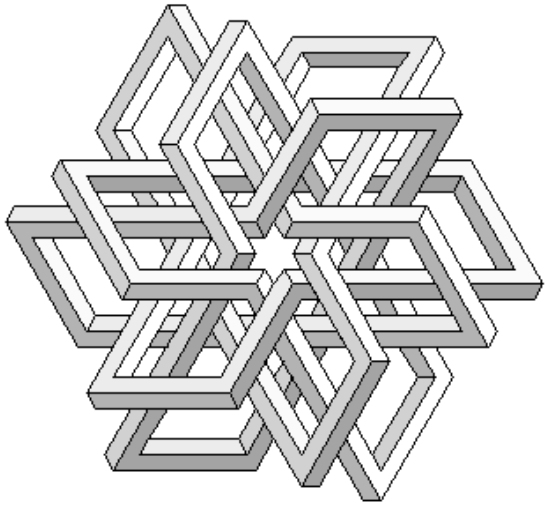


Приближенные алгоритмы

НГУ, ауд. 5273
пятница 14:30

Лектор: Кононов Александр Вениаминович

Экзамен

- Устный опрос (количество вопросов зависит от числа посещений занятий)
- Ответ на вопрос в билете (при подготовке можно пользоваться литературой)
- Решение задач

А.В. Кононов, П.А. Кононова, Приближенные алгоритмы для NP-трудных задач, РИЦ-НГУ, 2014.

http://www.math.nsc.ru/LBRT/k5/app.html

Объект исследования

NP-трудные задачи дискретной оптимизации

Что нужно знать!

- Задача
- Индивидуальная задача
- Оптимизационная задача
- Размер входа индивидуальной задачи
- Алгоритм
- Временная сложность алгоритма
- Полиномиальный алгоритм
- Задача линейного программирования
- NP-трудная задача

А также знать и уметь

- основы теории графов
- основы комбинаторики
- основы теории вероятностей
- формулировать комбинаторные задачи как задачи ЦЛП (или выпуклого программирования)
- теорию двойственности задач линейного программирования
- геометрическую интерпретацию задач линейного программирования

Литература по комбинаторной оптимизации

- ***М.Гэри, Д.Джонсон,*** Вычислительные машины и труднорешаемые задачи, Мир, 1982.
- ***Х. Пападимитриу, К. Стайглиц,*** Комбинаторная оптимизация: Алгоритмы и сложность, Мир, 1984.
- ***Т. Кормен, Ч. Лейзерсон, Р. Риверст, К. Штайн,*** Алгоритмы: построение и анализ, Вильямс, 2009.
- ***Дж. Клейнберг, Е. Тардос,*** Алгоритмы: разработка и применение, Питер, 2017.
- ***Korte B., Vygen J.,*** Combinatorial Optimization: theory and algorithms, (Algorithms and Combinatorics 21), Springer, Berlin, 2010.

Задача

Задача П определяется следующей информацией:

