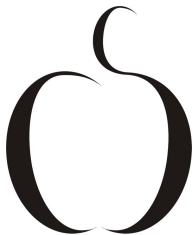


с/к “Эффективные алгоритмы”

Лекция 8: Приближенные алгоритмы

А. Куликов

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



План лекции

- 1 Задача о взвешенном вершинном покрытии

План лекции

- 1 Задача о взвешенном вершинном покрытии
- 2 Задача о минимальном множестве представителей

План лекции

- 1 Задача о взвешенном вершинном покрытии
- 2 Задача о минимальном множестве представителей

Задача о вершинном покрытии

Определение

Задача о вершинном покрытии (vertex cover problem) заключается в нахождении по данному графу такого минимального множества его вершин, что для любого ребра графа хотя бы один из его концов принадлежит этому множеству.

Задача о вершинном покрытии

Определение

Задача о вершинном покрытии (vertex cover problem) заключается в нахождении по данному графу такого минимального множества его вершин, что для любого ребра графа хотя бы один из его концов принадлежит этому множеству.

Задача о вершинном покрытии

Определение

Задача о вершинном покрытии (vertex cover problem) заключается в нахождении по данному графу такого минимального множества его вершин, что для любого ребра графа хотя бы один из его концов принадлежит этому множеству.

- Мы уже знаем очень простой детерминированный 2-приближенный алгоритм для задачи о вершинном покрытии.

Задача о вершинном покрытии

Определение

Задача о вершинном покрытии (vertex cover problem) заключается в нахождении по данному графу такого минимального множества его вершин, что для любого ребра графа хотя бы один из его концов принадлежит этому множеству.

- Мы уже знаем очень простой детерминированный 2-приближенный алгоритм для задачи о вершинном покрытии.
- Сейчас мы приведем **вероятностный** 2-приближенный алгоритм.

Задача о вершинном покрытии

Определение

Задача о вершинном покрытии (vertex cover problem) заключается в нахождении по данному графу такого минимального множества его вершин, что для любого ребра графа хотя бы один из его концов принадлежит этому множеству.

- Мы уже знаем очень простой детерминированный 2-приближенный алгоритм для задачи о вершинном покрытии.
- Сейчас мы приведем **вероятностный** 2-приближенный алгоритм.
- Говоря, что вероятностный алгоритм является 2-приближенным, мы имеем в виду, что математическое ожидание выдаваемого им результата не более чем в два раза хуже оптимального.

Алгоритм

Алгоритм

RAND-APPROX-VERTEX-COVER(G)

Алгоритм

Алгоритм

RAND-APPROX-VERTEX-COVER(G)

- $C = \emptyset$

Алгоритм

Алгоритм

RAND-APPROX-VERTEX-COVER(G)

- $C = \emptyset$
- пока есть непокрытое ребро, выбрать случайно один из его концов и добавить его к множеству C

Алгоритм

Алгоритм

RAND-APPROX-VERTEX-COVER(G)

- $C = \emptyset$
- пока есть непокрытое ребро, выбрать случайно один из его концов и добавить его к множеству C
- вернуть C

Алгоритм

Алгоритм

RAND-APPROX-VERTEX-COVER(G)

- $C = \emptyset$
- пока есть непокрытое ребро, выбрать случайно один из его концов и добавить его к множеству C
- вернуть C

Лемма

Алгоритм RAND-APPROX-VERTEX-COVER является 2-приближенным.

Доказательство

Доказательство

Доказательство

Доказательство

- Пусть C_{opt} — минимальное вершинное покрытие.

Доказательство

Доказательство

- Пусть C_{opt} — минимальное вершинное покрытие.
- Хотя бы один конец каждого ребра, рассмотренного алгоритмом, содержится в C_{opt} .

Доказательство

Доказательство

- Пусть C_{opt} — минимальное вершинное покрытие.
- Хотя бы один конец каждого ребра, рассмотренного алгоритмом, содержится в C_{opt} .
- Значит, на каждом шаге алгоритм добавляет в C вершину из C_{opt} с вероятностью хотя бы $1/2$.

Доказательство

Доказательство

- Пусть C_{opt} — минимальное вершинное покрытие.
- Хотя бы один конец каждого ребра, рассмотренного алгоритмом, содержится в C_{opt} .
- Значит, на каждом шаге алгоритм добавляет в C вершину из C_{opt} с вероятностью хотя бы $1/2$.
- Назовем такой шаг **успешным**.

