Лекция 5. Рандомизированные алгоритмы

Мельников Андрей Андреевич

Новосибирский Государственный Университет
Факультет Информационных Технологий
http://vk.com/fit2013_tdm/

13 апреля, 2013 г.

Содержание лекции

Определения

Рандомизированный алгоритм (RA)

алгоритм, содержащий процедуры, основанные на случайном выборе.

Алгоритм типа Лас-Вегас

Для любого входа I задачи P алгоритм A находит решение почти наверное. Время работы алгоритма является случайной величиной.

Алгоритм типа Монте-Карло

Для любого входа I задачи P алгоритм A находит решение с некоторой ненулевой вероятностью.

Пример алгоритма типа Лас-Вегас

Алгоритм быстрая сортировка $QS(S)$

- *Вход: массив* S.
- *Выход: массив* S, *отсортированный по неубыванию.*

1. *Случайным образом выбрать разделяющий элемент* $y \in S$.
2. *Если* $|S| = 1$, *то вернуть* S,
   иначе разделить множество S *на три подмножества:*
   $S_< := \{x \in S | x < y\}$,
   $S_= := \{x \in S | x = y\}$,
   $S_> := \{x \in S | x > y\}$
3. *Вернуть* $QS(S_<), S_=, QS(S_>)$.

Сложность алгоритма $QS(S)$

Теорема 1

Пусть $n = |S|$. Ожидаемое число сравнений в алгоритме $QS(S)$ не превосходит $2nH_n$.

Доказательство:

Пусть $s_1 \leqslant s_2 \leqslant \ldots \leqslant s_n$ — результат работы алгоритма $QS(S)$.
Пусть

$$x_{ij} = \begin{cases} 1, & s_i \text{ и } s_j \text{ сравнивались между собой в } QS \\ 0, & \text{иначе} \end{cases}, i < j.$$

x_{ij} — бернуллиевская случайная величина.
Пусть $t = \sum_{i=1}^{n} \sum_{j>i} x_{ij}$ — общее число сравнений.

Доказательство (продолжение):

Заметим, что s_i и s_j сравниваются тогда и только тогда, когда они на некотором этапе попали в одно из множеств $S_<$ или $S_>$ и, кроме того, s_i либо s_j выбран в качестве разделяющего элемента в этом множестве. Если s_i и s_j находятся в одном множестве S, то в нем же находятся все промежуточные элементы: $\{s_i, s_{i+1}, \ldots, s_{i+j-1}, s_j\} \in S$. Стало быть, p_{ij} — вероятность того, что s_i и s_j сравниваются, не превосходит величины $\frac{2}{j-i+1}$.

$$\mathbb{E}[t] = \sum_{i=1}^{n} \sum_{j>i} p_{ij} \leqslant \sum_{i=1}^{n} \sum_{j>i} \frac{2}{j-i+1} = \sum_{i=1}^{n} \sum_{k=1}^{n-i+1} \frac{2}{k} \leqslant$$

$$\leqslant 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} = 2nH_n \approx 2n \ln n + \Theta(1) = O(n \log n)$$

Пример алгоритма типа Монте-Карло. Задача о минимальном разрезе

- Дано: G — связный неориентированный мультиграф с n вершинами

Мультиграф

граф, в котором между любой парой вершин может быть несколько ребер

Разрез в графе G

*множество **ребер**, удаление которых приводит к распаду графа на две или более компонент связности*