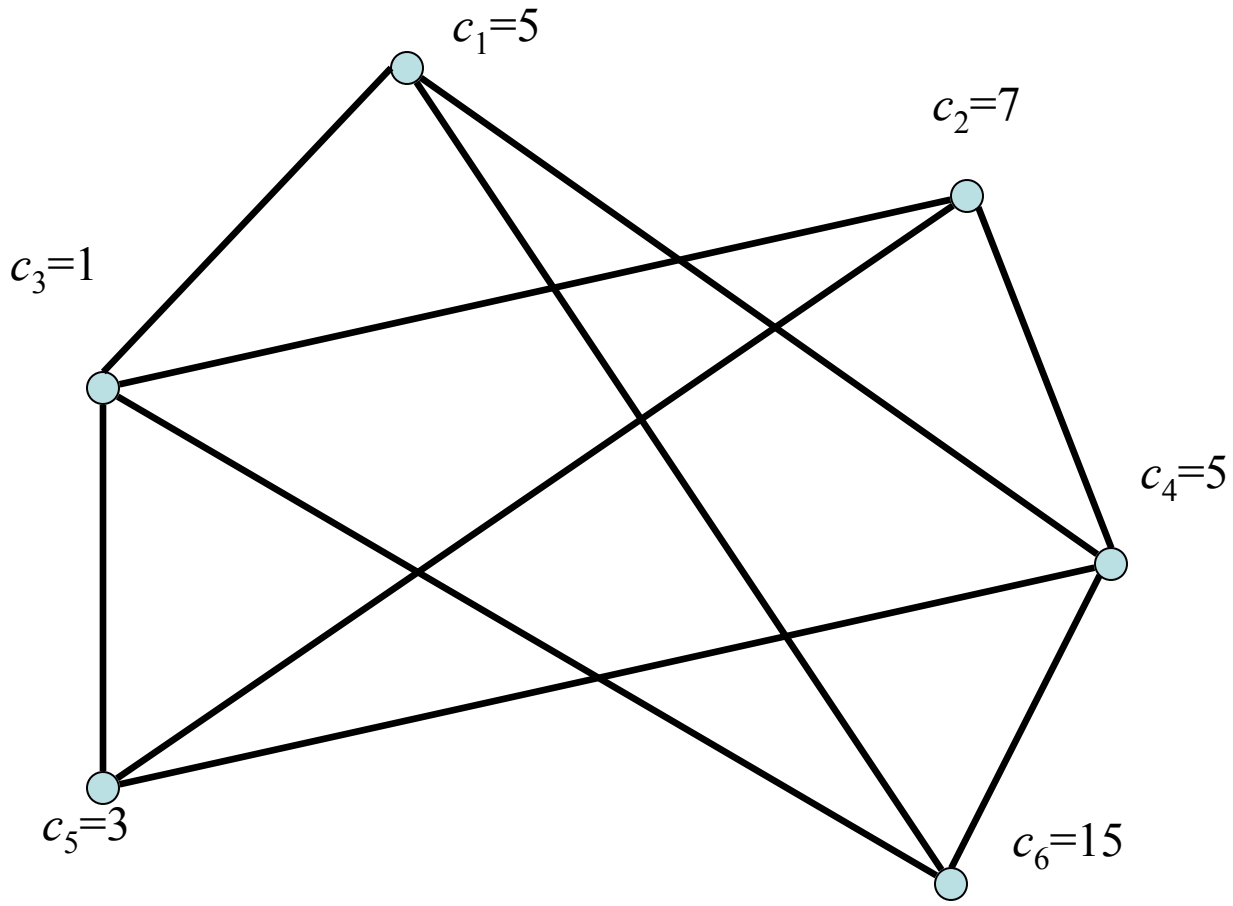
- общим списком всех ее параметров
- формулировкой тех свойств, которым должен удовлетворять ***ответ*** или, другими словами, решение задачи.

Индивидуальная задача *I* получается из задачи П, если всем параметрам задачи П присвоить конкретные значения.

Задача о вершинном покрытии

- *Дано*: Связный граф $G = (V, E)$, веса вершин c: $V \to \mathbf{Q}^+$.
- *Найти* вершинное покрытие наименьшего веса.
- Вершинное покрытие это множество вершин $V' \subseteq V$ такое, что каждое ребро имеет граничную точку в V'.

Пример



Размер входа

Размер входа индивидуальной задачи с рациональными данными равен числу бит, требуемых для ее двоичного представления в некоторой «наилучшей» бинарной кодировке.

