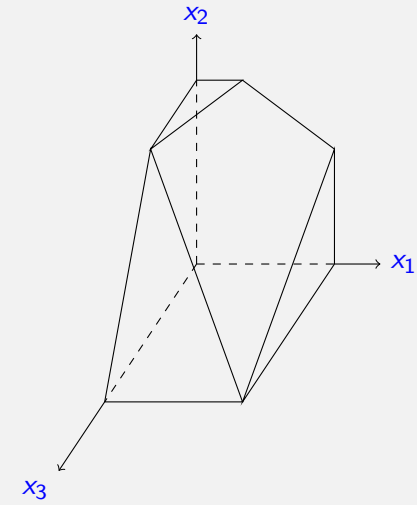
$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Множество допустимых решений

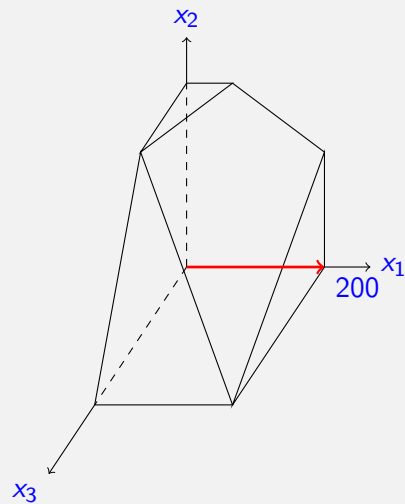
Множество допустимых решений

- Теперь пространство решений является трехмерным.
- Каждое линейное уравнение задает плоскость, а линейное неравенство — полупространство.
- Множество допустимых решений теперь является многогранником, который, в свою очередь, является пересечением семи полупространств.
- Точки прибыли c лежат на плоскости $x_1 + 6x_2 + 13x_3 = c$.
- Симплекс-метод будет путешествовать по вершинам этого многогранника, пока не найдет локальный оптимум.

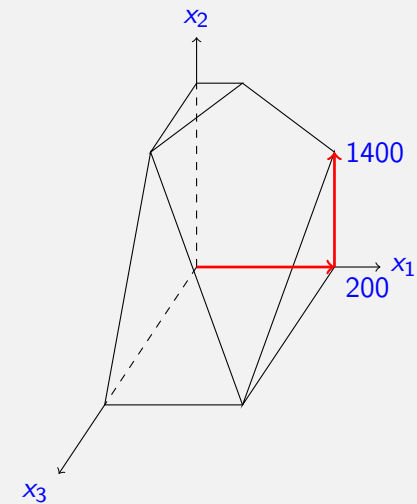
Многогранник допустимых решений



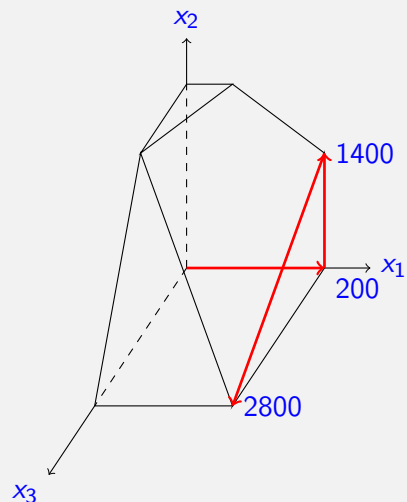
Многогранник допустимых решений



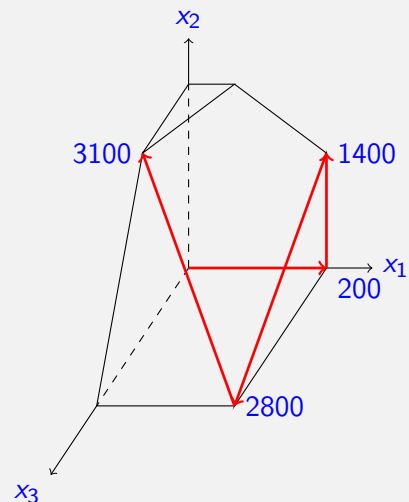
Многогранник допустимых решений



Многогранник допустимых решений



Многогранник допустимых решений



Двойственность

Как еще можно убедиться в том, что найденное решение оптимально?

$$\begin{aligned} \max x_1 + 6x_2 + 13x_3 \\ x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 + x_3 &\leq 400 \\ x_2 + 3x_3 &\leq 600 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

$$\begin{aligned} x_2 &\leq 300 \\ x_1 + x_2 + x_3 &\leq 400 \\ 4x_2 + 12x_3 &\leq 2400 \\ \hline x_1 + 6x_2 + 13x_3 &\leq 3100 \end{aligned}$$

Двойственность

- Таким образом, **3100** — действительно оптимальное значение.
- Вопрос только в том, как именно мы догадались, какие неравенства и с какими коэффициентами нужно сложить.
- Чуть позже мы увидим, что такие коэффициенты можно найти, решив другую задачу линейного программирования.
- А на самом деле, нам и решать ее не нужно будет, поскольку, решая исходную задачу линейного программирования, мы решаем и эту задачу.
- Именно поэтому она и называется **двойственной**.

Целочисленные решения

Целочисленные решения

- Итак, симплекс-метод выдает решение, но всегда ли оно нам подходит?
- Нет, не всегда, поскольку вершина, в которой достигается оптимум, может иметь **дробные** координаты.
- Конечно, можно полученные координаты округлить. В большинстве практических задач это изменит значение целевой функции несильно, но есть и задачи, в которых так просто решить проблему не удастся.
- На самом же деле, нахождение оптимального целочисленного решения задачи линейного программирования — NP-трудная задача.

Вариации

Вариации

Общая задача линейного программирования имеет несколько степеней свободы:

- задача может быть как минимизационной, так и максимизационной;
- ограничения могут быть как равенствами, так и неравенствами;
- переменные могут быть как положительными, так и любыми.

Мы покажем, что все эти вариации легко сводятся друг к другу.

