

## Лекция 5. Рандомизированные алгоритмы

Екатерина Вячеславовна Алексеева

Новосибирский Государственный Университет  
Факультет Информационных Технологий  
<http://math.nsc.ru/~alekseeva/>

7 апреля, 2012 г.

## Содержание лекции

Классификация рандомизированных алгоритмов

Примеры рандомизированных алгоритмов

RA для задачи о максимальной выполнимости

Дерандомизированный RA для задачи MAX SAT

Рекомендуемая литература

## Определения

### Рандомизированный алгоритм (RA)

- ▶ *алгоритм, в котором некоторые действия основаны на случайном выборе;*
- ▶ *множество детерминированных алгоритмов, один из которых выбирается вероятностным способом, зависящим от заданного входа.*



## Алгоритм типа Лас-Вегас

если для любого входа  $x$  задачи  $F$  алгоритм  $A$  находит решение с вероятностью единица. При этом время работы алгоритма является случайной величиной.



## Алгоритм типа Монте-Карло

если для любого входа  $x$  задачи  $F$  алгоритм  $A$  находит решение с некоторой ненулевой вероятностью.

## Пример алгоритма типа Лас-Вегас

### Алгоритм быстрая сортировка $QS(S)$

- ▶ *Вход:* множество  $S$  из  $n$  элементов.
  - ▶ *Выход:* множество элементов из  $S$  отсортированных по неубыванию.
1. *Выбрать элемент  $y$  случайным образом с равномерным распределением из  $n$  элементов.*
  2. *Если  $n = 1$ , то выдать  $S$  в качестве ответа, иначе разбить множество  $S$  на три подмножества:*  
 $S_{<} := \{\text{элементы из } S \text{ меньше, чем } y\},$   
 $S_{=} := \{\text{элементы из } S \text{ равные } y\},$   
 $S_{>} := \{\text{элементы из } S \text{ больше, чем } y\}$
  3. *Вызвать алгоритм  $QS(S_{<})$  и  $QS(S_{>})$ .*

## Сложность алгоритма $QS(S)$

### Теорема 1

Ожидаемое число сравнений в алгоритме  $QS(S)$  не более  $2nH_n$ .

### Доказательство:

Пусть  $s_1 < s_2 < \dots < s_n$  — результат работы алгоритма  $QS(S)$ .

Пусть  $x_{ij}$  — случайная величина:

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

$i, j \in \{1, \dots, n\}, i < j$ .  $T = \sum_{i=1}^n \sum_{j>i} x_{ij}$  — общее число сравнений.

Тогда ожидаемая сложность алгоритма:

$$E[T] = E\left[\sum_{i=1}^n \sum_{j>i} x_{ij}\right] = \sum_{i=1}^n \sum_{j>i} E[x_{ij}]$$

## Доказательство (продолжение):

Пусть  $p_{ij}$  — вероятность того, что  $s_i$  и  $s_j$  сравниваются.

$$E[x_{ij}] = p_{ij} \cdot 1 + (1 - p_{ij}) \cdot 0 = p_{ij}.$$

Рассмотрим для каждого  $i, j \in \{1, \dots, n\}, i < j$  последовательность  $s_i, s_{i+1}, \dots, s_{i+j-1}, s_j$ .

Элементы  $s_i$  и  $s_j$  сравниваются между собой только, если один из них был выбран в качестве первого элемента последовательности. Любой из элементов  $s_i, s_{i+1}, \dots, s_{i+j-1}, s_j$  с равной вероятностью может быть первым в последовательности, следовательно,

$$p_{ij} = \frac{2}{j-i+1}.$$

$$\begin{aligned} E[T] &= \sum_{i=1}^n \sum_{j>i}^n p_{ij} = \sum_{i=1}^n \sum_{j>i}^n \frac{2}{j-i+1} \leq \sum_{i=1}^n \sum_{k=1}^{n-i+1} \frac{2}{k} \\ &\leq 2 \sum_{i=1}^n \sum_{k=1}^n \frac{1}{k} = 2 \sum_{i=1}^n H(n) = 2nH(n) \approx 2n \ln n + \Theta(1) = O(n \log n). \end{aligned}$$