- Найти разрез минимальной мощности

Рандомизированный алгоритм для задачи о минимальном разрезе

Алгоритм стягивания вершин

- *Вход: мультиграф* $G = (V, E)$
- *Выход: разрез* C

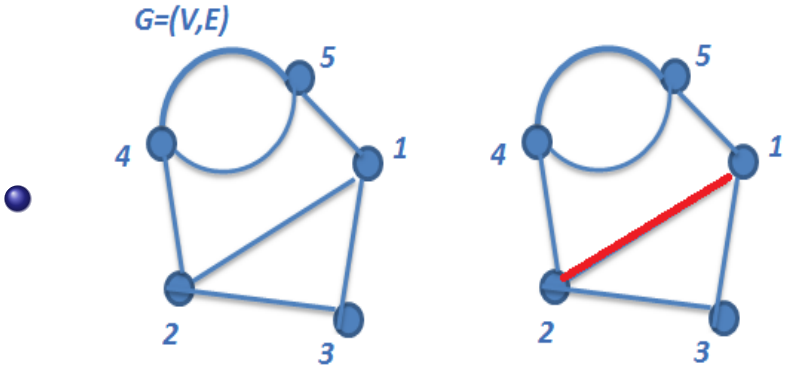
Пока не останется две вершины выполнить:

1. *выбрать случайным образом ребро* $e = (u_1, u_2)$
2. *слить вершины* u_1 *и* u_2 *в одну метавершину* w*:*

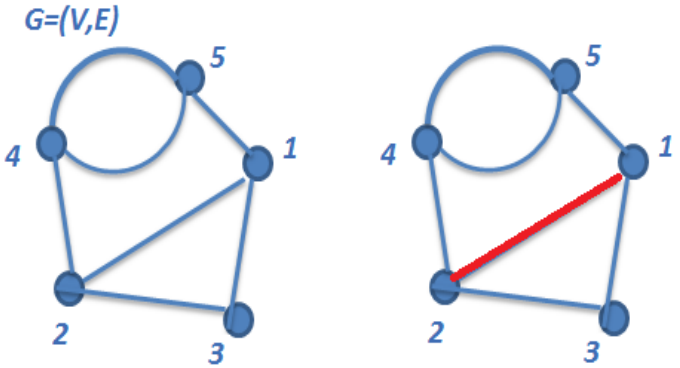
   удалить все ребра между u_1 *и* u_2*,*

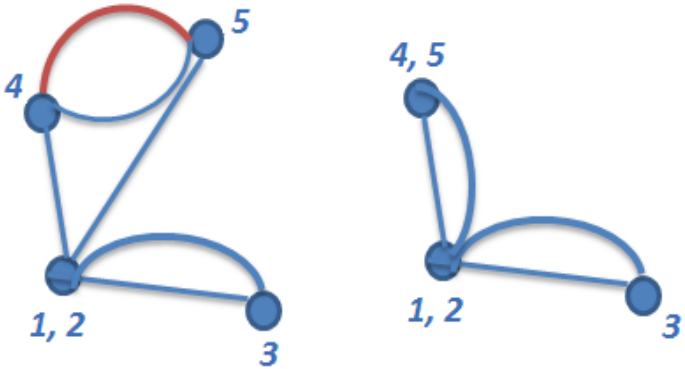
   ребра из E *инцидентные* u_1 *или* u_2 *перенаправить в* w