Сведения

Сведения

- Чтобы изменить минимизационную задачу на максимизационную, достаточно домножить коэффициенты целевой функции на -1 .
- Неравенство $\sum_{i=1}^n a_i x_i \leq b$ можно заменить на следующую систему:

$$\begin{aligned}\sum_{i=1}^n a_i x_i + s &= b \\ s &\geq 0\end{aligned}$$

- Равенство $ax = b$ можно заменить на пару неравенств $ax \leq b$ и $ax \geq b$.
- Переменную x , для которой нет ограничения на знак, можно заменить на две положительные переменные $x^+, x^- \geq 0$, заменив каждое вхождение x на $x^+ - x^-$.

Нормальная форма

Нормальная форма задачи линейного программирования

Итак, с помощью данных сведений любую задачу линейного программирования можно привести к **нормальной форме**, в которой все переменные неотрицательны, все ограничения записаны равенствами и требуется минимизировать целевую функцию.

Пример приведения к нормальной форме

$$\begin{aligned} \max x_1 + 6x_2 \\ x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$

$$\begin{aligned} \min -x_1 - 6x_2 \\ x_1 + s_1 &= 200 \\ x_2 + s_2 &= 300 \\ x_1 + x_2 + s_3 &= 400 \\ x_1, x_2, s_1, s_2, s_3 &\geq 0 \end{aligned}$$

Матричная форма

Матричная форма

$$\begin{aligned} \max c^T x \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

$$\begin{aligned} \max x_1 + 6x_2 \\ x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 200 \\ 300 \\ 400 \end{pmatrix}$$

Почему целочисленное линейное программирование трудно?

Почему целочисленное линейное программирование трудно?

- Покажем, что задача 3-выполнимости может быть сведена к задаче целочисленного линейного программирования.
- Для каждой переменной x входной формулы запишем два неравенства: $x \geq 0$, $x \leq 1$.
- Для каждого клона $(x \vee \neg y \vee z)$ запишем такое неравенство: $x + (1 - y) + z \geq 1$.
- Если у полученной системы неравенств есть хотя бы одно допустимое решение, то исходная формула выполнима, и наоборот.

План лекции

- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

Формулировка задачи

Определение

- **Сеть** (flow network) называется ориентированный граф, каждому ребру (u, v) которого сопоставлено неотрицательное число $c(u, v)$, называемое **пропускной способностью** (capacity), с двумя выделенными вершинами, называемыми **исток** s (source) и **сток** t (sink).
- **Потоком** (flow) в сети $G = (V, E)$ называется функция $f : E \rightarrow \mathbb{R}$, удовлетворяющая следующим свойствам:
 - пропускные способности не нарушены:
 $\forall (u, v) \in E, f(u, v) \leq c(u, v)$;
 - сохранение потока:
 $\forall u \in V \setminus \{s, t\}, \sum_{(w, u) \in E} f(w, u) = \sum_{(u, z) \in E} f(u, z)$.

Формулировка задачи (продолжение)

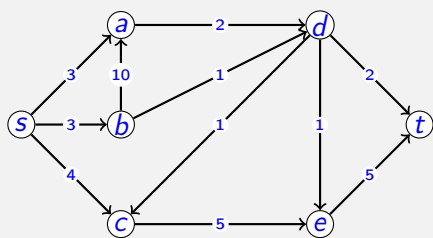
Определение

- **Величиной** (value) $|f|$ потока f называется сумма $\sum_{(s, u) \in E} f(s, u)$.
- **Задача о максимальном потоке** (maximum-flow problem) заключается в нахождении максимального потока в заданной сети.

Линейное программирование

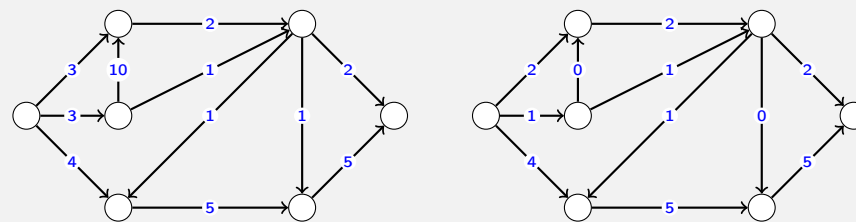
Таким образом, задача заключается в присвоении каждому ребру графа неотрицательного числа так, чтобы выполнялись некоторые линейные ограничения и максимизировалась некоторая линейная функция, а это и есть задача линейного программирования.

Пример сети



Максимизировать $f(s, a) + f(s, b) + f(s, c)$, выполнив 27 ограничений: 11 ограничений на неотрицательность (например, $f(s, a) \geq 0$), 11 ограничений на пропускную способность ($f(s, a) \leq 3$) и 5 на сохранение потока ($f(s, c) + f(d, c) = f(c, e)$).

Пример потока



сеть

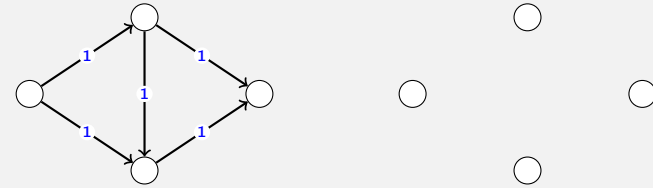
поток

Как решать задачу о максимальном потоке?

Как решать задачу о максимальном потоке?

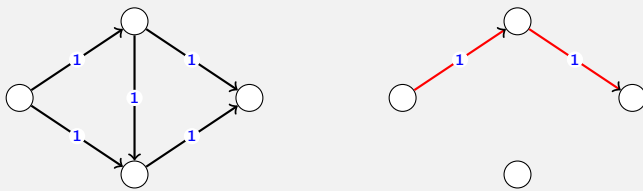
- О симплекс-методе мы пока знаем лишь то, что он движется по вершинам многогранника допустимых решений, улучшая значение целевой функции.
- Соответствующий алгоритм для задачи о максимальном потоке:
 - ▶ начать с пустого потока
 - ▶ повторять: взять подходящий путь из s в t и увеличить поток по ребрам этого пути настолько, насколько это возможно.