Оптимизационная задача

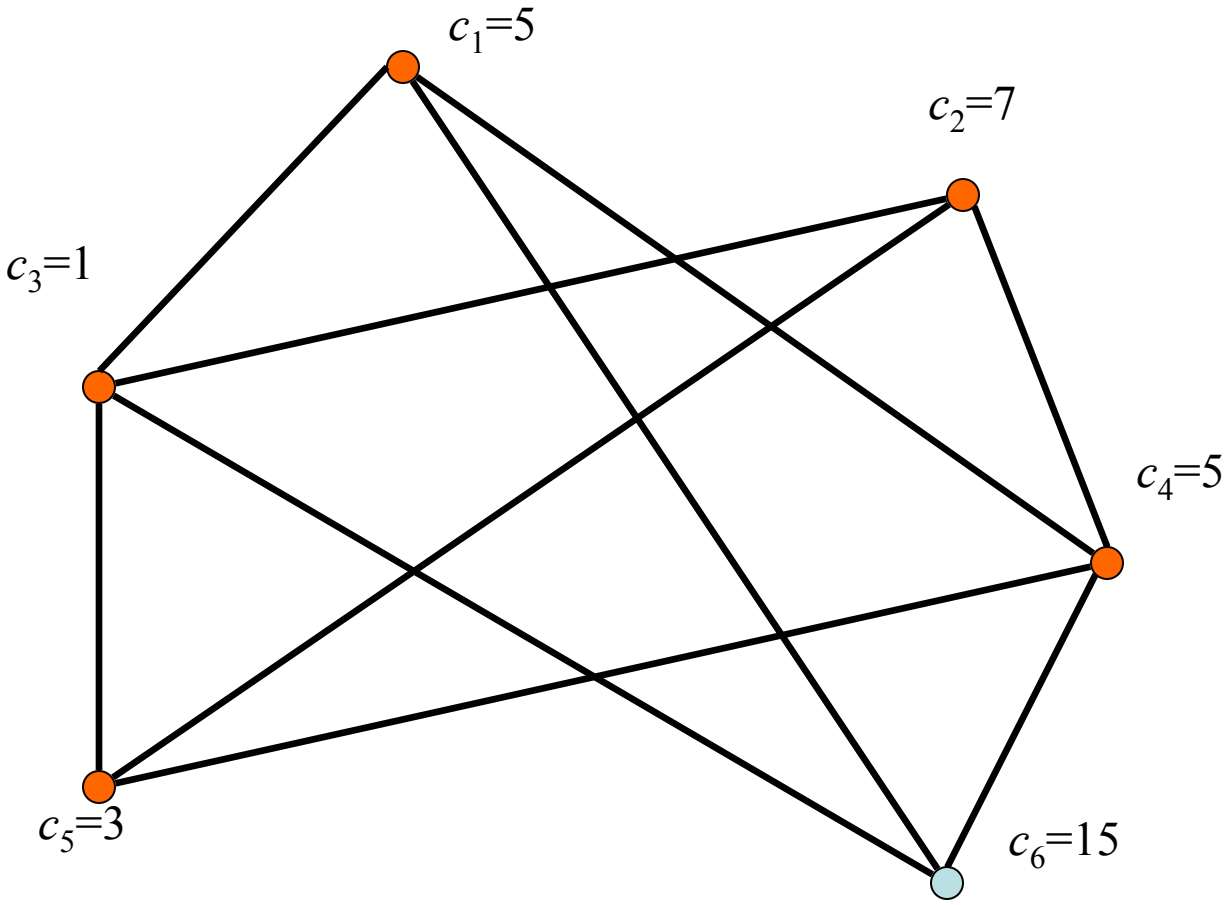
Оптимизационная задача П есть либо задача *минимизации*, либо задача *максимизации* и состоит из

- множества Ω_{Π} *индивидуальных задач*;
- для каждой $I \in \Omega_{\Pi}$ конечного множества Sol_{Π} *допустимых решений* индивидуальной задачи I;
- функции h_{Π}, сопоставляющей каждой индивидуальной задаче $I \in \Omega_{\Pi}$ и каждому *допустимому решению* $\sigma \in Sol_{\Pi}$ некоторое рациональное число $y(I, \sigma)$, называемое величиной решения σ.