Рандомизированный алгоритм для задачи о минимальном разрезе

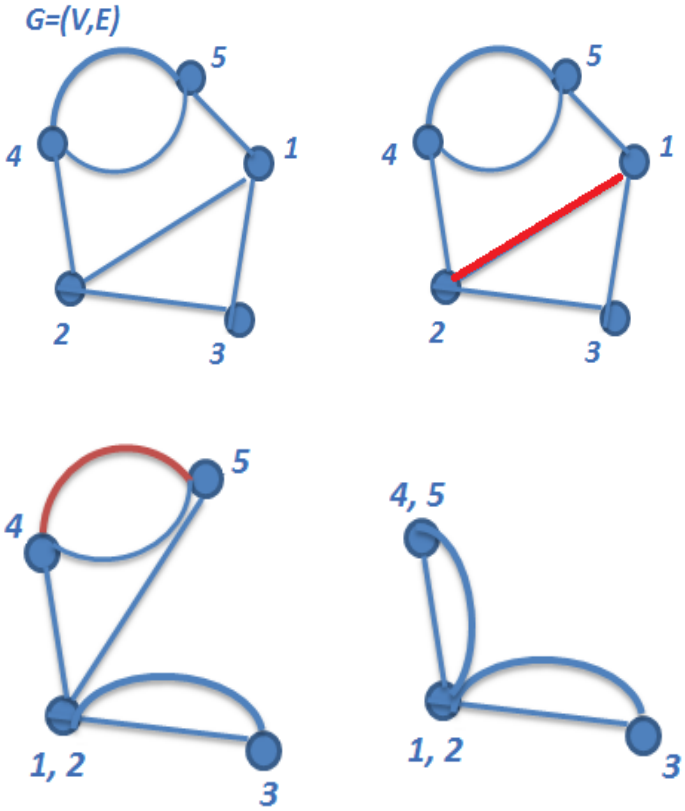


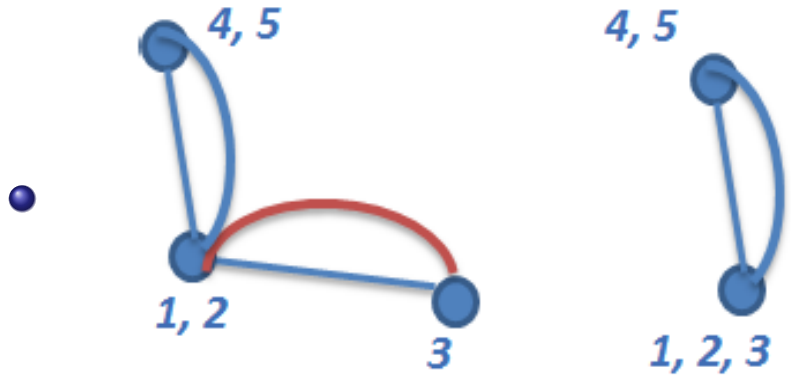
Рандомизированный алгоритм для задачи о минимальном разрезе

- 

- 

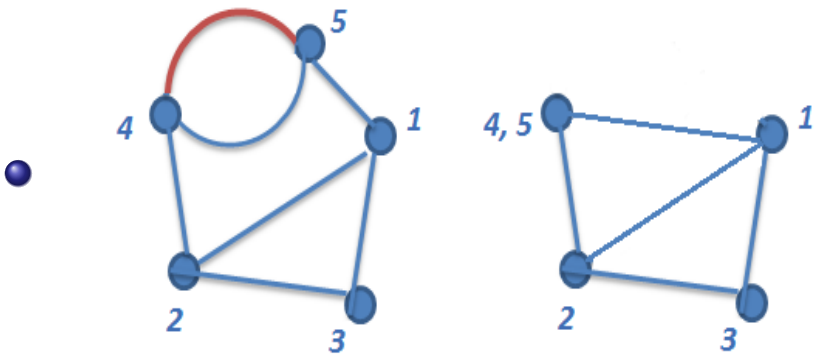
Рандомизированный алгоритм для задачи о минимальном разрезе

- 

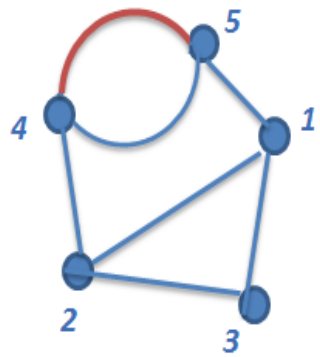
- 

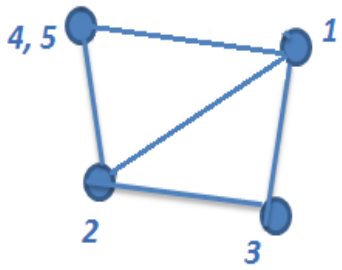
минимальный разрез

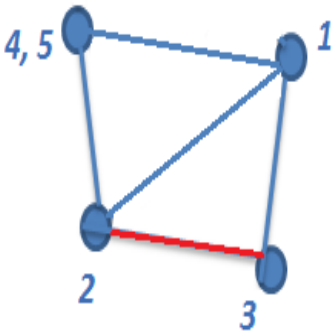
Рандомизированный алгоритм для задачи о минимальном разрезе