Простой пример



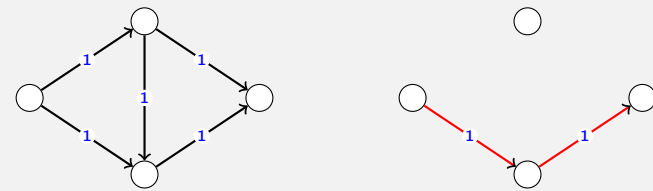
рассмотрим простую сеть

Простой пример



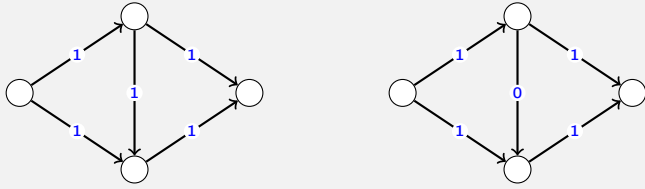
первый выбранный алгоритмом путь

Простой пример



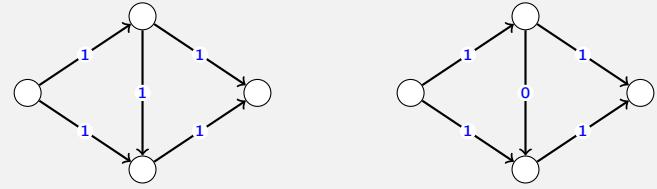
второй выбранный алгоритмом путь

Простой пример



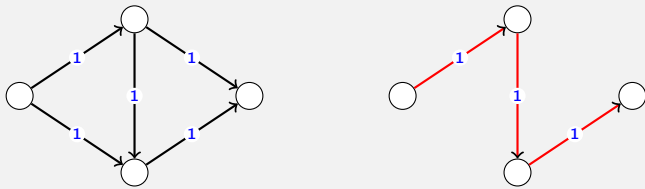
построенный поток

Простой пример



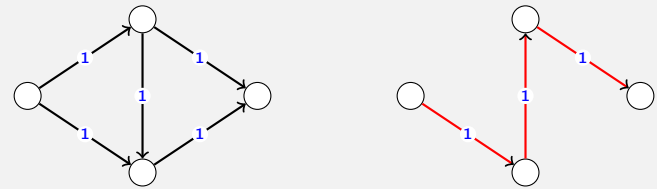
очевидно, найденное решение оптимально

Простой пример



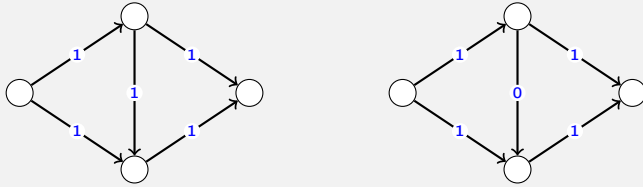
но что делать, если первым выбран этот путь?

Простой пример



тогда на следующем шаге алгоритм выберет этот путь

Простой пример



и получит все тот же оптимальный поток

Таким образом

Таким образом

- На каждом шаге алгоритм выбирает $s - t$ путь, ребра (u, v) которого могут быть двух типов:
 - ▶ (u, v) есть в исходной сети, но поток по нему меньше его пропускной способности (в этом случае по ребру можно увеличить поток на $\leq c(u, v) - f(u, v)$);
 - ▶ обратное ребро (v, u) есть в исходной сети и по нему есть положительный поток (тогда можно увеличить поток по ребру (u, v) на $\leq f(v, u)$).
- Для поиска таких путей используется остаточная сеть.

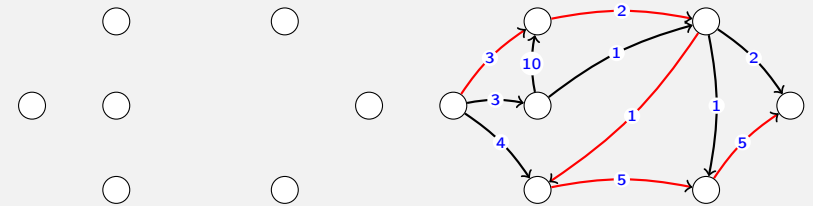
Остаточная сеть

Определение

Остаточной сетью (residual network) сети $G = (V, E)$, порожденной потоком f , называется сеть $G^f = (V, E^f)$ со следующими пропускными способностями:

$$c^f(u, v) = \begin{cases} c(u, v) - f(u, v), & \text{если } (u, v) \in E \text{ и } f(u, v) < c(u, v) \\ f(v, u), & \text{если } (v, u) \in E \text{ и } f(v, u) > 0 \end{cases}$$