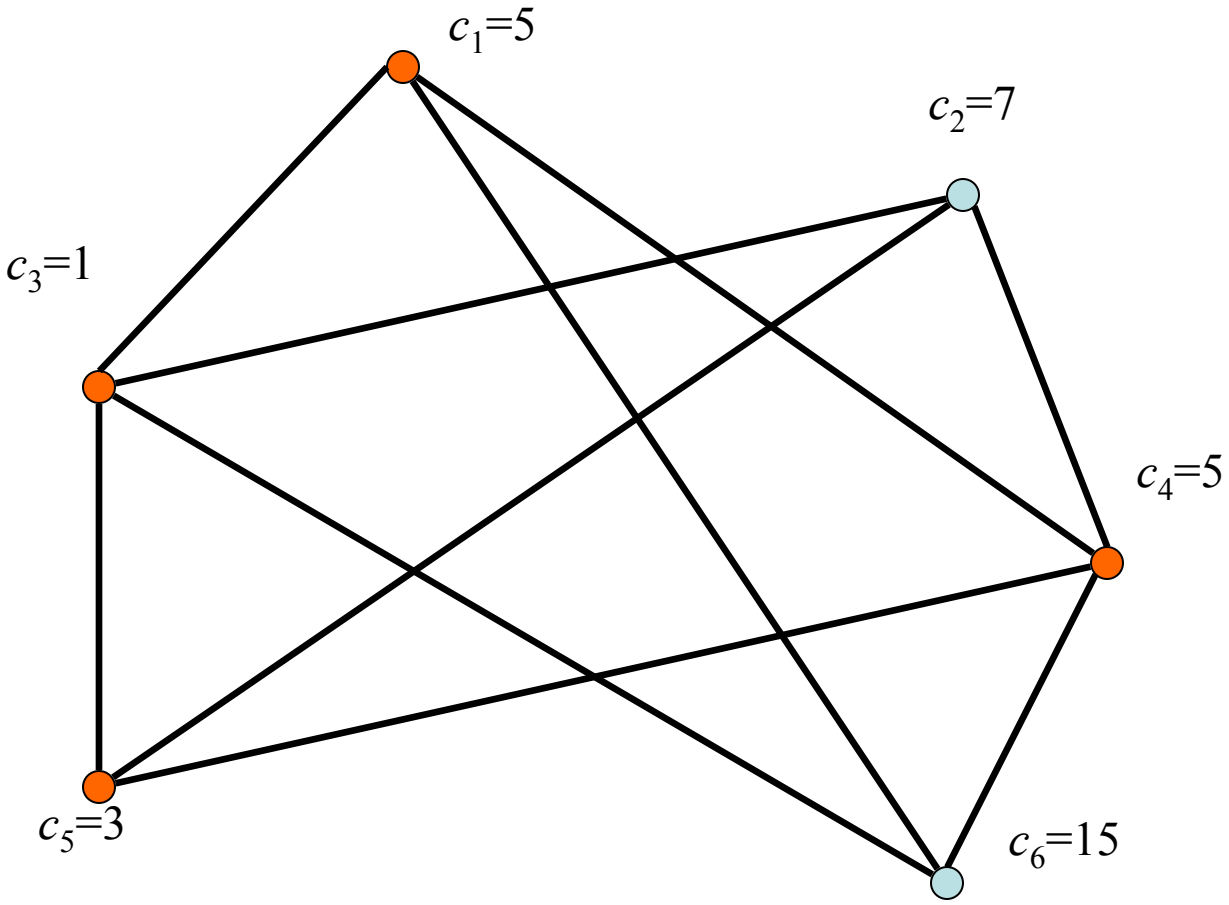
Оптимальное решение

- Если П — задача минимизации, то **оптимальным решением** индивидуальной задачи $I \in \Omega_{\Pi}$ является такое допустимое решение $\sigma^* \in Sol_{\Pi}$, что для всех $\sigma \in Sol_{\Pi}$ выполнено неравенство $y(I, \sigma^*) \leq y(I, \sigma)$.
- Для обозначения величины $y(I, \sigma^*)$ оптимального решения индивидуальной задачи I будет использоваться символ $OPT(I)$.

Допустимое решение



Оптимальное решение



Алгоритм

- Множество исходных данных (Вход)
- Последовательность инструкций, каждая из которых может быть представлена цепочкой элементарных шагов.
- Для каждого допустимого входа алгоритм в процессе вычисления выполняет единственно определенную серию элементарных шагов и выдает некоторый ответ.

Временная сложность алгоритма

- Время работы алгоритма удобно выражать в виде функции от одной переменной, характеризующей «размер» индивидуальной задачи, то есть от ее размера входа.
- Временная сложность алгоритма — это функция, которая каждому входу размера x ставит в соответствие максимальное по всем индивидуальным задачам время (число элементарных шагов), затрачиваемое алгоритмом на решение индивидуальных задач данного размера.

Полиномиальный алгоритм

- Алгоритм с рациональным входом называется **полиномиальным**, если существует целое k такое что алгоритм работает время $O(x^k)$, где x есть размер входа и все числа, используемые алгоритмом в процессе вычислений ограничены величиной $O(x^k)$ бит.
- Алгоритм с произвольным входом называется **сильно полиномиальным**, если существует целое k такое что алгоритм работает время $O(n^k)$ на любом входе состоящим из n чисел и он полиномиальный на рациональном входе.
- Если $k = 1$ алгоритм называется **линейным**.

NP-трудная задача

- Задача П является NP-трудной, если к ней сводится некоторая NP-полная задача.
- Существование точного полиномиального алгоритма для NP-трудной задачи влечет **P = NP**.

NP-трудная задача

- Задача П является NP-трудной, если к ней сводится некоторая NP-полная задача.
- Существование точного полиномиального алгоритма для NP-трудной задачи влечет **P = NP**.

Почти все интересные дискретные оптимизационные задачи — NP-трудны.

Что делать с NP-трудными задачами?

- Решать точно «переборными» алгоритмами
- Решать приближенно
  - с апостериорной оценкой точности
  - с априорной оценкой точности

Что делать с NP-трудными задачами?