Доказательство

Доказательство

- Пусть C_{opt} — минимальное вершинное покрытие.
- Хотя бы один конец каждого ребра, рассмотренного алгоритмом, содержится в C_{opt} .
- Значит, на каждом шаге алгоритм добавляет в C вершину из C_{opt} с вероятностью хотя бы $1/2$.
- Назовем такой шаг **успешным**.
- Ясно, что алгоритм может сделать не более $|C_{opt}|$ успешных шагов.

Доказательство

Доказательство

- Пусть C_{opt} — минимальное вершинное покрытие.
- Хотя бы один конец каждого ребра, рассмотренного алгоритмом, содержится в C_{opt} .
- Значит, на каждом шаге алгоритм добавляет в C вершину из C_{opt} с вероятностью хотя бы $1/2$.
- Назовем такой шаг **успешным**.
- Ясно, что алгоритм может сделать не более $|C_{opt}|$ успешных шагов.
- Значит, мат. ожидание общего количества шагов (а это есть $|C|$) не превосходит $2|C_{opt}|$.



Задача о взвешенном вершинном покрытии

Определение

Задача о взвешенном вершинном покрытии

Определение

- Дан граф $G = (V, E)$, в котором каждой вершине v присвоен некоторый вес $w(v)$.

Задача о взвешенном вершинном покрытии

Определение

- Дан граф $G = (V, E)$, в котором каждой вершине v присвоен некоторый вес $w(v)$.
- **Задача о взвешенном вершинном покрытии** заключается в нахождении вершинного покрытия минимального веса.

Задача о взвешенном вершинном покрытии

Определение

- Дан граф $G = (V, E)$, в котором каждой вершине v присвоен некоторый вес $w(v)$.
- **Задача о взвешенном вершинном покрытии** заключается в нахождении вершинного покрытия минимального веса.

Замечание

Задача о взвешенном вершинном покрытии

Определение

- Дан граф $G = (V, E)$, в котором каждой вершине v присвоен некоторый вес $w(v)$.
- **Задача о взвешенном вершинном покрытии** заключается в нахождении вершинного покрытия минимального веса.

Замечание

- Можно построить 2-приближенный алгоритм для задачи о взвешенном вершинном покрытии, модифицировав предыдущий алгоритм следующим образом: если ребро (u, v) еще не покрыто, то с вероятностью $\frac{1}{w(u)} \left(\frac{1}{w(u)} + \frac{1}{w(v)} \right)^{-1}$ добавляем в покрытие вершину u , а с вероятностью $\frac{1}{w(v)} \left(\frac{1}{w(u)} + \frac{1}{w(v)} \right)^{-1}$ — вершину v .

Формулировка в виде задачи целочисленного линейного программирования

Формулировка в виде задачи целочисленного линейного программирования

$$\min \sum_{v \in V} w(v)x(v)$$

$$x(v_1) + x(v_2) \geq 1, (v_1, v_2) \in E$$

$$x(v) \in \{0, 1\}, v \in V$$

Алгоритм

Алгоритм

Алгоритм

Алгоритм

- Заменяем условие $x(v) \in \{0, 1\}$ на $x(v) \geq 0$ и решим полученную задачу.

Алгоритм

Алгоритм

- Заменяем условие $x(v) \in \{0, 1\}$ на $x(v) \geq 0$ и решим полученную задачу.
- Пусть $\{\hat{x}(v)\}_{v \in V}$ — найденное решение.

Алгоритм

Алгоритм

- Заменяем условие $x(v) \in \{0, 1\}$ на $x(v) \geq 0$ и решим полученную задачу.
- Пусть $\{\hat{x}(v)\}_{v \in V}$ — найденное решение.
- Зададим тогда такой набор:

$$x(v) = \begin{cases} 1, & \text{если } \hat{x}(v) \geq 1/2 \\ 0, & \text{в противном случае} \end{cases}$$

Алгоритм

Алгоритм

- Заменяем условие $x(v) \in \{0, 1\}$ на $x(v) \geq 0$ и решим полученную задачу.
- Пусть $\{\hat{x}(v)\}_{v \in V}$ — найденное решение.
- Зададим тогда такой набор:

$$x(v) = \begin{cases} 1, & \text{если } \hat{x}(v) \geq 1/2 \\ 0, & \text{в противном случае} \end{cases}$$

- Положим $C = \{v \in V \mid x(v) = 1\}$.