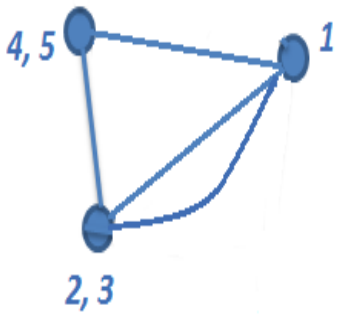


Рандомизированный алгоритм для задачи о минимальном разрезе

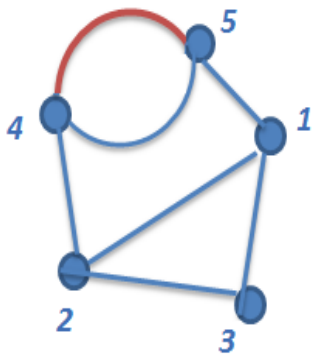
- 

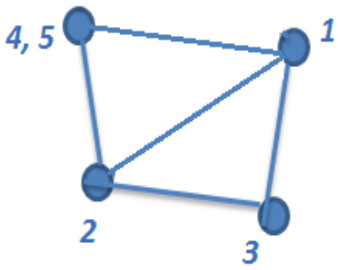


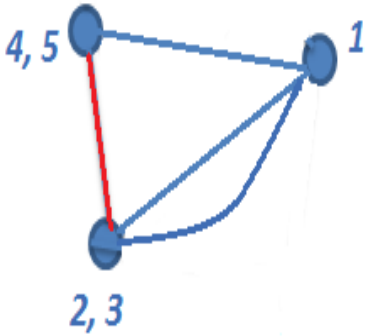
- 

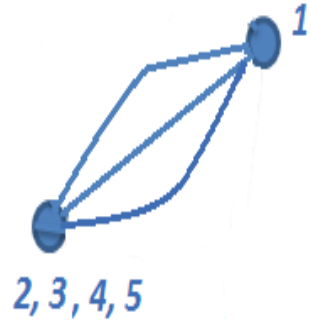


Рандомизированный алгоритм для задачи о минимальном разрезе

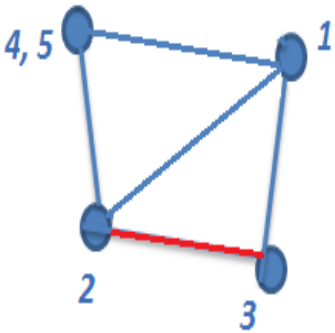
- 

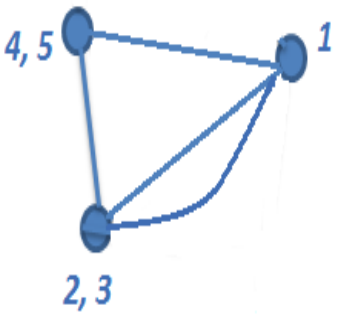


- 

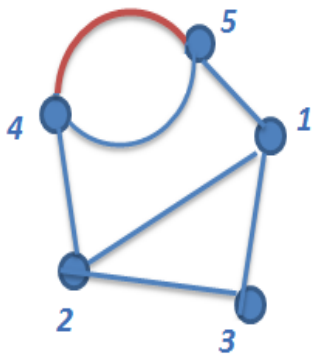


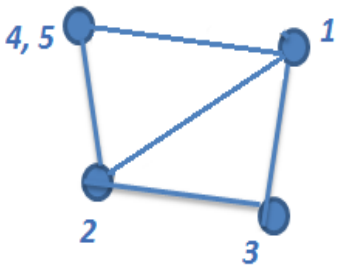
неминимальный разрез

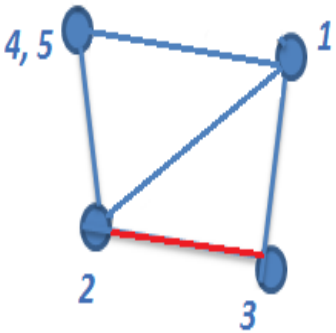
- 

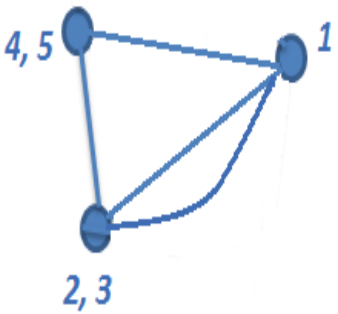


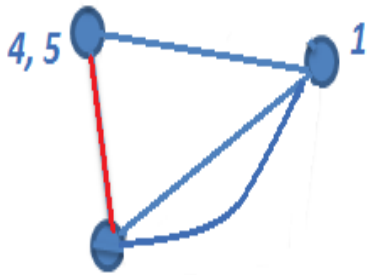
Рандомизированный алгоритм для задачи о минимальном разрезе

- 



- 



- 

неминимальный разрез

- С какой вероятностью алгоритм находит минимальный разрез?

Вероятность нахождения минимального разреза

Пусть в минимальном разрезе k ребер,
тогда в графе G ребер по крайней мере ...

Вероятность нахождения минимального разреза

Пусть в минимальном разрезе k ребер,
тогда в графе G ребер по крайней мере ...

- $nk/2$.

Зафиксируем минимальный разрез C. Оценим вероятность того, что в ходе работы алгоритма ни одного ребро из C не удалялось.

Вероятность нахождения минимального разреза

Пусть в минимальном разрезе k ребер,
тогда в графе G ребер по крайней мере ...

- $nk/2$.

Зафиксируем минимальный разрез C. Оценим вероятность того, что в ходе работы алгоритма ни одного ребро из C не удалялось.