- Решать точно «переборными» алгоритмами
- Решать приближенно
  - с апостериорной оценкой точности
  - с априорной оценкой точности

Мы будем строить приближенные полиномиальные алгоритмы с априорной оценкой точности.

Приближенный алгоритм

Полиномиальный алгоритм **A** называется **ρ-приближенным алгоритмом** для задачи минимизации П, если для каждой индивидуальной задачи $I \in \Omega_{\Pi}$,

$$A(I) \leq \rho OPT(I).$$

Полиномиальная приближенная схема (PTAS)

Семейство приближенных алгоритмов для задачи П, $\{A_\varepsilon\}_\varepsilon$ называется **полиномиальной приближенной схемой**, если алгоритм A_ε — $(1+\varepsilon)$-приближенный алгоритм и время его работы ограничено полиномом от размера входа при фиксированном ε.

Вполне полиномиальная приближенная схема (FPTAS)

Семейство приближенных алгоритмов для задачи П, $\{A_\varepsilon\}_\varepsilon$ называется **вполне полиномиальной приближенной схемой**, если алгоритм A_ε — $(1+\varepsilon)$-приближенный алгоритм и время его работы ограничено полиномом от размера входа и **1/ε**.

Алгоритм

- Как построить приближенный алгоритм?
  - Изучение комбинаторной структуры задачи.
  - Изучение свойств оптимального решения.
  - Поиск алгоритмической техники, которая использует эти свойства.
- Обобщение и расширение техники и опыта, накопленного при построении полиномиальных алгоритмов.

Задача линейного программирования

$$z(x) = c_1 x_1 + \ldots + c_n x_n \rightarrow \min$$

$$a_{11} x_1 + \ldots + a_{1n} x_n \geq b_1$$
$$\vdots$$
$$a_{m1} x_1 + \ldots + a_{mn} x_n \geq b_m$$
$$x_i \geq 0 \ \text{ for } \ i = 1, \ldots, n.$$

Полиномиально разрешимые задачи

- Задача об остовном дереве минимального веса
- Задача о максимальном потоке
- Задача о максимальном паросочетании
- Задача о k-факторе минимального веса

●●●

Вопрос о качестве алгоритма

- Как оценить качество алгоритма?

Вопрос о качестве алгоритма

- Как оценить качество алгоритма?
- Сравнить стоимость получаемых решений со стоимостью оптимального решения.

Вопрос о качестве алгоритма

- Как оценить качество алгоритма?
- Сравнить стоимость получаемых решений со стоимостью оптимального решения.
- Найти стоимость оптимального решения также сложно, как и само оптимальное решение.

Как оценить качество алгоритма?

- Найти «хорошую» полиномиально вычислимую нижнюю оценку на стоимость оптимального решения.

Задача о вершинном покрытии наименьшей мощности

- *Дано*: Связный граф $G = (V, E)$.
- *Найти* вершинное покрытие наименьшей мощности.

Максимальное паросочетание

Дан граф $G = (V, E)$, подмножество ребер $M \subseteq E$ называется **паросочетанием**, если никакие два ребра из M не смежны, то есть не имеют общей граничной точки.

- Паросочетание максимальное по включению.
- Паросочетание максимальное по мощности.

Размер паросочетания максимального по включению является нижней границей на стоимость оптимального решения задачи «Вершинное покрытие наименьшей мощности».

Алгоритм «Простой»

1. Найти в графе G произвольное паросочетание M максимальное по включению.
2. Выдать множество вершин, попавших в паросочетание M.

Оценка качества алгоритма «Простой»

