Алгоритмы для NP-трудных задач

Лекция 4: Вероятностные алгоритмы

А. Куликов

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

План лекции

1. 1.5^n алгоритм для задачи о 3-раскрашиваемости графа
2. $\log(2n)$-приближенный алгоритм для задачи о минимальном множе
3. FPT-алгоритм для задачи о пути длины k
4. $2^{2n/3}$ алгоритм для 3-SAT

План лекции

1. 1.5^n алгоритм для задачи о 3-раскрашиваемости графа
2. $\log(2n)$-приближенный алгоритм для задачи о минимальном множе
3. FPT-алгоритм для задачи о пути длины k
4. $2^{2n/3}$ алгоритм для 3-SAT

3-раскрашиваемость

Определение

Задача 3-раскрашиваемости (3-coloring problem) заключается в проверке, можно ли раскрасить данный граф правильным образом в три цвета (смежные вершины не покрашены в один цвет).

Полный перебор

Всего кандидатов на решение — 3^n.

3-раскрашиваемость

Алгоритм

Randomized-3-Coloring(G)

- повторить $c \cdot 1.5^n$ раз:
  - для каждой вершины выбрать случайным образом один из трех цветов, в который данная вершина не покрашена
  - записать формулу в 2-КНФ, которая выполнима тогда и только тогда, когда граф можно раскрасить с заданными ограничениями: для ребра (u, v), где u покрашена в цвет 1 или 2, а v — в цвет 2 или 3, запишем клозы

$$(u_1 \vee u_2) \wedge (v_2 \vee v_3) \wedge (\neg u_2 \vee \neg v_2)$$

  - если полученная формула выполнима, выдать ответ "да"
- выдать ответ "нет"

Анализ алгоритма

- Если алгоритм нашёл раскраску, то она, очевидно, правильная.
- Нужно оценить вероятность, с которой алгоритм не найдёт раскраску для 3-раскрашиваемого графа.
- Вероятность того, что на одной итерации он не угадает, не превосходит $(1 - (2/3)^n)$.
- Тогда вероятность того, что он не угадает ни за одну из $c \cdot 1.5^n$ итераций, не превосходит

$$(1 - (2/3)^n)^{c \cdot 1.5n} \le e^{-(2/3)^n \cdot c \cdot 1.5^n} = e^{-c}\,.$$

План лекции

1. 1.5^n алгоритм для задачи о 3-раскрашиваемости графа
2. $\log(2n)$-приближенный алгоритм для задачи о минимальном множе
3. FPT-алгоритм для задачи о пути длины k
4. $2^{2n/3}$ алгоритм для 3-SAT

Задача о минимальном множестве представителей

Определение

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

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

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

- 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 $\sum_{i \in S_j} \hat{x}_i \geq 1$.
-

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

Повторение эксперимента

Повторение эксперимента

- Если взять объединение $t = \log(2n)$ таких множеств C, то вероятность того, что объединение 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 равносилен 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 $\log(2n)$ раз хуже оптимального.

План лекции

1. 1.5^n алгоритм для задачи о 3-раскрашиваемости графа
2. $\log(2n)$-приближенный алгоритм для задачи о минимальном множе
3. FPT-алгоритм для задачи о пути длины k
4. $2^{2n/3}$ алгоритм для 3-SAT

Задача о пути длины k

Определение

Задачи о пути длины k (k-path problem) заключается в нахождении по данному графу G с двумя выделенными вершинами s и t и числу k простого пути между s и t, содержащего ровно k промежуточных вершин.

Color coding

Color coding

- Присвоим равновероятно и независимо всем вершинам, кроме s и t, цвета от 1 до k.
- Проверим, есть ли полноцветный s–t путь, то есть путь, в котором промежуточные вершины покрашены во все k цветов.

Оценка вероятности ошибки

Оценка вероятности ошибки

- Если путь длины k есть, то вероятность того, что он полноцветен, равна

$$\frac{k!}{k^k} > \frac{\left(\frac{k}{e}\right)^k}{k^k} = e^{-k}.$$

- Вероятность того, что путь не найдётся (если он есть) после e^k повторений, не превосходит

$$(1 - e^{-k})^{e^k} < \left(e^{-e^{-k}}\right)^{e^k} = e^{-1}.$$

Как же проверять наличие полноцветного пути?

Как же проверять наличие полноцветного пути?