Алгоритм

Алгоритм

- Заменяем условие $x(v) \in \{0, 1\}$ на $x(v) \geq 0$ и решим полученную задачу.
- Пусть $\{\hat{x}(v)\}_{v \in V}$ — найденное решение.
- Зададим тогда такой набор:

$$x(v) = \begin{cases} 1, & \text{если } \hat{x}(v) \geq 1/2 \\ 0, & \text{в противном случае} \end{cases}$$

- Положим $C = \{v \in V \mid x(v) = 1\}$.
- Найденное решение является допустимым (то есть покрытием), поскольку для каждого ребра (u, v) выполнено неравенство $\hat{x}(u) + \hat{x}(v) \geq 1$, а значит, $\hat{x}(u) \geq 1/2$ или $\hat{x}(v) \geq 1/2$.

Алгоритм

Алгоритм

- Заменяем условие $x(v) \in \{0, 1\}$ на $x(v) \geq 0$ и решим полученную задачу.
- Пусть $\{\hat{x}(v)\}_{v \in V}$ — найденное решение.
- Зададим тогда такой набор:

$$x(v) = \begin{cases} 1, & \text{если } \hat{x}(v) \geq 1/2 \\ 0, & \text{в противном случае} \end{cases}$$

- Положим $C = \{v \in V \mid x(v) = 1\}$.
- Найденное решение является допустимым (то есть покрытием), поскольку для каждого ребра (u, v) выполнено неравенство $\hat{x}(u) + \hat{x}(v) \geq 1$, а значит, $\hat{x}(u) \geq 1/2$ или $\hat{x}(v) \geq 1/2$.
- $w(C) = \sum_{v \in V} w(v)x(v) \leq 2 \sum_{v \in V} w(v)\hat{x}(v) \leq 2w(C_{\text{opt}})$.

Еще один алгоритм

Алгоритм

WEIGHTED-VERTEX-COVER-1(G, w)

Еще один алгоритм

Алгоритм

WEIGHTED-VERTEX-COVER-1(G, w)

- $C = \emptyset$

Еще один алгоритм

Алгоритм

WEIGHTED-VERTEX-COVER-1(G, w)

- $C = \emptyset$
- пока есть непокрытое ребро (v_1, v_2) , для которого $w(v_1) > 0$ и $w(v_2) > 0$, повторять:

Еще один алгоритм

Алгоритм

WEIGHTED-VERTEX-COVER-1(G, w)

- $C = \emptyset$
- пока есть непокрытое ребро (v_1, v_2) , для которого $w(v_1) > 0$ и $w(v_2) > 0$, повторять:
 - ▶ $\epsilon = \min\{w(v_1), w(v_2)\}$

Еще один алгоритм

Алгоритм

WEIGHTED-VERTEX-COVER-1(G, w)

- $C = \emptyset$
- пока есть непокрытое ребро (v_1, v_2) , для которого $w(v_1) > 0$ и $w(v_2) > 0$, повторять:
 - ▶ $\epsilon = \min\{w(v_1), w(v_2)\}$
 - ▶ $w(v_1) = w(v_1) - \epsilon$

Еще один алгоритм

Алгоритм

WEIGHTED-VERTEX-COVER-1(G, w)

- $C = \emptyset$
- пока есть непокрытое ребро (v_1, v_2) , для которого $w(v_1) > 0$ и $w(v_2) > 0$, повторять:
 - ▶ $\epsilon = \min\{w(v_1), w(v_2)\}$
 - ▶ $w(v_1) = w(v_1) - \epsilon$
 - ▶ $w(v_2) = w(v_2) - \epsilon$

Еще один алгоритм

Алгоритм

WEIGHTED-VERTEX-COVER-1(G, w)

- $C = \emptyset$
- пока есть непокрытое ребро (v_1, v_2) , для которого $w(v_1) > 0$ и $w(v_2) > 0$, повторять:
 - ▶ $\epsilon = \min\{w(v_1), w(v_2)\}$
 - ▶ $w(v_1) = w(v_1) - \epsilon$
 - ▶ $w(v_2) = w(v_2) - \epsilon$
- вернуть множество всех вершин веса 0

Переформулировка

Алгоритм

WEIGHTED-VERTEX-COVER-2(G, w)

Переформулировка

Алгоритм

WEIGHTED-VERTEX-COVER-2(G, w)

- если для каждого ребра $(v_1, v_2) \in E$ либо $w(v_1) = 0$, либо $w(v_2) = 0$, вернуть множество вершин веса 0

Переформулировка

Алгоритм

WEIGHTED-VERTEX-COVER-2(G, w)