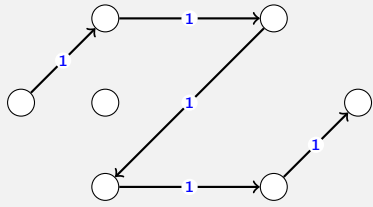
Пример работы алгоритма



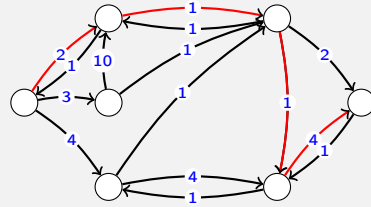
ПОТОК

ОСТАТОЧНАЯ СЕТЬ

Пример работы алгоритма

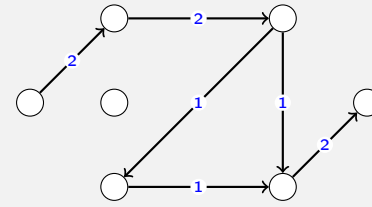


ПОТОК

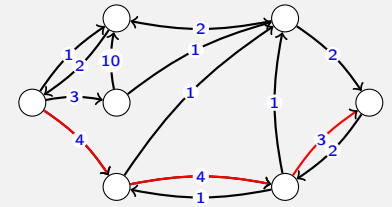


остаточная сеть

Пример работы алгоритма

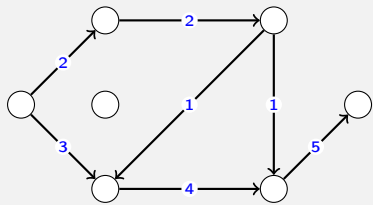


ПОТОК

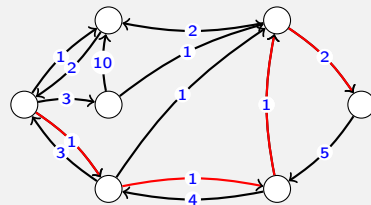


остаточная сеть

Пример работы алгоритма

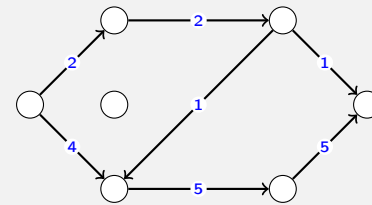


ПОТОК

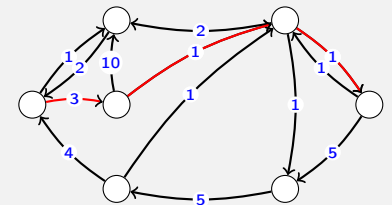


остаточная сеть

Пример работы алгоритма

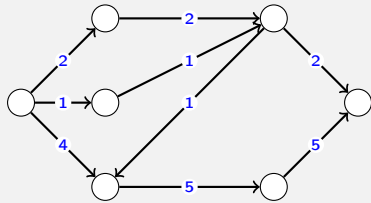


ПОТОК

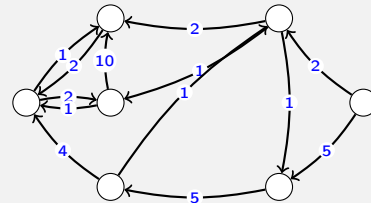


остаточная сеть

Пример работы алгоритма



ПОТОК



остаточная сеть

Минимальный разрез

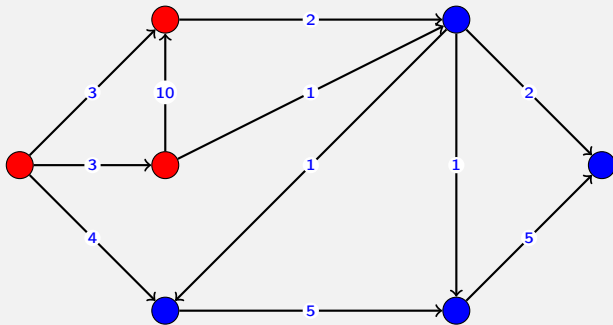
Определение

- (s, t) -разрезом $((s, t)$ -cut) сети G называется разбиение множества вершин V на две части L и R , таких что $s \in L$ и $t \in R$.
- Пропускной способностью разреза называется суммарная пропускная способность всех ребер из L в R .

Сертификат оптимальности

- Легко видеть, что величина любого потока не превосходит пропускной способности любого (s, t) -разреза.
- Более того, величина максимального потока равна пропускной способности минимального (s, t) -разреза.
- Другими словами, минимальный разрез является сертификатом максимальности потока.

Пример минимального разреза



Доказательство

Лемма

Величина максимального потока равна пропускной способности минимального (s, t) -разреза.

Доказательство

- Пусть f — поток, найденный алгоритмом.
- Мы знаем, что в остаточной сети G^f нет пути из s в t .
- Рассмотрим такой (s, t) -разрез: L — множество вершин, достижимых из s , R — множество оставшихся вершин.
- Ясно, что $f(u, v) = c(u, v)$, где $u \in L, v \in R$.
- Также $f(v, u) = 0$, где $u \in L, v \in R$.
- Таким образом, поток через построенный разрез (равный $|f|$) совпадает с пропускной способностью этого разреза.

□

Анализ времени работы

Анализ времени работы

- Допустим, все пропускные способности ребер графа являются целыми числами и ограничены сверху числом C .
- Каждая итерация выполняется за время $O(|E|)$.
- Ясно, что на каждой итерации величина потока является целым числом и каждый раз увеличивается хотя бы на 1.
- Значит, всего понадобится не более $C|E|$ итераций.
- Если же каждый раз выбирать кратчайший путь в остаточной сети, то время работы алгоритма будет $O(|V| \cdot |E|^2)$.

План лекции

- 1 Введение
- 2 Поток в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

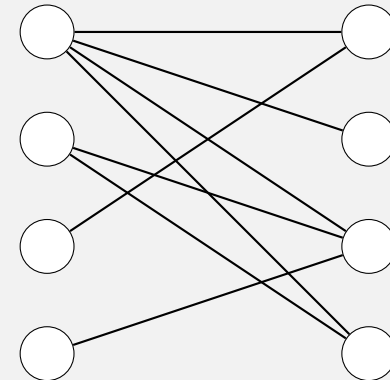
Постановка задачи

Определение

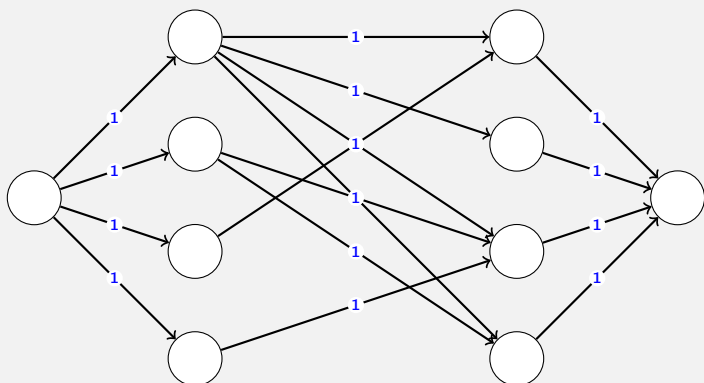
- **Задача о максимальном паросочетании в двудольном графе** (maximum bipartite matching problem) заключается в нахождении в данном двудольном графе G такого максимального подмножества ребер, никакие два из которых не имеют общего конца.
- **Задача о совершенном паросочетании в двудольном графе** (perfect bipartite matching problem) заключается в проверке существования в данном двудольном графе с равными размерами долей паросочетания, покрывающего все вершины.

Мы покажем, что задача о максимальном паросочетании в двудольном графе может быть сведена к задаче о максимальном потоке, а следовательно, и к задаче линейного программирования.