- Событие e_i: ребро из C не было выбрано на i-ом шаге, $1 \leqslant i \leqslant (n-2)$.

Вероятность нахождения минимального разреза

Пусть в минимальном разрезе k ребер,
тогда в графе G ребер по крайней мере ...

- $nk/2$.

Зафиксируем минимальный разрез C. Оценим вероятность того, что в ходе работы алгоритма ни одного ребро из C не удалялось.

- Событие e_i: ребро из C не было выбрано на i-ом шаге, $1 \leqslant i \leqslant (n-2)$.
- 1-й шаг: $P(\neg e_1) \leqslant \frac{k}{\frac{(nk)}{2}} = \frac{2}{n}$.

  Следовательно, $P(e_1) \geqslant 1 - \frac{2}{n}$.

Вероятность нахождения минимального разреза

Пусть в минимальном разрезе k ребер,
тогда в графе G ребер по крайней мере ...

- $nk/2$.

Зафиксируем минимальный разрез C. Оценим вероятность того, что в ходе работы алгоритма ни одного ребро из C не удалялось.

- Событие e_i: ребро из C не было выбрано на i-ом шаге, $1 \leqslant i \leqslant (n-2)$.
- 1-й шаг: $P(\neg e_1) \leqslant \frac{k}{\frac{(nk)}{2}} = \frac{2}{n}$.

  Следовательно, $P(e_1) \geqslant 1 - \frac{2}{n}$.
- 2-й шаг: $P(e_2|e_1) \geqslant 1 - \frac{k}{\frac{k(n-1)}{2}} = 1 - \frac{2}{n-1}$.

Вероятность нахождения минимального разреза

Пусть в минимальном разрезе k ребер,
тогда в графе G ребер по крайней мере ...

- $nk/2$.

Зафиксируем минимальный разрез C. Оценим вероятность того, что в ходе работы алгоритма ни одного ребро из C не удалялось.

- Событие e_i: ребро из C не было выбрано на i-ом шаге, $1 \leqslant i \leqslant (n-2)$.
- 1-й шаг: $P(\neg e_1) \leqslant \frac{k}{\frac{(nk)}{2}} = \frac{2}{n}$.