- если для каждого ребра $(v_1, v_2) \in E$ либо $w(v_1) = 0$, либо $w(v_2) = 0$, вернуть множество вершин веса 0
- в противном случае выберем ребро, для которого $w(v_1) > 0$ и $w(v_2) > 0$

Переформулировка

Алгоритм

WEIGHTED-VERTEX-COVER-2(G, w)

- если для каждого ребра $(v_1, v_2) \in E$ либо $w(v_1) = 0$, либо $w(v_2) = 0$, вернуть множество вершин веса 0
- в противном случае выберем ребро, для которого $w(v_1) > 0$ и $w(v_2) > 0$
- определим весовую функцию δ так:

$$\delta(v) = \begin{cases} \min\{w(v_1), w(v_2)\}, & \text{если } v \in \{v_1, v_2\} \\ 0, & \text{в противном случае} \end{cases}$$

Перепформулировка

Алгоритм

WEIGHTED-VERTEX-COVER-2(G, w)

- если для каждого ребра $(v_1, v_2) \in E$ либо $w(v_1) = 0$, либо $w(v_2) = 0$, вернуть множество вершин веса 0
- в противном случае выберем ребро, для которого $w(v_1) > 0$ и $w(v_2) > 0$
- определим весовую функцию δ так:

$$\delta(v) = \begin{cases} \min\{w(v_1), w(v_2)\}, & \text{если } v \in \{v_1, v_2\} \\ 0, & \text{в противном случае} \end{cases}$$

- вернуть WEIGHTED-VERTEX-COVER-2($G, w - \delta$)

Анализ алгоритма

Лемма

Алгоритм `WEIGHTED-VERTEX-COVER-2` является 2-приближенным.

Анализ алгоритма

Лемма

Алгоритм `WEIGHTED-VERTEX-COVER-2` является 2-приближенным.

Доказательство

Анализ алгоритма

Лемма

Алгоритм WEIGHTED-VERTEX-COVER-2 является 2-приближенным.

Доказательство

- Обозначим через $\text{OPT}(w)$, $\text{OPT}(w - \delta)$, $\text{OPT}(\delta)$ веса оптимальных покрытий с весовыми функциями w , $w - \delta$, δ , соответственно, а через C_{opt} , $C_{\text{opt}}^{w-\delta}$, C_{opt}^{δ} — сами оптимальные покрытия.

Анализ алгоритма

Лемма

Алгоритм WEIGHTED-VERTEX-COVER-2 является 2-приближенным.

Доказательство

- Обозначим через $\text{OPT}(w)$, $\text{OPT}(w - \delta)$, $\text{OPT}(\delta)$ веса оптимальных покрытий с весовыми функциями w , $w - \delta$, δ , соответственно, а через C_{opt} , $C_{\text{opt}}^{w-\delta}$, C_{opt}^{δ} — сами оптимальные покрытия.
- $w(C_{\text{opt}}) = (w - \delta)(C_{\text{opt}}) + \delta(C_{\text{opt}}) \geq (w - \delta)(C_{\text{opt}}^{w-\delta}) + \delta(C_{\text{opt}}^{\delta})$

Анализ алгоритма

Лемма

Алгоритм WEIGHTED-VERTEX-COVER-2 является 2-приближенным.

Доказательство

- Обозначим через $\text{OPT}(w)$, $\text{OPT}(w - \delta)$, $\text{OPT}(\delta)$ веса оптимальных покрытий с весовыми функциями w , $w - \delta$, δ , соответственно, а через C_{opt} , $C_{\text{opt}}^{w-\delta}$, C_{opt}^{δ} — сами оптимальные покрытия.
- $w(C_{\text{opt}}) = (w - \delta)(C_{\text{opt}}) + \delta(C_{\text{opt}}) \geq (w - \delta)(C_{\text{opt}}^{w-\delta}) + \delta(C_{\text{opt}}^{\delta})$
- $\text{OPT}(w) \geq \text{OPT}(w - \delta) + \text{OPT}(\delta)$

Анализ алгоритма

Лемма

Алгоритм WEIGHTED-VERTEX-COVER-2 является 2-приближенным.