Идея сведения



Идея сведения



Однако

Однако

- Итак, достаточно найти максимальный поток.
- Ясно, что найденный поток будет давать максимальное паросочетание: если бы существовало лучшее паросочетание, мы бы могли по нему построить лучший поток.
- Но найденный поток мог оказаться не целочисленным.
- К счастью, задача о максимальном потоке обладает хорошим свойством: если все пропускные способности целые, то и найденный алгоритмом поток будет целочисленным.
- Но это скорее исключение, чем правило. В общем случае найти оптимальное целочисленное решение трудно.
- Трудно, на самом деле, даже проверить, есть ли хоть одно допустимое целочисленное решение.

План лекции

- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность**
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

Неформально

- Как мы увидели, в сетях величина любого потока меньше пропускной способности любого разреза.
- Более того, существует разрез, пропускная способность которого совпадает с величиной максимального потока.
- Таким образом, минимальный разрез является сертификатом оптимальности потока.
- Оказывается, для любой задачи линейного программирования существует двойственная задача с теми же свойствами.

Сертификат оптимальности

$$\begin{aligned} \max x_1 + 6x_2 \\ x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$

- Оптимальное значение 1900 достигается в точке $(x_1, x_2) = (100, 300)$.
- Как доказать его оптимальность?
- Сложив первое неравенство со вторым, умноженным на 6, получим: $x_1 + 6x_2 \leq 2000$.
- Сложив второе, умноженное на 5, с третьим, получим: $x_1 + 6x_2 \leq 1900$.
- Такой сертификат есть для каждой задачи линейного программирования.

Как найти такой сертификат?

Как найти такой сертификат?

- $x_1 \leq 200, x_2 \leq 300, x_1 + x_2 \leq 400$
- Обозначим через y_1, y_2, y_3 коэффициенты, на которые нужно домножить данные неравенства, чтобы получить сертификат.
- Умножив и сложив:
 $(y_1 + y_3)x_1 + (y_2 + y_3)x_2 \leq 200y_1 + 300y_2 + 400y_3$.
- Мы хотим, чтобы левая часть этого неравенства выглядела как наша целевая функция, то есть чтобы $y_1 + y_3$ и $y_2 + y_3$ были равны 1 и 6, соответственно.
- Впрочем, если $y_1 + y_3 \geq 1$ и $y_2 + y_3 \geq 6$, то приведенное выше неравенство тоже даст некоторую оценку.
- Например, $(y_1, y_2, y_3) = (5, 3, 6)$ удовлетворяет всем ограничениям, но дает лишь верхнюю оценку 4300 на значение исходной целевой функции.

Как найти такой сертификат?

Как найти такой сертификат?

- Мы же хотим найти как можно лучшую оценку, то есть найти минимум функции $200y_1 + 300y_2 + 400y_3$ для (y_1, y_2, y_3) , удовлетворяющих ограничениям выше.
- А это и есть задача линейного программирования:

$$\begin{aligned} \min 200y_1 + 300y_2 + 400y_3 \\ y_1 + y_3 &\geq 1 \\ y_2 + y_3 &\geq 6 \\ y_1, y_2, y_3 &\geq 0 \end{aligned}$$

- Ясно, что любое допустимое значение целевой функции является верхней оценкой на значение целевой функции исходной задачи.
- Значит, если найти равные значения целевых функций обеих задач, то это значение будет оптимально для них обеих.
- Например, $(x_1, x_2) = (100, 300), (y_1, y_2, y_3) = (0, 5, 1)$.

Двойственная задача

Исходная задача

$$\begin{aligned} \max c^T x \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

Двойственная задача

$$\begin{aligned} \min y^T b \\ y^T A &\geq c^T \\ y &\geq 0 \end{aligned}$$

Лемма

Если у исходной задачи есть ограниченное оптимальное решение, то оно есть и у двойственной и значения целевых функций совпадают.

План лекции

- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

План лекции

- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

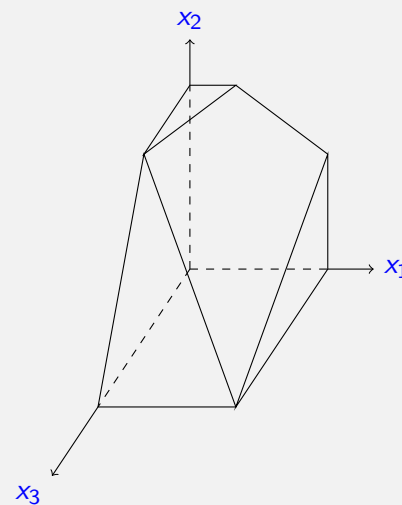
Общая схема

Общая схема

- выбрать произвольную вершину v многогранника допустимых решений
- пока у v есть соседняя вершина v' , в которой достигается лучшее значение целевой функции, заменять v на v'

Осталось научиться находить вершины и их соседей.