- Мы знаем, что в полноцветном пути должны встречаться вершины всех k цветов, но не знаем, в каком порядке.
- Можно перебрать все $k!$ порядков. Когда порядок зафиксирован, нужно просто проверить наличие s–t пути в графе.

Итого

Алгоритм за время $O(e^k k! |E(G)|)$ решает задачу о пути длины k с вероятностью ошибки не более e^{-1}.

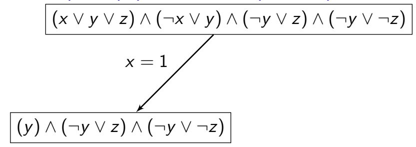
План лекции

1. 1.5^n алгоритм для задачи о 3-раскрашиваемости графа
2. $\log(2n)$-приближенный алгоритм для задачи о минимальном множе
3. FPT-алгоритм для задачи о пути длины k
4. $2^{2n/3}$ алгоритм для 3-SAT

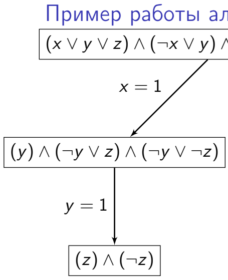
Пример работы алгоритма расщепления

$\boxed{(x \vee y \vee z) \wedge (\neg x \vee y) \wedge (\neg y \vee z) \wedge (\neg y \vee \neg z)}$

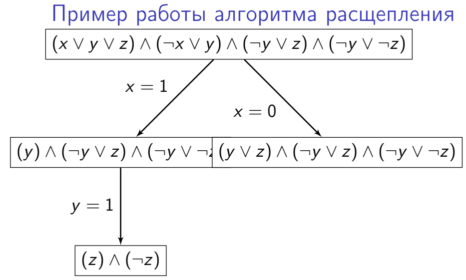
Пример работы алгоритма расщепления



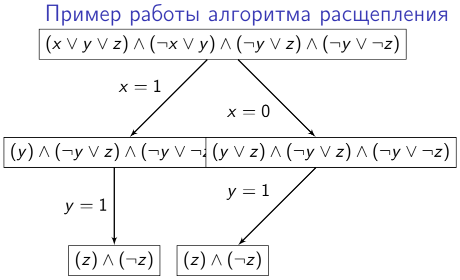
Пример работы алгоритма расщепления



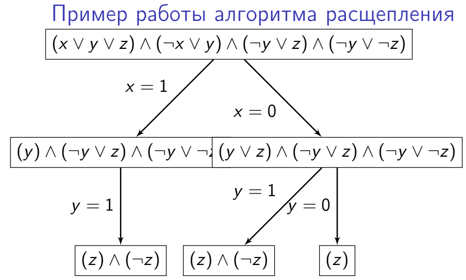
Пример работы алгоритма расщепления



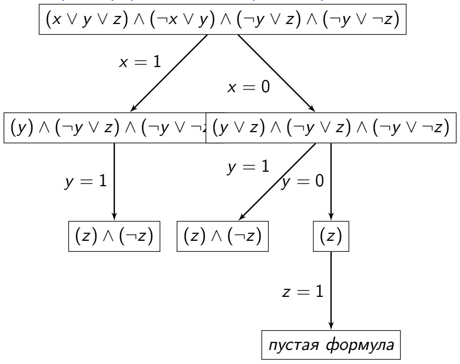
Пример работы алгоритма расщепления



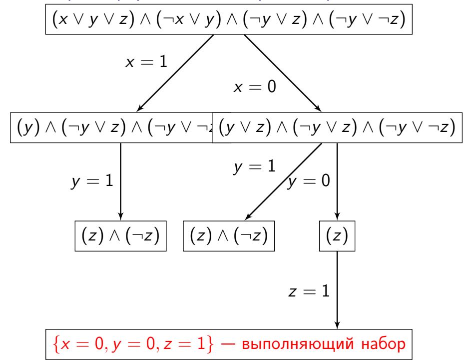
Пример работы алгоритма расщепления



Пример работы алгоритма расщепления



Пример работы алгоритма расщепления



Основные идеи алгоритма

- выберем случайную перестановку переменных и будем расщеплять по переменным в соответствующем порядке
- возможно, расщеплять нужно будет не по всем переменным, поскольку некоторые переменные могут получить значения в результате применения правил упрощения
- для оценки времени работы будем считать, по скольким переменным нам расщеплять не пришлось