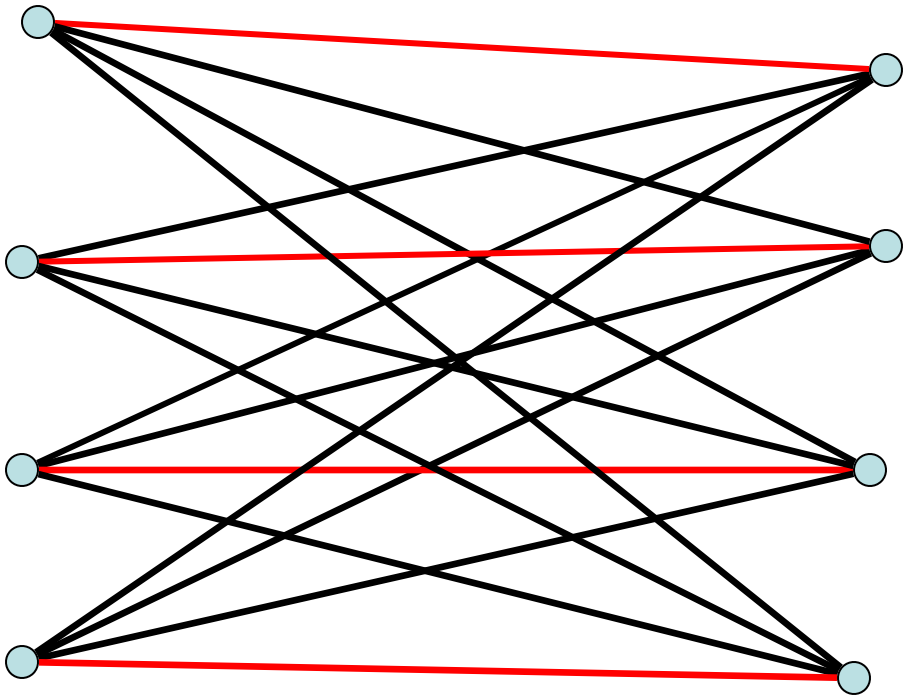
Теорема 1.1

Алгоритм «Простой» является 2-приближенным алгоритмом для задачи «Вершинное покрытие наименьшей мощности».

Качество анализа, нижней оценки, …

- Можно ли за полиномиальное время получать решение с гарантированной оценкой лучше чем 2?
  - Можно ли улучшить анализ качества алгоритма «Простой»?
  - Можно ли построить алгоритм, который всегда находит решение с оценкой лучше чем 2 от предложенной нижней границы?
  - Существует ли другой алгоритм с гарантированной оценкой лучше чем 2?