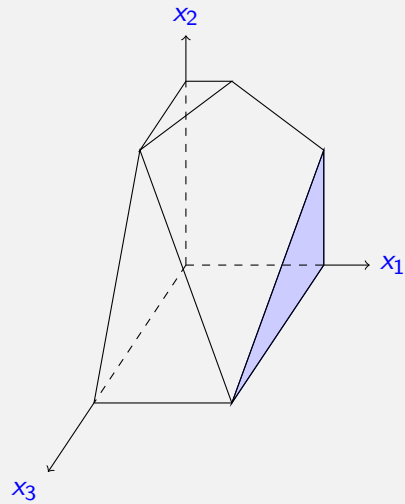
Многогранник допустимых решений



Задача

$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{aligned}$$

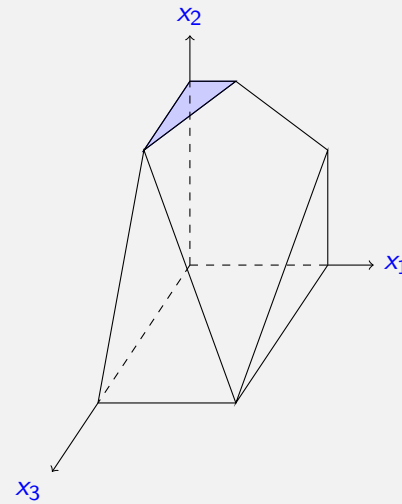
Многогранник допустимых решений



Задача

$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{aligned}$$

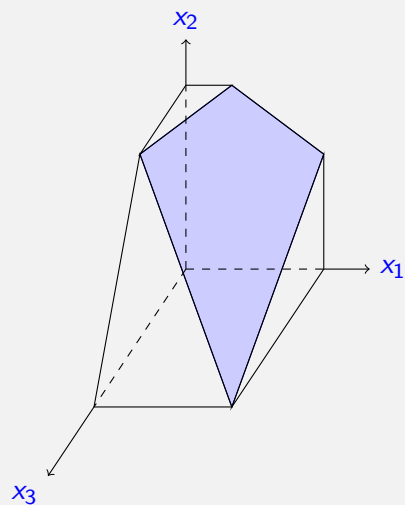
Многогранник допустимых решений



Задача

$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{aligned}$$

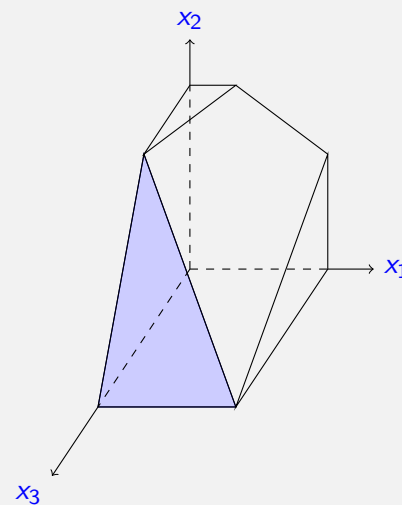
Многогранник допустимых решений



Задача

$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{aligned}$$

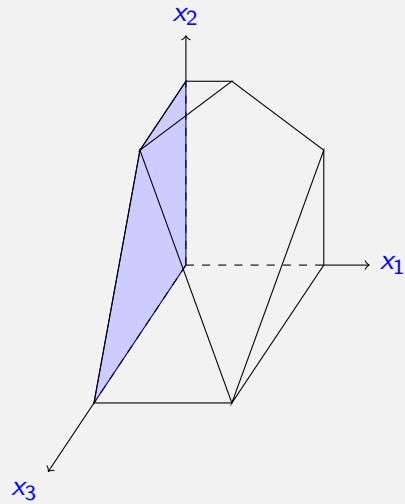
Многогранник допустимых решений



Задача

$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{aligned}$$

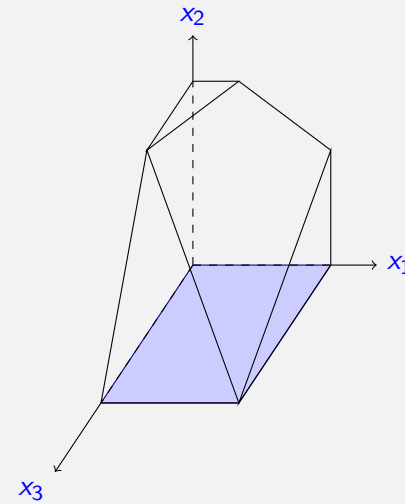
Многогранник допустимых решений



Задача

$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{aligned}$$

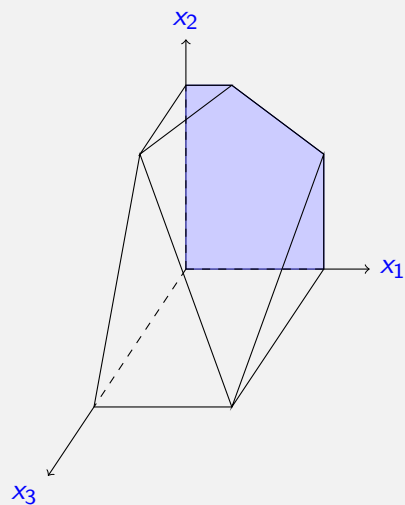
Многогранник допустимых решений



Задача

$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{aligned}$$

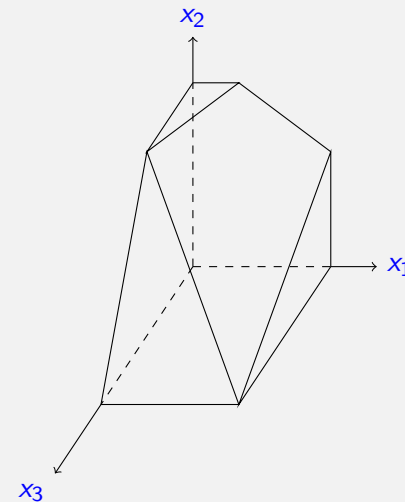
Многогранник допустимых решений



Задача

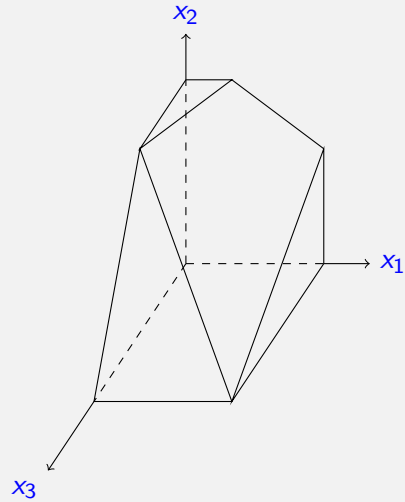
$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{aligned}$$

Вершина



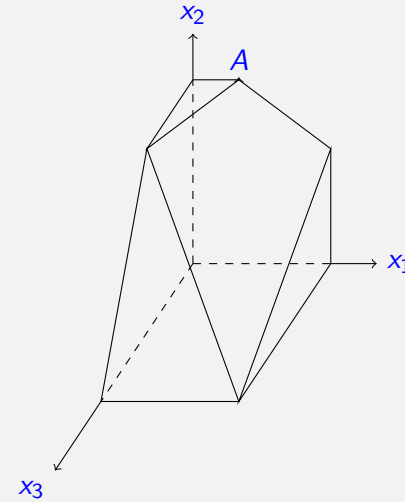
- Вершина — это точка пересечения нескольких гиперплоскостей.
- Например, A — точка, в которой неравенства $x_2 \leq 300$, $x_1 + x_2 + x_3 \leq 400$, $x_3 \geq 0$ превращаются в равенства.
- А вот гиперплоскости, задаваемые неравенствами $x_2 + 3x_3 \leq 600$ и $x_2 \geq 0$, пересекаются не по точке, а по прямой.