Доказательство

- Обозначим через $\text{OPT}(w)$, $\text{OPT}(w - \delta)$, $\text{OPT}(\delta)$ веса оптимальных покрытий с весовыми функциями w , $w - \delta$, δ , соответственно, а через C_{opt} , $C_{\text{opt}}^{w-\delta}$, C_{opt}^{δ} — сами оптимальные покрытия.
- $w(C_{\text{opt}}) = (w - \delta)(C_{\text{opt}}) + \delta(C_{\text{opt}}) \geq (w - \delta)(C_{\text{opt}}^{w-\delta}) + \delta(C_{\text{opt}}^{\delta})$
- $\text{OPT}(w) \geq \text{OPT}(w - \delta) + \text{OPT}(\delta)$
- Доказываем утверждение по индукцией по $\sum_{v \in V} w(v)$.

Анализ алгоритма

Лемма

Алгоритм WEIGHTED-VERTEX-COVER-2 является 2-приближенным.

Доказательство

- Обозначим через $\text{OPT}(w)$, $\text{OPT}(w - \delta)$, $\text{OPT}(\delta)$ веса оптимальных покрытий с весовыми функциями w , $w - \delta$, δ , соответственно, а через C_{opt} , $C_{\text{opt}}^{w-\delta}$, C_{opt}^{δ} — сами оптимальные покрытия.
- $w(C_{\text{opt}}) = (w - \delta)(C_{\text{opt}}) + \delta(C_{\text{opt}}) \geq (w - \delta)(C_{\text{opt}}^{w-\delta}) + \delta(C_{\text{opt}}^{\delta})$
- $\text{OPT}(w) \geq \text{OPT}(w - \delta) + \text{OPT}(\delta)$
- Доказываем утверждение по индукцией по $\sum_{v \in V} w(v)$.
- Если есть вершинное покрытие веса 0, то алгоритм обязательно его найдет.

Доказательство (продолжение)

Доказательство

Доказательство (продолжение)

Доказательство

- В противном случае алгоритм вызывает себя рекурсивно для весовой функции $w - \delta$.

Доказательство (продолжение)

Доказательство

- В противном случае алгоритм вызывает себя рекурсивно для весовой функции $w - \delta$.
- По индукционному предположению мы знаем, что $(w - \delta)(C) \leq 2OPT(w - \delta)$.

Доказательство (продолжение)

Доказательство

- В противном случае алгоритм вызывает себя рекурсивно для весовой функции $w - \delta$.
- По индукционному предположению мы знаем, что $(w - \delta)(C) \leq 2OPT(w - \delta)$.
- Также $\delta(C) \leq 2 \min\{w(v_1), w(v_2)\} \leq 2OPT(\delta)$.

Доказательство (продолжение)

Доказательство

- В противном случае алгоритм вызывает себя рекурсивно для весовой функции $w - \delta$.
- По индукционному предположению мы знаем, что $(w - \delta)(C) \leq 2OPT(w - \delta)$.
- Также $\delta(C) \leq 2 \min\{w(v_1), w(v_2)\} \leq 2OPT(\delta)$.
- Таким образом, $w(C) \leq 2(OPT(w - \delta) + OPT(\delta)) \leq 2OPT(w)$.



Еще один метод

Исходная

$$\min \sum_{v \in V} w(v)x(v)$$

$$x(v_1) + x(v_2) \geq 1, (v_1, v_2) \in E$$

$$x(v) \geq 0, v \in V$$

Двойственная

$$\max \sum_{e \in E} y(e)$$

$$\sum_{e: v \in e} y(e) \leq w(v), v \in V$$

$$y(e) \geq 0, e \in E$$

Из двойственности мы знаем, что $w(C_{\text{opt}}) \geq \sum_{e \in E} y(e)$, для любого допустимого решения y .

Алгоритм

Алгоритм

Алгоритм

Алгоритм

- положить $y(e) = 0$ для всех $e \in E$

Алгоритм

Алгоритм

- положить $y(e) = 0$ для всех $e \in E$
- пока есть ребро $e = (v_1, v_2)$, такое что $y(e)$ можно увеличить, увеличить $y(e)$, обратив хотя бы одно из ограничений $(\sum_{e:v \in e} \leq w(v))$ в равенство

Алгоритм

Алгоритм

- положить $y(e) = 0$ для всех $e \in E$
- пока есть ребро $e = (v_1, v_2)$, такое что $y(e)$ можно увеличить, увеличить $y(e)$, обратив хотя бы одно из ограничений $(\sum_{e:v \in e} \leq w(v))$ в равенство
- вернуть множество C , состоящее из всех вершин, для которых соответствующие ограничения обращены в равенства