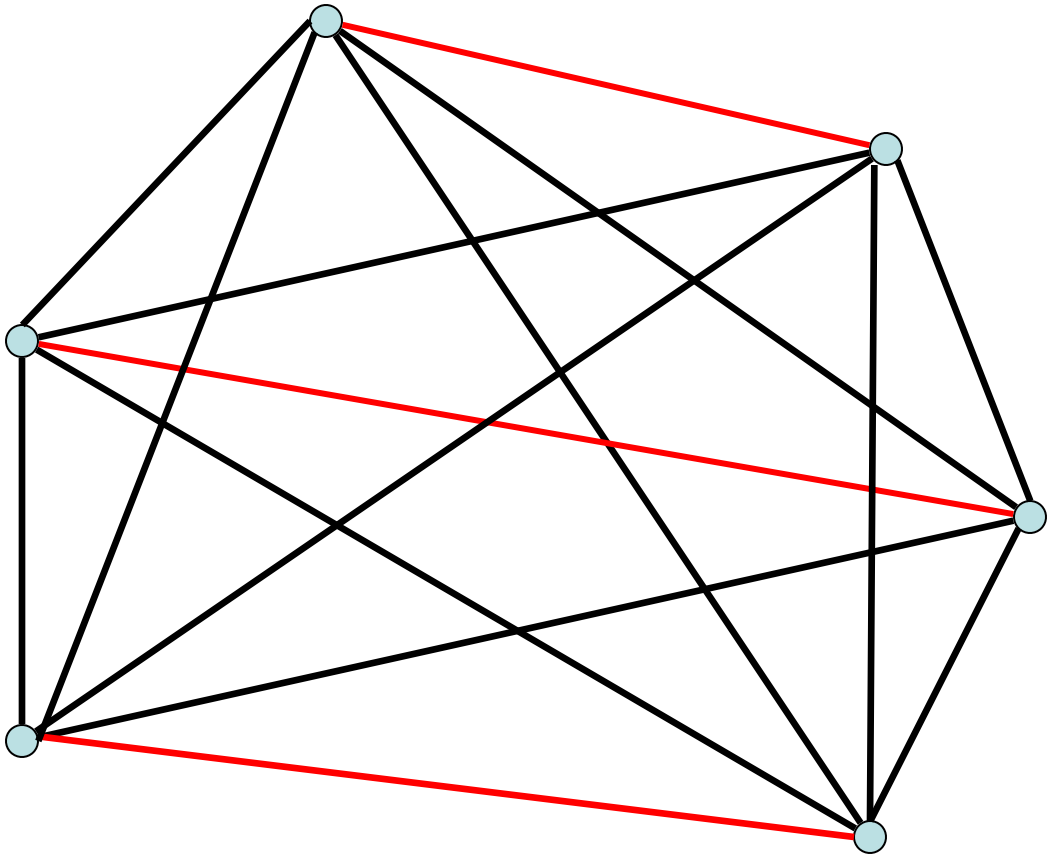
Точность оценки



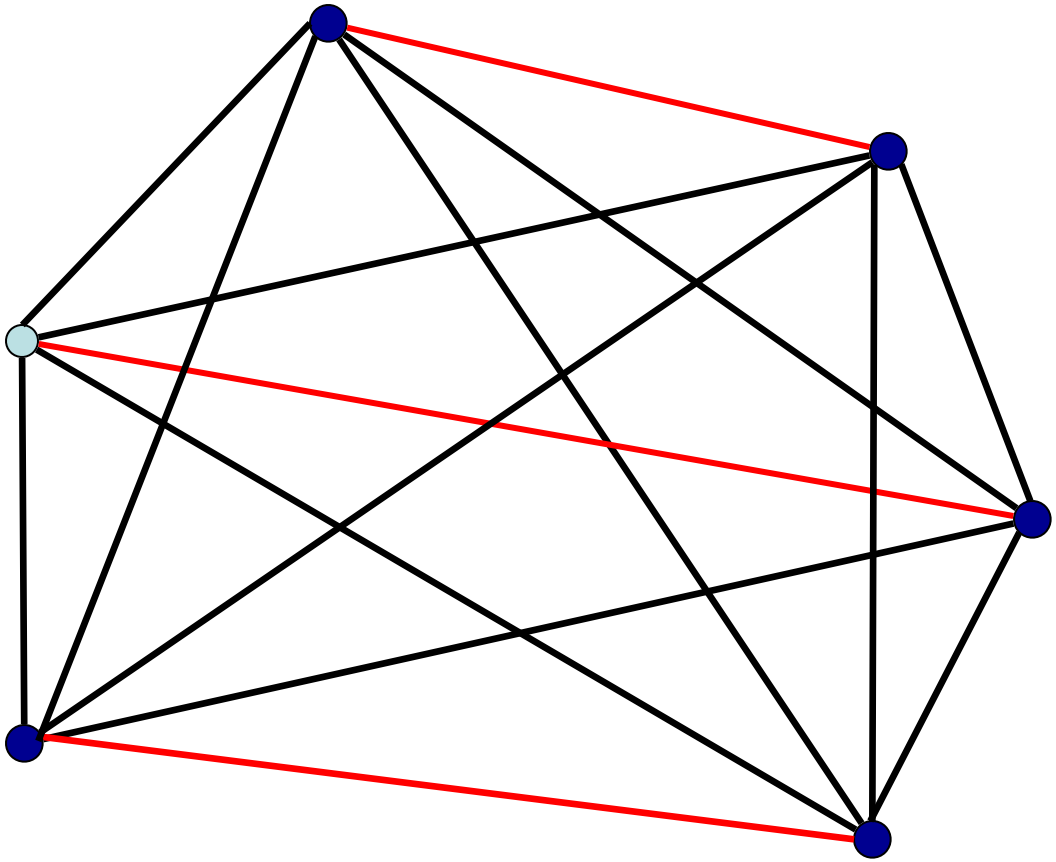
Качество анализа, нижней оценки, …

- Можно ли за полиномиальное время получать решение с гарантированной оценкой лучше чем 2?
  - Можно ли улучшить анализ качества алгоритма «Простой»? Ответ нет.
  - Можно ли построить алгоритм, который всегда находит решение с оценкой лучше чем 2 от предложенной нижней границы?
  - Существует ли другой алгоритм с гарантированной оценкой лучше чем 2?

Сравнение нижней оценки и оптимального решения



Сравнение нижней оценки и оптимального решения



Качество анализа, нижней оценки, …

- Можно ли за полиномиальное время получать решение с гарантированной оценкой лучше чем 2?
  - Можно ли улучшить анализ качества алгоритма «Простой»? Ответ нет.
  - Можно ли построить алгоритм, который всегда находит решение с оценкой лучше чем 2 от предложенной нижней границы? Ответ нет.
  - Существует ли другой алгоритм с гарантированной оценкой лучше чем 2?

Существует ли другой алгоритм с гарантированной оценкой лучше чем 2?

Khot S., Regev O., Vertex cover might be hard to approximate to within $2-\varepsilon$ // *Journal of Computer and System Science* 74(3): 335-349, (2008).