## 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

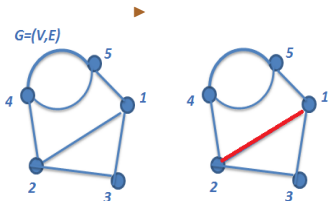
### Алгоритм стягивания вершин

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

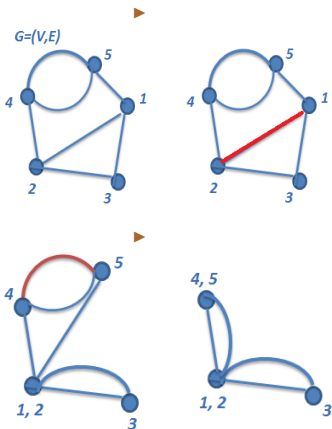
*Пока не останется две вершины выполнить:*

- 1. выбрать случайным образом с равномерным распределением ребро  $e = (u, v)$*
- 2. слить вершины  $u$  и  $v$  в одну "метавершину":  
удалить все ребра между вершинами  $u$  и  $v$ ,  
все ребра из  $E$  инцидентные вершинам  $u$  и  $v$  инцидентны "метавершине"*

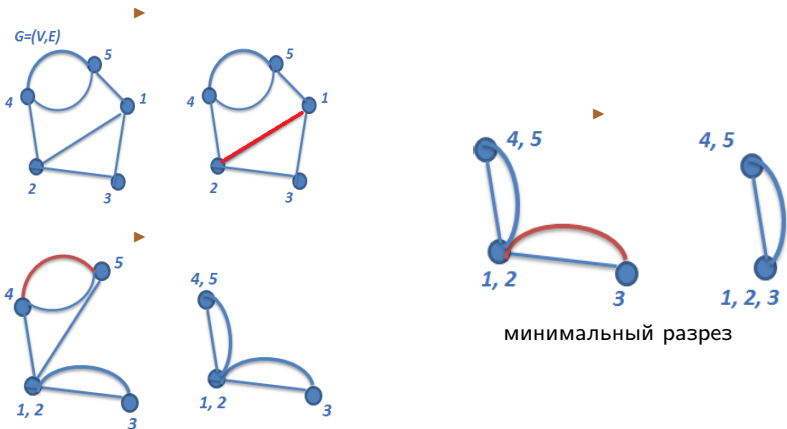
## Рандомизированный алгоритм для задачи о минимальном разрезе



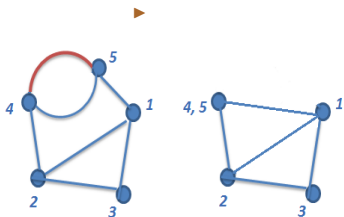
## Рандомизированный алгоритм для задачи о минимальном разрезе



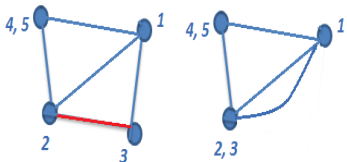
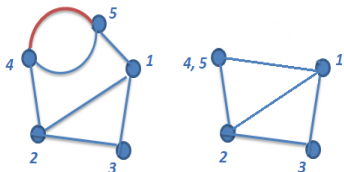
## Рандомизированный алгоритм для задачи о минимальном разрезе



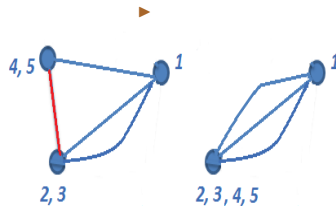
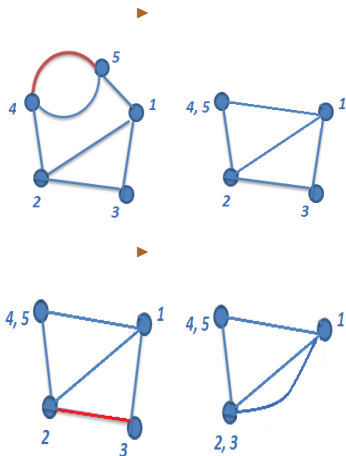
## Рандомизированный алгоритм для задачи о минимальном разрезе



## Рандомизированный алгоритм для задачи о минимальном разрезе

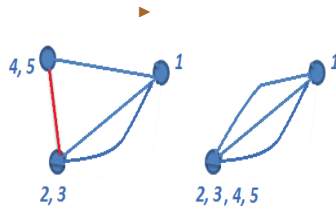
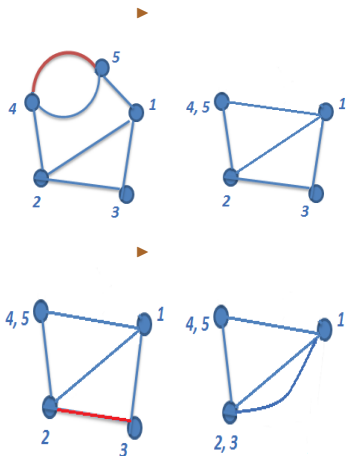


## Рандомизированный алгоритм для задачи о минимальном разрезе



неминимальный разрез

## Рандомизированный алгоритм для задачи о минимальном разрезе



- нелиминальный 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 ...