  Следовательно, $P(e_1) \geqslant 1 - \frac{2}{n}$.
- 2-й шаг: $P(e_2|e_1) \geqslant 1 - \frac{k}{\frac{k(n-1)}{2}} = 1 - \frac{2}{n-1}$.
- i-й шаг: $P(e_i|\&_{j=1}^{i-1} e_j) \geqslant 1 - \frac{2}{(n-i+1)}$.

Вероятность нахождения минимального разреза

Вероятность того, что в ходе алгоритма ни одно ребро из C не удалялось (алгоритм на выходе даст в точности C):

$$P(\&_{i=1}^{n-2} e_i) \geqslant \prod_{i=1}^{n-2} (1 - \frac{2}{(n-i+1)}) = \frac{2}{n(n-1)}$$

Вероятность нахождения минимального разреза

Вероятность того, что в ходе алгоритма ни одно ребро из C не удалялось (алгоритм на выходе даст в точности C):

$$P(\&_{i=1}^{n-2} e_i) \geqslant \prod_{i=1}^{n-2} (1 - \frac{2}{(n-i+1)}) = \frac{2}{n(n-1)}$$

Probability boosting.
Запускаем алгоритм $n^2/2$ раз. Оценка на вероятность несрабатывания:

$$\left(1 - \frac{2}{n(n-1)}\right)^{n^2/2} < \left(1 - \frac{2}{n^2}\right)^{n^2/2} < 1/e.$$

Вероятность нахождения минимального разреза

Вероятность того, что в ходе алгоритма ни одно ребро из C не удалялось (алгоритм на выходе даст в точности C):

$$P(\&_{i=1}^{n-2} e_i) \geqslant \prod_{i=1}^{n-2} (1 - \frac{2}{(n-i+1)}) = \frac{2}{n(n-1)}$$

Probability boosting.
Запускаем алгоритм $n^2/2$ раз. Оценка на вероятность несрабатывания:

$$\left(1 - \frac{2}{n(n-1)}\right)^{n^2/2} < \left(1 - \frac{2}{n^2}\right)^{n^2/2} < 1/e.$$

Запускаем алгоритм n^2 раз, вероятность несрабатывания менее $1/e^2$

Задача максимальная выполнимость (MAX-SAT)

- Дано: набор дизъюнкций C на множестве булевых переменных $x_1, \ldots, x_n$, веса дизъюнкций $\omega_c \geqslant 0$, $c \in C$
- Найти: назначение булевых переменных с максимальным суммарным весом выполненных дизъюнкций

Обозначения:
$size(c)$ — число литералов (переменных или их отрицаний), входящих в дизъюнкцию c, размер каждой дизъюнкции произвольный;
W_c — случайная величина, равная весу, вносимому дизъюнкцией c;
$W = \sum_{c \in C} W_c$ — суммарный вес выполненных дизъюнкций;
$E(W_c) = \omega_c P(c = 1)$ — ожидаемый вклад дизъюнкции c.

RA для MAX-SAT

Алгоритм Джонсона

for $i = 1$ *to* n *do*
begin
$p := Random(0, 1);$
if $p \leqslant 1/2$ *then* $x_i := true$
else $x_i := false;$
end;

RA для MAX-SAT

Замечание

Если $size(c) = k$, *то* $E(W_c) = \alpha_k \omega_c$, *где* $\alpha_k = 1 - \frac{1}{2^k}$, $k \geqslant 1$.

Для $k \geqslant 1$, $\alpha_k \geqslant \frac{1}{2}$, $E(W) = \sum_{c \in C} E(W_c) \geqslant \frac{1}{2} \sum_{c \in C} \omega_c \geqslant \frac{1}{2} OPT$
Алгоритм хорош при больших значениях k.

MAX-SAT в виде задачи ЦЛП

Пусть S_c^+ и (S_c^-) — множества переменных, входящих в дизъюнкцию c без отрицания и с отрицанием соответственно.