Алгоритм

$\mathrm{PPSZ\text{-}SAT}(F)$

- выбрать случайным образом перестановку π из множества всех перестановок $\{1, \dots, n\}$
- построить дерево рекурсии глубины не более $2n/3$, где на каждом шаге
  - сначала применяем правило удаления единичных клозов,
  - а потом выбираем первую переменную из перестановки, все еще входящую в формулу, и расщепляемся по ней
- если на каком-то шаге выяснилось, что формула выполнима, выдать "выполнима"
- в противном случае выдать "невыполнима"

Анализ алгоритма

- очевидно, время работы этого алгоритма равно $\mathrm{poly}(|F|) \cdot 2^{2n/3}$
- мы покажем, что $\mathrm{PPSZ\text{-}SAT}$ с достаточно большой вероятностью находит правильное решение для формулы, у которой не более одного выполняющего набора
- рассмотрим дерево расщепления алгоритма, однако без ораничения $2n/3$ на его глубину
- если S — выполняющий набор для F, то это дерево содержит ветвь, ведущую в S
- каждое расщепление вдоль этой ветви присваивает значение переменной
- некоторым переменным значения также присваиваются правилами упрощения

Анализ алгоритма (продолжение)

- под описанием набора S будем понимать набор, отвечающий всем присваиваниям во время расщеплений, а под длиной описания — кол-во переменных в описании
- ясно, что по описанию выполняющего набора легко восстановить и сам выполняющий набор: присваиваем переменным значения из описания, при появлении единичных клозов — присваиваем значения их литералам

Лемма о кодировании выполняющего набора

Лемма

Пусть F — одновыполнимая формула в 3-КНФ, а S — ее выполняющий набор. Рассмотрим описание относительно перестановки, случайно и равновероятно выбранной из множества всех перестановок. Тогда математическое ожидание длины описания S не превосходит $2n/3$.

Доказательство

- заметим, что для каждой переменной x из F найдется хотя бы один клоз, в котором S выполняет только литерал, соответствующий x: если бы такого клоза не нашлось, то, изменив значение x в S, мы получили бы другой выполняющий набор
- клоз, удовлетворяющий такому условию, назовем критическим для переменной x

Доказательство (продолжение)

Доказательство

- поскольку π выбирается случайно и равновероятно, переменная x окажется в этой перестановке последней переменной критического клоза для x с вероятностью не менее $1/3$
- ясно, что в таком случае x не появится в описании S
- таким образом, с вероятностью хотя бы $1/3$ x не попадет в описание S
- осталось воспользоваться линейностью математического ожидания

$\square$

Анализ алгоритма (продолжение)

- пусть p_l — вероятность того, что длина описания выполняющего набора в точности равна l
- по только что доказанной лемме

$$\frac{2n}{3} \geq \sum_{l=0}^{n} p_l l \geq \left(\left\lfloor \frac{2n}{3} \right\rfloor + 1 \right) \sum_{l=\lfloor \frac{2n}{3} \rfloor + 1}^{n} p_l$$

- следовательно,

$$\sum_{l=\lfloor \frac{2n}{3} \rfloor + 1}^{n} p_l \leq \frac{2n}{3} \left(\left\lfloor \frac{2n}{3} \right\rfloor + 1 \right)^{-1} \leq \frac{2n}{3} \cdot \frac{3}{2n+1} = 1 - \frac{1}{2n+1}$$

Анализ алгоритма (продолжение)

- $$\sum_{l=0}^{\lfloor \frac{2n}{3} \rfloor + 1} p_l \geq \frac{1}{2n+1}$$
- итак, для случайной перестановки длина описания не превосходит $2n/3$ с вероятностью хотя бы $1/(2n+1)$
- если запустить алгоритм $(2n+1)$ раз, то алгоритм не найдет набор с вероятностью не более $(1 - \frac{1}{2n+1})^{2n+1} < \frac{1}{e}$
- а если запустить $c(2n+1)$ раз, то вероятность ошибки будет не более $\frac{1}{e^c}$

□

Основные идеи алгоритма еще раз

- случайно выбираем порядок переменных и расщепляем, периодически удаляя единичные клозы
- если у F есть ровно один выполняющий набор S, то для любой переменной x найдется клоз $(x \lor y \lor z)$, такой что S выполняет только литерал x в этом клозе (то есть S присваивает x, y, z значения 1, 0, 0, соответственно)
- если в перестановке π x идет после y и z, то в описание S она не попадет
- в среднем, примерно треть всех переменных не попадет в описание
- таким образом, если мы переберем все описания длины не более $2n/3$, то с хорошей вероятностью мы найдем описание выполняющего набора