Результат основан на выполнении двух гипотез

- $P \neq NP$
- UGC (гипотеза об уникальной игре)

$1|r_j,q_j|L_{\max}$

- $J = \{1,..., n\}$ – работы.
- Одна машина.
- работа j :
  - $p_j > 0$ – длительность работы j,
  - $r_j > 0$ – момент поступления работы j,
  - $q_j > 0$ – время доставки работы j.
- Прерывания запрещены.
- Машина обслуживает не более одной работы одновременно.
- Минимизировать самую позднюю доставку ($L_{\max}$).

Допустимое расписание σ

- работа j :
  - $s_j(\sigma) \geq 0$ – момент начала обслуживания работы j,
  - $C_j(\sigma) \geq 0$ – момент завершения обслуживания работы j,
- Работа не может начаться до ее поступления: $s_j \geq r_j$.
- Прерывания запрещены: $C_j(\sigma) = s_j(\sigma) + p_j$.
- Машина обслуживает не более одной работы одновременно: $[s_i(\sigma), C_i(\sigma)) \cap [s_j(\sigma), C_j(\sigma)) = \varnothing$.
- Минимизировать $L_{\max} = \max\limits_{j=1,\ldots,n} (s_j + p_j + q_j)$.

Нижние оценки

$$L^*_{\max} \geq P = \sum_{j=1}^{n} p_j;$$

$$L^*_{\max} \geq D = \max_{j=1,\ldots,n} (r_j + p_j + q_j)$$

$$LB = \max\{P, D\}$$

Жадный алгоритм (GL)

- Задан список работ в произвольном порядке.
- Когда машина освобождается, первая «доступная» работа из списка назначается на эту машину.
- Работа J_j — доступна в момент t, если $r_j \leq t$.

Алгоритм *GL для* $1|r_j|L_{\max}$

Теорема 1.2

$L_{\max}^{GL} < 2L_{\max}^{*}$, и эта оценка точна.

Доказательство

Рассмотрим расписание σ, полученное алгоритмом *GL*.

- J_k : $\boxed{L_{\max}^{GL} = C_k + q_k.}$

- В расписании σ не может быть простоя между r_k и s_k.

$$L_{\max}^{GL} = s_k + p_k + q_k < (r_k + P) + p_k + q_k$$
$$= (r_k + p_k + q_k) + P \le 2L_{\max}^{*}.$$

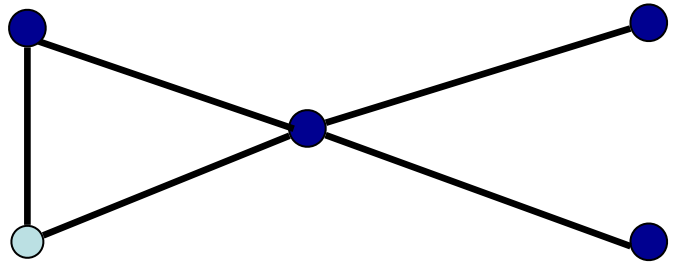
Качество анализа, нижней оценки, …

- Можно ли за полиномиальное время получать решение с гарантированной оценкой лучше чем 2?
  - Можно ли улучшить анализ качества жадного алгоритма ?
  - Можно ли построить алгоритм, который всегда находит решение с оценкой лучше чем 2 от предложенной нижней границы?
  - Существует ли другой алгоритм с гарантированной оценкой лучше чем 2?

Литература

- А.В. Кононов, П.А. Кононова, *Приближенные алгоритмы для NP-трудных задач,* РИЦ-НГУ, 2014.
- *Approximation Algorithms for NP-hard problems*, edited by D.Hochbaum, PWS Publishing Company, 1997.
- V. Vazirani**,** *Approximation Algorithms*, Springer-Verlag, Berlin, 2001.
- P. Schuurman, G. Woeginger**,** *Approximation Schemes – A Tutorial*, chapter of the book “Lecture on Scheduling”, to appear in 2008.
- D. P. Williamson, D. B. Shmoys, *The Design of Approximation Algorithms,* Cambridge University Press, 2011.

Практика



1. Запишите прямую (ЦЛП) и двойственную программу решающую задачу о вершинном покрытии наименьшей мощности для графа *G*.
2. Принадлежит ли задача распознавания проверить существует ли в графе остовное дерево размера *D* или меньше к классу *NP* и почему?
3. Докажите точность нижней оценки жадного алгоритма *GL* в теореме 1.2.