Переменные задачи:

$$y_i = \begin{cases} 1, \text{ если } x_i = true; \\ 0, \text{иначе.} \end{cases} \qquad z_c = \begin{cases} 1, \text{ если } c \text{ выполнена}, \\ 0, \text{ иначе.} \end{cases}$$

Математическая модель:

$$\max \sum_{c \in C} \omega_c z_c$$

$$\sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geqslant z_c, \forall c \in C;$$

$$z \in \{0, 1\} \quad \forall c \in C;$$

$$y_i \in \{0, 1\} \quad \forall i \in \{1, \ldots, n\}.$$

RA для MAX-SAT

Алгоритм вероятностного округления

1. *Найти* (y^*, z^*) *— оптимальное решение LP-релаксации*
2. *for* $i = 1$ *to* n *do*

```
begin
    p := Random(0, 1);
    if p ⩽ y_i^* then x_i := 1
    else x_i := 0;
end;
```

RA для MAX-SAT

Теорема 2

Если $size(c) = k$, *то* $E(W_c) = \beta_k \omega_c z_c^*$, *где* $\beta_k = 1 - (1 - \frac{1}{k})^k$, $k \geqslant 1$.

Доказательство:

Можно считать, что c *не содержит отрицаний переменных. Для вероятности того, что* c *выполнена, имеем:*

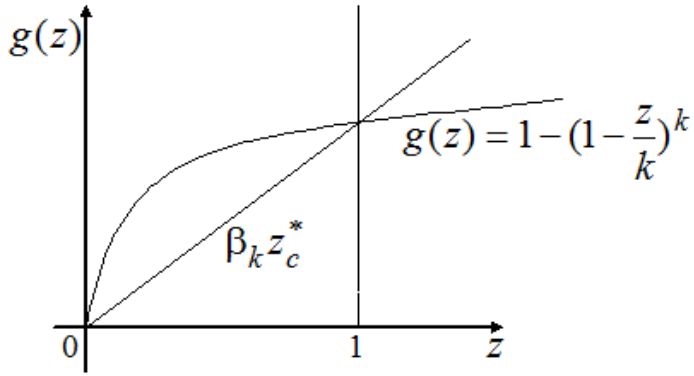
$$1 - \prod_{i=1}^{k}(1 - y_i^*) \geqslant^{(1)} 1 - \left(\frac{\sum_{i=1}^{k}(1 - y_i^*)}{k}\right)^k =$$

$$1 - \left(1 - \frac{\sum_{i=1}^{k} y_i^*}{k}\right)^k \geqslant^{(2)} 1 - \left(1 - \frac{z_c^*}{k}\right)^k$$

(1): $\frac{a_1 + \ldots + a_k}{k} \geqslant \sqrt[k]{a_1 \cdot \ldots \cdot a_k}$

(2): $y_1^* + \ldots + y_k^* \geqslant z_c^*$

Доказательство (продолжение):



$$g(z) \text{ — вогнутая, } g(0) = 0,\ g(1) = \beta_k$$

$$\Downarrow$$

$$\text{для } z \in [0,1],\ g(z) \geqslant \beta_k z; \quad P(c = true) = g(z_c^*) \geqslant \beta_k z_c^*.$$

$$E(W) = \sum_{c \in C} E(W_c) \geqslant \beta_k \sum_{c \in C} \omega_c z_c^* = \beta_k OPT_{LR} \geqslant \beta_k OPT,$$

где OPT_{LR} — оптимум LP-релаксации.
Заметим, $(1 - \frac{1}{k})^k > \frac{1}{e},\ \forall k \in \mathbb{Z}^+$.

Алгоритм Гойманса и Уильямсона с оценкой $3/4$

С равной вероятностью выполняется либо алгоритм Джонсона, либо алгоритм вероятностного округления.