►  $nk/2$ .

Оценим вероятность того, что в ходе работы алгоритма ни одного ребро  
из разреза не удалялось.

## Вероятность нахождения минимального разреза

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

- ▶  $nk/2$ .

Оценим вероятность того, что в ходе работы алгоритма ни одного ребро  
из разреза не удалялось.

- ▶ Пусть  $e_i$  означает, что ребро из разреза не было выбрано алгоритмом  
на  $i$ -ом шаге,  $1 \leq i \leq (n - 2)$ .

## Вероятность нахождения минимального разреза

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

- ▶  $nk/2$ .

Оценим вероятность того, что в ходе работы алгоритма ни одного ребро  
из разреза не удалялось.

- ▶ Пусть  $e_i$  означает, что ребро из разреза не было выбрано алгоритмом на  $i$ -ом шаге,  $1 \leq i \leq (n-2)$ .
- ▶ 1-й шаг:  $P(\text{ребро выбранное на первом шаге} \in C) \geq \frac{k}{\frac{(nk)}{2}} = \frac{2}{n}$ .

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

## Вероятность нахождения минимального разреза

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

- ▶  $nk/2$ .

Оценим вероятность того, что в ходе работы алгоритма ни одного ребро из разреза не удалялось.

- ▶ Пусть  $e_i$  означает, что ребро из разреза не было выбрано алгоритмом на  $i$ -ом шаге,  $1 \leq i \leq (n-2)$ .
- ▶ 1-й шаг:  $P(\text{ребро выбранное на первом шаге} \in C) \geq \frac{k}{\frac{(nk)}{2}} = \frac{2}{n}$ .

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

- ▶ 2-й шаг: число вершин уменьшилось на одну, а ребер не меньше, чем  $k(n-1)/2$ , следовательно,  $P(e_2|e_1) \geq 1 - \frac{2}{(n-1)}$ .

## Вероятность нахождения минимального разреза

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

- ▶  $nk/2$ .

Оценим вероятность того, что в ходе работы алгоритма ни одного ребро из разреза не удалялось.

- ▶ Пусть  $e_i$  означает, что ребро из разреза не было выбрано алгоритмом на  $i$ -ом шаге,  $1 \leq i \leq (n-2)$ .
- ▶ 1-й шаг:  $P(\text{ребро выбранное на первом шаге} \in C) \geq \frac{k}{\frac{(nk)}{2}} = \frac{2}{n}$ .

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

- ▶ 2-й шаг: число вершин уменьшилось на одну, а ребер не меньше, чем  $k(n-1)/2$ , следовательно,  $P(e_2|e_1) \geq 1 - \frac{2}{(n-1)}$ .
- ▶  $i$ -й шаг: ребер не меньше, чем  $k(n-i+1)/2$ , следовательно,  $P(e_i | \cap_{j=1}^{i-1} e_j) \geq 1 - \frac{2}{(n-i+1)}$ .

## Вероятность нахождения минимального разреза

Итого, вероятность того, что в ходе алгоритма ни одно ребро из разреза не удалялось:

$$P(\cap_{i=1}^{n-2} e_i) \geq \prod_{i=1}^{n-2} (1 - \frac{2}{(n-i+1)}) = \frac{2}{n(n-1)}$$

Сколько раз следует запустить алгоритм, чтобы вероятность несрабатывания была как можно меньше?

## Вероятность нахождения минимального разреза

Итого, вероятность того, что в ходе алгоритма ни одно ребро из разреза не удалялось:

$$P(\cap_{i=1}^{n-2} e_i) \geq \prod_{i=1}^{n-2} (1 - \frac{2}{(n-i+1)}) = \frac{2}{n(n-1)}$$

Сколько раз следует запустить алгоритм, чтобы вероятность несрабатывания была как можно меньше?

- ▶ Пусть запустили алгоритм  $n^2/2$  раз независимо с равномерным распределением.  
Вероятность того, что за  $n^2/2$  раз минимальный разрез не найден не больше, чем

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



## Пример алгоритма типа Монте-Карло

### Алгоритм

1. Запустить алгоритм с входом  $x$   $t$  раз. Пусть  $y_1, \dots, y_t$  — полученные решения.
2. Предъявить в качестве ответа решение  $y$ , полученное по крайней мере  $\lceil \frac{t}{2} \rceil$  раз среди  $t$  запусков, если такого  $y$  нет, то задача не решена.