Вершина



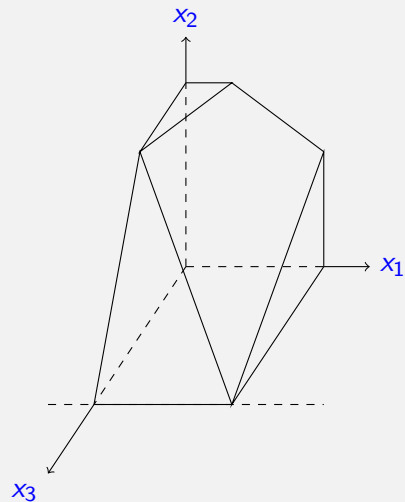
- Вершина — это точка пересечения нескольких гиперплоскостей.
- Например, A — точка, в которой неравенства $x_2 \leq 300$, $x_1 + x_2 + x_3 \leq 400$, $x_3 \geq 0$ превращаются в равенства.
- А вот гиперплоскости, задаваемые неравенствами $x_2 + 3x_3 \leq 600$ и $x_2 \geq 0$, пересекаются не по точке, а по прямой.

Вершина



- Вершина — это точка пересечения нескольких гиперплоскостей.
- Например, A — точка, в которой неравенства $x_2 \leq 300$, $x_1 + x_2 + x_3 \leq 400$, $x_3 \geq 0$ превращаются в равенства.
- А вот гиперплоскости, задаваемые неравенствами $x_2 + 3x_3 \leq 600$ и $x_2 \geq 0$, пересекаются не по точке, а по прямой.

Вершина



- Вершина — это точка пересечения нескольких гиперплоскостей.
- Например, A — точка, в которой неравенства $x_2 \leq 300$, $x_1 + x_2 + x_3 \leq 400$, $x_3 \geq 0$ превращаются в равенства.
- А вот гиперплоскости, задаваемые неравенствами $x_2 + 3x_3 \leq 600$ и $x_2 \geq 0$, пересекаются не по точке, а по прямой.

Формально

Формально

- Итак, если некоторое подмножество ограничений обращаются в равенства в единственной точке, то эту точку мы будем называть вершиной.
- Сколько нужно равенств для однозначного задания точки?
- Нужно n линейных уравнений.
- Значит, вершина задается множеством из n ограничений.
- Две вершины называются соседями, если множества задающих их ограничений пересекаются по $n - 1$ -му ограничению.

Детали

- На каждой итерации алгоритм проверяет, является ли текущая вершина оптимальной, и, если нет, определяет, в какую вершину перейти.
- Мы покажем, что данные операции легко выполнимы, если текущая вершина является началом координат.
- В противном же случае мы просто передвинем начало координат.

Начало координат

- Итак, рассмотрим задачу линейного программирования:

$$\begin{aligned} \max c^T x \\ Ax \leq b \\ x \geq 0 \end{aligned}$$
- Допустим, что $x = (0, 0, \dots, 0)$ является допустимым решением.
- Очевидно, что тогда x является вершиной (в нем неравенства $x_1 \geq 0, \dots, x_n \geq 0$ обращаются в равенства).
- Легко видеть, что x является оптимальной вершиной тогда и только тогда, когда $c_i \leq 0$ для любого i .
- Если же $c_i > 0$, то, увеличив x_i , мы увеличим и значение целевой функции.
- Но насколько можно увеличить x_i ?
- Пока одно из ограничений не обратится в равенство.

Что делать, если мы не в начале координат?

Что делать, если мы не в начале координат?

- Рассмотрим текущую вершину u .
- Пусть она задается n неравенствами $a_i \cdot x \leq b_i$.
- Пусть $y_i = b_i - a_i \cdot x$.
- Все x_i можно выразить через y и переписать систему.

Пример работы алгоритма

Исходная задача

$$\begin{aligned} \max 2x_1 + 5x_2 \\ 2x_1 - x_2 &\leq 4 \quad (1) \\ x_1 + 2x_2 &\leq 9 \quad (2) \\ -x_1 + x_2 &\leq 3 \quad (3) \\ x_1 &\geq 0 \quad (4) \\ x_2 &\geq 0 \quad (5) \end{aligned}$$

- текущая вершина: $\{(4), (5)\}$ (начало координат)
- значение целевой функции: 0
- шаг: увеличить x_2 до 3; (3) обращается в равенство
- новая вершина $\{(4), (3)\}$ имеет локальные координаты (y_1, y_2) :

$$y_1 = x_1, y_2 = 3 + x_1 - x_2$$

Пример работы алгоритма

Переписанная задача

$$\begin{aligned} \max 15 + 7y_1 - 5y_2 \\ y_1 + y_2 &\leq 7 \quad (1) \\ 3y_1 - 2y_2 &\leq 3 \quad (2) \\ y_2 &\geq 0 \quad (3) \\ y_1 &\geq 0 \quad (4) \\ -y_1 + y_2 &\leq 3 \quad (5) \end{aligned}$$

- текущая вершина: $\{(4), (3)\}$
- значение целевой функции: 15
- шаг: увеличить y_1 до 1; (2) обращается в равенство
- новая вершина $\{(2), (3)\}$ имеет локальные координаты (z_1, z_2) :
$$z_1 = 3 - 3y_1 + 2y_2, z_2 = y_2$$

Пример работы алгоритма

Переписанная задача

$$\begin{aligned} \max 22 - \frac{7}{3}z_1 - \frac{1}{3}z_2 \\ -\frac{1}{3}z_1 + \frac{5}{3}z_2 &\leq 6 \quad (1) \\ z_1 &\geq 0 \quad (2) \\ z_2 &\geq 0 \quad (3) \\ \frac{1}{3}z_1 - \frac{2}{3}z_2 &\leq 1 \quad (4) \\ \frac{1}{3}z_1 + \frac{1}{3}z_2 &\leq 4 \quad (5) \end{aligned}$$

- текущая вершина: $\{(2), (3)\}$
- значение целевой функции: 22
- оптимальная вершина
- решив (2), (3) в исходной задаче, найдем оптимальную вершину $(x_1, x_2) = (1, 4)$

План лекции

- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

Начальная вершина

Начальная вершина

- Пока что каждый раз мы использовали начало координат в качестве начальной вершины.
- Но начало координат может и не быть допустимым решением.
- Оказывается, что нахождение начальной вершины тоже можно свести к задаче линейного программирования.
- Для этого запишем нашу задачу в таком виде:

$$\begin{aligned} \min c^T x \\ Ax = b \\ x \geq 0 \end{aligned}$$

- НУО, считаем, что все координаты b неотрицательны.