Анализ

Алгоритм

Алгоритм

- положить $y(e) = 0$ для всех $e \in E$
- пока есть ребро $e = (v_1, v_2)$, такое что $y(e)$ можно увеличить, увеличить $y(e)$, обратив хотя бы одно из ограничений $(\sum_{e:v \in e} \leq w(v))$ в равенство
- вернуть множество C , состоящее из всех вершин, для которых соответствующие ограничения обращены в равенства

Анализ

- ясно, что найденное множество покрытием является

Алгоритм

Алгоритм

- положить $y(e) = 0$ для всех $e \in E$
- пока есть ребро $e = (v_1, v_2)$, такое что $y(e)$ можно увеличить, увеличить $y(e)$, обратив хотя бы одно из ограничений $(\sum_{e:v \in e} \leq w(v))$ в равенство
- вернуть множество C , состоящее из всех вершин, для которых соответствующие ограничения обращены в равенства

Анализ

- ясно, что найденное множество покрытием является
- $w(C) = \sum_{v \in C} w(v) = \sum_{v \in C} \sum_{e:v \in e} y(e) \leq 2 \sum_{e \in E} y(e) \leq 2w(C_{\text{opt}})$

План лекции

- 1 Задача о взвешенном вершинном покрытии
- 2 Задача о минимальном множестве представителей

Задача о минимальном множестве представителей

Определение

Задача о минимальном множестве представителей (hitting set problem) заключается в нахождении по данному множеству $X = \{x_1, \dots, x_m\}$ и множеству его подмножеств $S_1, \dots, S_n \subseteq X$ такого множества $C \subseteq X$, что $C \cap S_i \neq \emptyset$ для любого i .

Задача о минимальном множестве представителей

Определение

Задача о минимальном множестве представителей (hitting set problem) заключается в нахождении по данному множеству $X = \{x_1, \dots, x_m\}$ и множеству его подмножеств $S_1, \dots, S_n \subseteq X$ такого множества $C \subseteq X$, что $C \cap S_i \neq \emptyset$ для любого i .

NP-трудность

NP-трудность легко доказывается сведением от вершинного покрытия.

Задача о покрытии множествами

Определение

Задача о покрытии множествами (set cover problem) заключается в нахождении по данному универсальному множеству $U = \{u_1, \dots, u_n\}$ и множеству его подмножеств S_1, \dots, S_m , в объединении покрывающих все множество U , минимального набора подмножеств, покрывающего все множество U и имеющего минимальный возможный вес.

Двойственная задача

Двойственность

Двойственная задача

Двойственность

- По входу задачи о минимальном множестве представителей построим следующий двудольный граф:

Двойственная задача

Двойственность

- По входу задачи о минимальном множестве представителей построим следующий двудольный граф:
 - ▶ слева будут вершины x_1, \dots, x_m , справа — S_1, \dots, S_n ;

Двойственная задача

Двойственность

- По входу задачи о минимальном множестве представителей построим следующий двудольный граф:
 - ▶ слева будут вершины x_1, \dots, x_m , справа — S_1, \dots, S_n ;
 - ▶ вершину x_i соединяем ребром с вершиной S_j , если $x_i \in S_j$.

Двойственная задача

Двойственность

- По входу задачи о минимальном множестве представителей построим следующий двудольный граф:
 - ▶ слева будут вершины x_1, \dots, x_m , справа — S_1, \dots, S_n ;
 - ▶ вершину x_i соединяем ребром с вершиной S_j , если $x_i \in S_j$.
- Тогда требуется найти такое минимальное множество вершин в левой доли, что любая вершина из правой части имеет соседа в этом множестве.

Двойственная задача

Двойственность

- По входу задачи о минимальном множестве представителей построим следующий двудольный граф:
 - ▶ слева будут вершины x_1, \dots, x_m , справа — S_1, \dots, S_n ;
 - ▶ вершину x_i соединяем ребром с вершиной S_j , если $x_i \in S_j$.
- Тогда требуется найти такое минимальное множество вершин в левой доли, что любая вершина из правой части имеет соседа в этом множестве.
- Точно такой же граф строится по входу задачи о покрытии множествами (слева — подмножества, справа — элементы универсального множества).

Задача линейного программирования

Задача линейного программирования

Задача линейного программирования

Задача линейного программирования

- Задача о минимальном множестве представителей легко записывается в виде задачи целочисленного линейного программирования.

Задача линейного программирования

Задача линейного программирования

- Задача о минимальном множестве представителей легко записывается в виде задачи целочисленного линейного программирования.
- Пусть $x_i = 1$, если $x_i \in C$, и $x_i = 0$ в противном случае.