Таким образом $x_i = 1$ с вероятностью $\frac{1}{4} + \frac{1}{2}y_i^*$

Теорема 3

$E(W_c) \geqslant \frac{3}{4}\omega_c z_c^*$

Доказательство:

Пусть $b = 0$, *если выполняется алгоритм Джонсона,*
$b = 1$, *если выполняется вероятностное округление,*
z^* — *оптимальное решение LP-релаксации,* $size(c) = k$.
по теореме 2: $E(W_c|b = 0) = \alpha_k\omega_c \geqslant \alpha_k\omega_c z_c^*$, $z_c^* \leqslant 1$.
по теореме 3: $E(W_c|b = 1) \geqslant \beta_k\omega_c z_c^*$.
$\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = 3/2$, *для* $k \geqslant 3$ $\alpha_k + \beta_k \geqslant 7/8 + (1 - \frac{1}{e}) \geqslant \frac{3}{2}$.
$E(W_c) = \frac{1}{2}(E(W_c|b = 0) + E(W_c|b = 1)) \geqslant \omega_c z_c^* \frac{\alpha_k+\beta_k}{2}$
$E(W) = \sum_{c\in C} E(W_c) \geqslant \frac{3}{4}\sum_{c\in C} \omega_c z_c^* = \frac{3}{4}OPT_f \geqslant \frac{3}{4}OPT$.

Пример. Оценка $3/4$ точна для релаксации

- Дано: $C = \{(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2})\}$, $\omega_c = 1$
- Оптимальное решение LP-релаксации:

Пример. Оценка $3/4$ точна для релаксации

- Дано: $C = \{(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2})\}$, $\omega_c = 1$
- Оптимальное решение LP-релаксации:
- $y_i = \frac{1}{2}, \forall i,\ z_c = 1, \forall c,\ OPT_f = 4$

Пример. Оценка $3/4$ точна для релаксации

- Дано: $C = \{(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2})\}$, $\omega_c = 1$
- Оптимальное решение LP-релаксации:
- $y_i = \frac{1}{2}, \forall i,\ z_c = 1, \forall c,\ OPT_f = 4$
- Оптимальное решение:

Пример. Оценка $3/4$ точна для релаксации

- Дано: $C = \{(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2})\}$, $\omega_c = 1$
- Оптимальное решение LP-релаксации:
- $y_i = \frac{1}{2}, \forall i,\ z_c = 1, \forall c,\ OPT_f = 4$
- Оптимальное решение:
- $OPT = 3$.

Пример. Оценка $3/4$ точна для алгоритма Гойманса и Уильямсона

- Дано: $C = \{(x_1 \vee x_2), (\overline{x_1} \vee x_3), (x_1 \vee \overline{x_2})\}$, $\omega_1 = 1$, $\omega_2 = 1$, $\omega_3 = 2 + \varepsilon$
- Убедитесь, что оценка точна для алгоритма Гойманса и Уильямсона.

Дерандомизация методом условных мат. ожиданий

Общая схема дерандомизации

for $i = 1$ *to* n *do*

вычислить $E_1 = E(Z|X_1 = \beta_1, \ldots, X_{i-1} = \beta_{i-1}, X_i = 1)$

вычислить $E_0 = E(Z|X_1 = \beta_1, \ldots, X_{i-1} = \beta_{i-1}, X_i = 0)$

if $E_1 \geqslant E_0$ *then* $\beta_i := 1$

else $\beta_i := 0$

$\beta = \beta_1\beta_2 \ldots \beta_n$ — *ответ*

Нетрудно заметить, что $E(Z|X_1, X_2, \ldots, X_{i-1}) \leqslant E(Z|X_1, X_2, \ldots, X_i)$

Этот алгоритм полиномиальный, если условные мат. ожидания вычислимы за полиномиальное время.

Рекомендуемая литература

- Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы построение и анализ, 2-е издание. М.: Изд. дом «Вильямс», 2009
- V. Vazirani Approximation Algorithms. Springer-Verlag Berlin 2001.
- Juraj Hromkovi$\acute{c}$ Algorithmics for Hard Problems. Introduction to combi-natorial optimization, randomization, approximation, and heuristics. Second edition, Springer-Verlag Berlin 2001, 2003
- R. Motwani, P. Raghavan Randomized Algorithms. Cambridge University Press, 1995
- R. C. T. Lee, S. S. Tseng, R. C. Chang, Y. T. Tsai Introduction to the Design and Analysis of Algorithms. A strategic Approach. MCGraw-Hill, 2005