Поиск начальной вершины

Поиск начальной вершины

- Запишем тогда такую задачу:
 - ▶ Для каждого равенства системы $Ax = b$ введем новую переменную $z_i \geq 0$.
 - ▶ Прибавим z_i к левой части каждого из равенств.
 - ▶ В качестве целевой функции, которую нужно минимизировать, рассмотрим функцию $z_1 + \dots + z_m$.
- У полученной задачи, очевидно, есть допустимая вершина:
 $z_i = b_i, x_i = 0$.
- Начав с нее, мы можем решить новую задачу.

Решение новой задачи

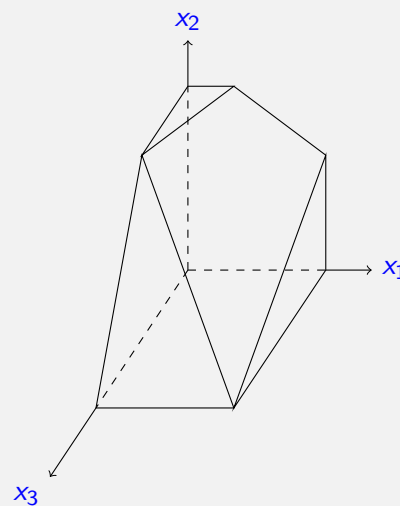
Решение новой задачи

- Итак, рассмотрим решение новой задачи.
- Если в нем $z_1 + \dots + z_m = 0$, то $z_1 = \dots = z_m = 0$
- Тогда значения координат x в этом решении дают нам допустимую вершину исходной задачи.
- Если же для новой задачи $\min z_1 + \dots + z_m > 0$, то у начальной задачи просто нет ни одного допустимого решения.
- Именно таким образом симплекс-метод распознает задачи, у которых нет ни одного допустимого решения.

План лекции

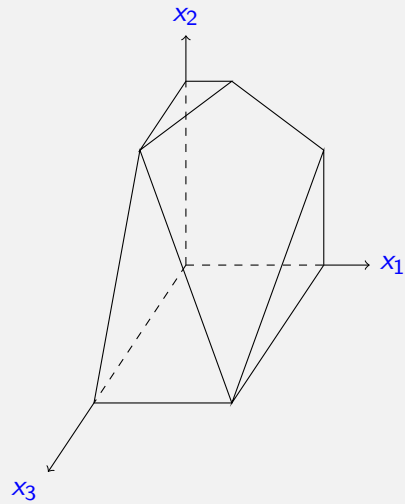
- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

Вырожденная вершина



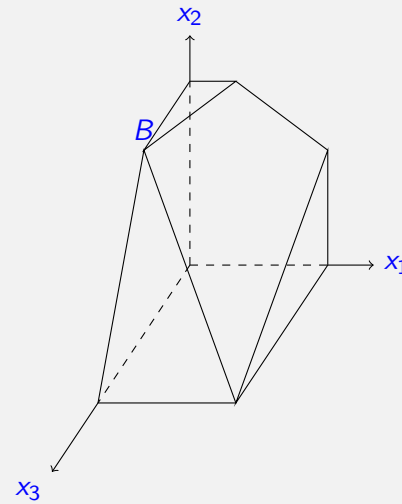
- Вершина B называется **вырожденной**, поскольку она является пересечением более чем $n = 3$ гиперплоскостей: (2), (3), (4), (5).
- Такая вершина является своим собственным соседом.
- Избавиться от таких вершин можно, слегка изменив вектор b :
 $b_i \pm \epsilon_i$.

Вырожденная вершина



- Вершина B называется **вырожденной**, поскольку она является пересечением более чем $n = 3$ гиперплоскостей: (2), (3), (4), (5).
- Такая вершина является своим собственным соседом.
- Избавиться от таких вершин можно, слегка изменив вектор b : $b_i \pm \epsilon_j$.

Вырожденная вершина



- Вершина B называется **вырожденной**, поскольку она является пересечением более чем $n = 3$ гиперплоскостей: (2), (3), (4), (5).
- Такая вершина является своим собственным соседом.
- Избавиться от таких вершин можно, слегка изменив вектор b : $b_i \pm \epsilon_j$.

Как симплекс-метод распознает неограниченные задачи?

Как симплекс-метод распознает неограниченные задачи?

- При поиске соседней вершины алгоритм заменяет одно из равенств, задающих текущую вершину.
- Полученная система может иметь бесконечное число решений.

План лекции

- 1 Введение
- 2 Потоки в сетях
- 3 Максимальное паросочетание в двудольном графе
- 4 Двойственность
- 5 Симплекс-метод
 - Алгоритм
 - Начальная вершина
 - Вырожденные вершины
 - Время работы

Время работы

Время работы

- В общем случае количество вершин, которые алгоритму придется перебрать ограничено сверху числом $\binom{m+n}{n}$.
- Но это выражение экспоненциально от n .
- Более того, есть задачи, на которых симплекс-метод действительно будет работать экспоненциально долго.
- Такие задачи, однако, на практике не возникают, и на практических задачах симплекс-метод работает довольно быстро.
- Известны и алгоритмы с доказуемыми полиномиальными верхними оценками на время работы.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Задачей линейного программирования называется задача, в которой требуется выполнить линейные ограничения и максимизировать линейную функцию.
- Если множество допустимых решений существует и ограничено, то оно является выпуклым многогранником.
- Симплекс-метод движется по вершинам многогранника, улучшая значение целевой функции.
- Симплекс-метод хорошо решает практические задачи, но для него есть экспоненциальные нижние оценки времени работы.
- Известны и алгоритмы с доказуемыми полиномиальными верхними оценками на время работы.
- Задача о нахождении максимального потока в сети — задача линейного программирования.