Задача линейного программирования

Задача линейного программирования

- Задача о минимальном множестве представителей легко записывается в виде задачи целочисленного линейного программирования.
- Пусть $x_i = 1$, если $x_i \in C$, и $x_i = 0$ в противном случае.
- Тогда задача запишется так:

$$\min \sum_{i=1}^m x_i$$

$$\sum_{i \in S_j} x_i \geq 1, 1 \leq j \leq n$$

$$x_i \in \{0, 1\}, 1 \leq i \leq m$$

Релаксация

Релаксация

Релаксация

Релаксация

- Итак, если x_1, \dots, x_n — оптимальное решение для полученной задачи, то $C = \{i | x_i = 1\}$ является минимальным множеством представителей.

Релаксация

Релаксация

- Итак, если x_1, \dots, x_n — оптимальное решение для полученной задачи, то $C = \{i | x_i = 1\}$ является минимальным множеством представителей.
- Заменяем условие $x_i \in \{0, 1\}$ на условие $0 \leq x_i \leq 1$ и решим полученную задачу линейного программирования.

Релаксация

- Итак, если x_1, \dots, x_n — оптимальное решение для полученной задачи, то $C = \{i | x_i = 1\}$ является минимальным множеством представителей.
- Заменяем условие $x_i \in \{0, 1\}$ на условие $0 \leq x_i \leq 1$ и решим полученную задачу линейного программирования.
- Пусть $\hat{x}_1, \dots, \hat{x}_m$ — найденное оптимальное решение.

Релаксация

- Итак, если x_1, \dots, x_n — оптимальное решение для полученной задачи, то $C = \{i | x_i = 1\}$ является минимальным множеством представителей.
- Заменяем условие $x_i \in \{0, 1\}$ на условие $0 \leq x_i \leq 1$ и решим полученную задачу линейного программирования.
- Пусть $\hat{x}_1, \dots, \hat{x}_m$ — найденное оптимальное решение.
- Ясно, что $\sum_{i=1}^m \hat{x}_i$ является нижней оценкой на размер минимального множества представителей C_{opt} .

Релаксация

- Итак, если x_1, \dots, x_n — оптимальное решение для полученной задачи, то $C = \{i | x_i = 1\}$ является минимальным множеством представителей.
- Заменяем условие $x_i \in \{0, 1\}$ на условие $0 \leq x_i \leq 1$ и решим полученную задачу линейного программирования.
- Пусть $\hat{x}_1, \dots, \hat{x}_m$ — найденное оптимальное решение.
- Ясно, что $\sum_{i=1}^m \hat{x}_i$ является нижней оценкой на размер минимального множества представителей C_{opt} .
- Рассмотрим теперь такой целочисленный набор: присваиваем x_i значение 1 с вероятностью \hat{x}_i .

Релаксация

- Итак, если x_1, \dots, x_n — оптимальное решение для полученной задачи, то $C = \{i | x_i = 1\}$ является минимальным множеством представителей.
- Заменяем условие $x_i \in \{0, 1\}$ на условие $0 \leq x_i \leq 1$ и решим полученную задачу линейного программирования.
- Пусть $\hat{x}_1, \dots, \hat{x}_m$ — найденное оптимальное решение.
- Ясно, что $\sum_{i=1}^m \hat{x}_i$ является нижней оценкой на размер минимального множества представителей C_{opt} .
- Рассмотрим теперь такой целочисленный набор: присваиваем x_i значение 1 с вероятностью \hat{x}_i .
- Положим $C = \{i | x_i = 1\}$.

Оценка вероятности

Оценка вероятности

Оценка вероятности

Оценка вероятности

- Конечно, C может и не быть допустимым решением (то есть не содержать ни одного представителя какого-то множества S_j).

Оценка вероятности

Оценка вероятности

- Конечно, C может и не быть допустимым решением (то есть не содержать ни одного представителя какого-то множества S_j).
- Мы хотим оценить сверху вероятность того, что $C \cap S_j = \emptyset$.

Оценка вероятности

Оценка вероятности

- Конечно, C может и не быть допустимым решением (то есть не содержать ни одного представителя какого-то множества S_j).
- Мы хотим оценить сверху вероятность того, что $C \cap S_j = \emptyset$.
- Пусть $|S_j| = k$.

Оценка вероятности

Оценка вероятности

- Конечно, C может и не быть допустимым решением (то есть не содержать ни одного представителя какого-то множества S_j).
- Мы хотим оценить сверху вероятность того, что $C \cap S_j = \emptyset$.
- Пусть $|S_j| = k$.
- Мы знаем, что $\sum_{i \in S_j} \hat{x}_i \geq 1$.