## Пример алгоритма типа Монте-Карло

Пусть  $p(x)$  — вероятность того, что алгоритм  $A$  находит правильный ответ на заданном входе  $x$  за один запуск:

$$p = p(x) \geq \frac{1}{2} + \epsilon, \quad 0 < \epsilon \leq \frac{1}{2}$$

Вероятность, что алгоритм  $A$  находит правильный ответ на заданном входе  $x$   $i$  раз за  $t$  запусков:

$$\begin{aligned} p_i(x) &= C_i^t p^i (1-p)^{t-i} = \\ C_i^t (p(1-p))^i (1-p)^{2(\frac{t}{2}-i)} &\leq \\ C_i^t \left(\frac{1}{4} - \epsilon^2\right)^i \left(\frac{1}{4} - \epsilon^2\right)^{(\frac{t}{2}-i)} &= \\ C_i^t \left(\frac{1}{4} - \epsilon^2\right)^{\frac{1}{2}} \end{aligned}$$

## Пример алгоритма типа Монте-Карло

Вероятность, что алгоритм  $A_t$  решает задачу:

$$\begin{aligned} P &\geq (1 - \sum_{i=0}^{\lfloor \frac{t}{2} \rfloor} p_i(x)) > (1 - \sum_{i=0}^{\lfloor \frac{t}{2} \rfloor} C_i^t (\frac{1}{4} - \varepsilon^2)^{\frac{t}{2}})) \\ &> (1 - 2^{t-1} (\frac{1}{4} - \varepsilon^2)^{\frac{t}{2}}) = (1 - \frac{1}{2} (\frac{1}{4} - \varepsilon^2)^{\frac{t}{2}}) \end{aligned}$$

Чтобы найти  $k$  такое, что  $P(A_k(x) \text{ решает задачу}) \geq 1 - \delta$  для выбранного  $\delta$  и любого входа  $x$  достаточно взять  $k \geq \frac{2 \log_2 2\delta}{\log_2(1-4\varepsilon^2)}$

## Задача максимальная выполнимость (MAX-SAT)

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

Обозначения:

$size(c)$  — число литералов, входящих в дизъюнкцию  $c$ , размер каждой дизъюнкции произвольный;

$W_c$  — случайная переменная, обозначающая вес, вносимый дизъюнкцией  $c$ ;

$W = \bigcup_{c \in C} W_c$  — суммарный вес выполненных дизъюнкций;

$E(W_c) = \omega_c P(c = 1)$  — мат. ожидание выполнимости дизъюнкции  $c$ .

## RA для MAX-SAT

### Алгоритм Джонсона

```
for  $i = 1$  to  $n$  do  
begin  
     $p := \text{Random}(0, 1)$ ;  
    if  $p \leq 1/2$  then  $x_i := 1$   
    else  $x_i := 0$ ;  
end;
```

## RA для MAX-SAT

### Теорема 2

Если  $size(c) = k$ , то ожидаемый вес, вносимый дизъюнкцией  $c$  оценивается  $E(W_c) = \alpha_k \omega_c$ , где  $\alpha_k = 1 - \frac{1}{2^k}$ ,  $k \geq 1$ .

### Доказательство:

$c$  — не выполнима  $\Leftrightarrow$  все литералы в  $c$  принимают значение 0, вероятность этого события  $\frac{1}{2^k}$ .

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

## MAX-SAT в виде задачи ЦЛП

Пусть  $S_c^+$ ,  $(S_c^-)$  — множество переменных, входящих в дизъюнкцию  $c$  без(с) отрицания(ем).

Переменные задачи:

$$y_i = \begin{cases} 1, & \text{если } x_i = 1, \\ 0, & \text{если } x_i = 0 \end{cases} \quad z_c = \begin{cases} 1, & \text{если } c \text{ выполнима,} \\ 0 & \text{иначе} \end{cases}$$

Математическая модель:

$$\begin{aligned} & \max \sum_{c \in C} \omega_c z_c \\ & \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c, \forall c \in C \\ & z \in \{0, 1\} \forall c \in C; y_i \in \{0, 1\} \forall i \in \{1, \dots, n\} \end{aligned}$$

## LP-релаксация MAX-SAT

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

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

$$0 \leq z_c \leq 1, \forall c \in C$$

$$0 \leq y_i \leq 1, \forall i \in \{1, \dots, n\}$$



## RA для MAX-SAT

### Вероятностный алгоритм Джонсона

1. Найти  $(y^*, z^*)$  — оптимальное решение LP-релаксации
2. for  $i = 1$  to  $n$  do  
    begin  
         $p := \text{Random}(0, 1)$ ;  
        if  $p \leq y_i^*$  then  $x_i := 1$   
        else  $x_i := 0$ ;  
    end;

## RA для MAX-SAT

### Теорема 3

Если  $size(c) = k$ , то ожидаемый вес, вносимый дизъюнкцией  $c$  оценивается  $E(W_c) = \beta_k \omega_c z_c^*$ , где  $\beta_k = 1 - (1 - \frac{1}{k})^k$ ,  $k \geq 1$ .

### Доказательство:

Можно считать, что все переменные входят в дизъюнкции без отрицания, иначе сделать замену. Вероятность того, что дизъюнкция  $c$  выполнена:

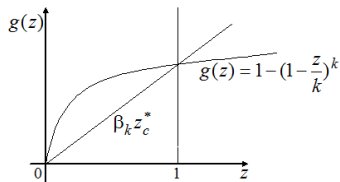
$$1 - \prod_{i=1}^k (1 - y_i) \geq^{(1)} 1 - \left( \frac{\sum_{i=1}^k (1 - y_i)}{k} \right)^k =$$

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

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

$$(2): y_1 + \dots + y_k \geq z_c$$

## Доказательство (продолжение):



$g(z)$  — вогнутая,  $g(0) = 0$ ,  $g(1) = \beta_k$ , следовательно,  $z \in [0, 1]$ ,  $g(z) \geq \beta_k z$ , следовательно,  $P(\text{"с выполнена"}) \geq \beta_k z_c^*$ .

$\beta_k$  — убывающая функция по  $k$ ;

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

где  $OPT_f$  — оптимум LP-релаксации и  $OPT_f \geq OPT$ .

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 вероятностный алгоритм, основанный на LP-релаксации.*

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

### Теорема 4

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

### Доказательство:

Пусть  $b = 0$ , если выполняется алгоритм Джонсона,  
 $b = 1$ , если выполняется вероятностный алгоритм Джонсона,  
 $z^*$  — оптимальное решение LP-релаксации,  $size(c) = k$ .

по теореме 2:  $E(W_c|b=0) = \alpha_k \omega_c \geq \alpha_k \omega_c z_c^*, z_c^* \leq 1$ .

по теореме 3:  $E(W_c|b=1) \geq \beta_k \omega_c z_c^*$ .

$$\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = 3/2, \text{ для } k \geq 3 \quad \alpha_k + \beta_k \geq 7/8 + (1 - \frac{1}{e}) \geq \frac{3}{2}.$$

$$E(W_c) = \frac{1}{2}(E(W_c|b=0) + E(W_c|b=1)) \geq \omega_c z_c^* \frac{\alpha_k + \beta_k}{2}$$

$$E(W) = \sum_{c \in C} E(W_c) \geq \frac{3}{4} \sum_{c \in C} \omega_c z_c^* = \frac{3}{4} OPT_f \geq \frac{3}{4} OPT.$$

## Пример, что оценка $3/4$ точная для LP

- ▶ 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, что оценка $3/4$ точная для LP

- ▶ 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, что оценка $3/4$ точная для LP

- ▶ 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, что оценка $3/4$ точная для LP

- ▶ 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, что оценка $3/4$ точная 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 — конвертирование рандомизированного алгоритма в детерминированный.

Например, методом "условных вероятностей".

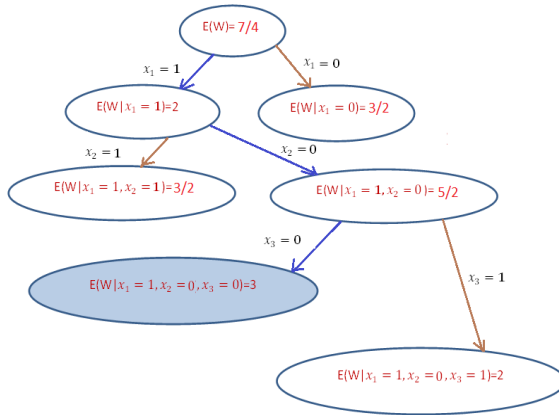


Рис.: Дерево назначений в задаче MAX SAT

## Дерандомизация

### Общая схема дерандомизации

- ▶ *Вход: постановка задачи в виде ЦЛП.*
- ▶ *Выход:  $\beta$  —приближенное решение (назначение переменных).*

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

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

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

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






*else  $\beta_i := 0$*

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

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

Этот алгоритм полиномиальный, если условные вероятности 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 Hromkovič 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