Оценка вероятности

Оценка вероятности

- Конечно, C может и не быть допустимым решением (то есть не содержать ни одного представителя какого-то множества S_j).
- Мы хотим оценить сверху вероятность того, что $C \cap S_j = \emptyset$.
- Пусть $|S_j| = k$.
- Мы знаем, что $\sum_{i \in S_j} \hat{x}_i \geq 1$.
-

$$\Pr(C \cap S_j = \emptyset) = \prod_{i \in S_j} (1 - \hat{x}_i) \leq \left(\frac{\sum_{i \in S_j} (1 - \hat{x}_i)}{k} \right)^k \leq (1 - 1/k)^k < e^{-1}$$

Повторение эксперимента

Повторение эксперимента

Повторение эксперимента

Повторение эксперимента

- Если взять объединение $t = \log(2n)$ таких множеств C , то вероятность того, что C не будет содержать ни одного представителя множества S_j , будет не более $(e^{-1})^{\log(2n)} = 1/2n$.

Повторение эксперимента

Повторение эксперимента

- Если взять объединение $t = \log(2n)$ таких множеств C , то вероятность того, что C не будет содержать ни одного представителя множества S_j , будет не более $(e^{-1})^{\log(2n)} = 1/2n$.
- Значит, такое объединение с вероятностью хотя бы $1/2$ является множеством представителей.

Повторение эксперимента

Повторение эксперимента

- Если взять объединение $t = \log(2n)$ таких множеств C , то вероятность того, что C не будет содержать ни одного представителя множества S_j , будет не более $(e^{-1})^{\log(2n)} = 1/2n$.
- Значит, такое объединение с вероятностью хотя бы $1/2$ является множеством представителей.
- Выбор $t = \log(2n)$ таких множеств C равносильен присваиванию x_i значения 1 с вероятностью $1 - (1 - \hat{x}_i)^t$.

Повторение эксперимента

Повторение эксперимента

- Если взять объединение $t = \log(2n)$ таких множеств C , то вероятность того, что C не будет содержать ни одного представителя множества S_j , будет не более $(e^{-1})^{\log(2n)} = 1/2n$.
- Значит, такое объединение с вероятностью хотя бы $1/2$ является множеством представителей.
- Выбор $t = \log(2n)$ таких множеств C равносильен присваиванию x_i значения 1 с вероятностью $1 - (1 - \hat{x}_i)^t$.
- Мат. ожидание размера C равно $\sum_{i=1}^m \hat{x}_i \leq C_{\text{opt}}$.

Повторение эксперимента

Повторение эксперимента

- Если взять объединение $t = \log(2n)$ таких множеств C , то вероятность того, что C не будет содержать ни одного представителя множества S_j , будет не более $(e^{-1})^{\log(2n)} = 1/2n$.
- Значит, такое объединение с вероятностью хотя бы $1/2$ является множеством представителей.
- Выбор $t = \log(2n)$ таких множеств C равносильно присваиванию x_i значения 1 с вероятностью $1 - (1 - \hat{x}_i)^t$.
- Мат. ожидание размера C равно $\sum_{i=1}^m \hat{x}_i \leq C_{\text{opt}}$.
- Таким образом, мат. ожидание размера построенного множества представителей не более чем в $\log n$ раз хуже оптимального.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Несколько 2-приближенных алгоритмов для задачи о взвешенном вершинном покрытии.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Несколько 2 -приближенных алгоритмов для задачи о взвешенном вершинном покрытии.
- $\log n$ -приближенный алгоритм для задачи о минимальном множестве представителей.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Несколько 2 -приближенных алгоритмов для задачи о взвешенном вершинном покрытии.
- $\log n$ -приближенный алгоритм для задачи о минимальном множестве представителей.
- Некоторые из приближенных алгоритмов являются вероятностными. В таком случае оценивается отношение мат. ожидания результата к оптимальному результату.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Несколько 2 -приближенных алгоритмов для задачи о взвешенном вершинном покрытии.
- $\log n$ -приближенный алгоритм для задачи о минимальном множестве представителей.
- Некоторые из приближенных алгоритмов являются вероятностными. В таком случае оценивается отношение мат. ожидания результата к оптимальному результату.
- Рассмотренные алгоритмы находят вещественное решение соответствующей задачи линейного программирования, а потом некоторым образом строят из него целочисленное решение.

Спасибо за внимание!