

Алгоритмы для NP-трудных задач

Лекция 8: Алгоритмы расщепления

А. Куликов

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

1 / 19

2 / 19

План лекции

- 1 Комбинированные меры сложности
- 2 Запоминание дизъюнктов

3 / 19

Меры сложности

Стандартные меры сложности формул

- n — количество переменных
- K — количество кловов
- L — количество дизъюнктов

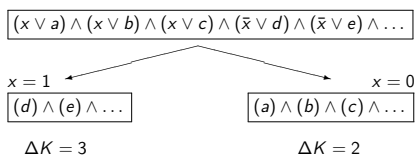
Стандартные меры сложности графов

- n — количество вершин
- m — количество ребер

Как правило, именно относительно этих мер хочется получить верхнюю оценку на время работы алгоритма. Однако для доказательства оценки можно использовать и другую меру.

5 / 19

Пример: K и K_2



6 / 19

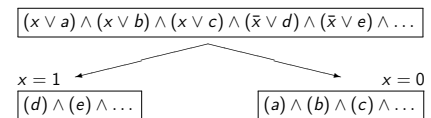
- 1 Комбинированные меры сложности
- 2 Запоминание дизъюнктов

Задача максимальной 2-выполнимости

- **Задача максимальной 2-выполнимости** (maximum 2-satisfiability) заключается в нахождении набора, выполняющего максимальное количество кловов данной формулы в 2-КНФ (то есть каждый клов которой содержит не более двух литералов).
- Является NP-трудной (сравните с 2-выполнимостью).
- В терминах максимальной 2-выполнимости легко формулируются многие задачи на графах.

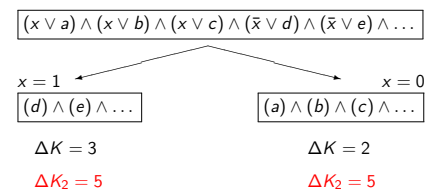
4 / 19

Пример: K и K_2



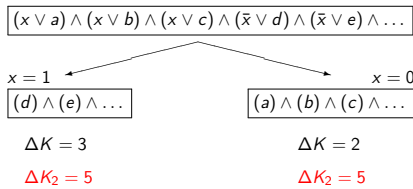
6 / 19

Пример: K и K_2



6 / 19

Пример: K и K_2



- $K_2 \leq K$
- K_2 — честная мера сложности: если $K_2(F) = 0$, от решение для F может быть найдено за линейное время

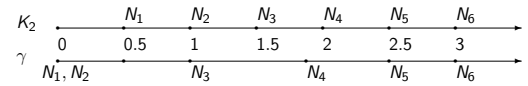
6 / 19

Комбинированная мера сложности

Обозначим через N_i количество переменных, входящих в ровно i 2-клоза.

количество 2-кловов: $K_2 = \sum_{i=1}^N \frac{i}{2} N_i$

комбинированная мера: $\gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=5}^N \frac{i}{2} N_i$



7 / 19

Верхняя оценка $2^{K/5.5}$

Теорема

MAX-2-SAT может быть решена за время $\text{poly}(K) \cdot 2^{K/5.5}$, где K — количество кловов формулы F .

Доказательство

- докажем оценку $2^{\gamma/5.5}$; этого достаточно, поскольку $\gamma \leq K_2 \leq K$
- для этого покажем, что присваивание значения переменной формулы уменьшает γ хотя бы на 5.5

8 / 19

Доказательство



- упрощенная формула не содержит переменных, входящих менее чем в три 2-клоза

9 / 19

Доказательство



- упрощенная формула не содержит переменных, входящих менее чем в три 2-клоза
- рассмотрим переменную входящую в хотя бы шесть 2-кловов: $\Delta\gamma \geq 3 + 6 \cdot 0.5 = 6$

9 / 19

Доказательство



- упрощенная формула не содержит переменных, входящих менее чем в три 2-клоза
- рассмотрим переменную входящую в хотя бы шесть 2-кловов: $\Delta\gamma \geq 3 + 6 \cdot 0.5 = 6$
- рассмотрим переменную входящую в ровно пять 2-кловов: $\Delta\gamma \geq 2.5 + 5 \cdot 0.6 = 5.5$

9 / 19

Доказательство



- упрощенная формула не содержит переменных, входящих менее чем в три 2-клоза
- рассмотрим переменную входящую в хотя бы шесть 2-кловов: $\Delta\gamma \geq 3 + 6 \cdot 0.5 = 6$
- рассмотрим переменную входящую в ровно пять 2-кловов: $\Delta\gamma \geq 2.5 + 5 \cdot 0.6 = 5.5$
- рассмотрим переменную входящую в ровно четыре 2-кловов: $\Delta\gamma \geq 1.9 + 4 \cdot 0.9 = 5.5$

9 / 19

$(n, 3)$ -MAX-2-SAT

- все переменные F входят не более чем в три 2-клоза
- если F упрощена, то все ее переменные входят ровно в три 2-клоза; значит, количество переменных четно
- $\gamma(F) = N(F)$
- необходимо показать, что после присваивания значения переменной еще хотя бы четыре другие переменные будут удалены правилами упрощения
- доказывается небольшим перебором случаев (есть и автоматическое доказательство)

10 / 19

1 Комбинированные меры сложности

2 Запоминание дизъюнктов

Как решить k -SAT за менее чем 2^n шагов?

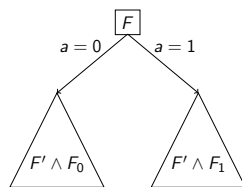
Алгоритм

- Выбрать произвольный клок $(x_1 \vee x_2 \vee \dots \vee x_t)$ данной формулы F в k -КНФ и произвести рекурсивные вызовы для
 - $F[x_1 = 1]$,
 - $F[x_1 = 0, x_2 = 1]$,
 - $F[x_1 = 0, x_2 = 0, x_3 = 1]$,
 - ...
 - $F[x_1 = 0, x_2 = 0, \dots, x_{t-1} = 0, x_t = 1]$.

Анализ

- Расщепление выше является $(1, 2, \dots, t)$ -расщеплением относительно количества переменных n .
- Поскольку $t \leq k$, $\tau(1, 2, \dots, t) \leq \tau(1, 2, \dots, k) = c_k < 2$.

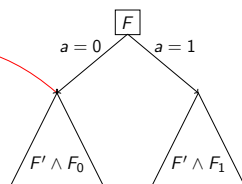
Основные идеи: иллюстрация



Основные идеи: иллюстрация

проверяем выполнимость F_0

если F_0 невыполнима, тогда не нужно идти в левую ветку



Запоминание дизъюнктов для задачи выполнимости

В алгоритмах, решающих задачу выполнимости, запоминание дизъюнктов означает, что алгоритм хранит частичные наборы переменных, которые не могут быть расширены до выполняющего.

Пример

$\{y = 0\}$ является таким набором для формулы

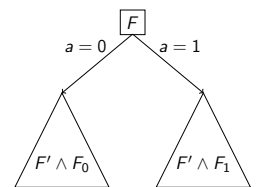
$$(x \vee y \vee z) \wedge (\neg x \vee \neg y) \wedge (y \vee z) \wedge (y \vee \neg z).$$

Основные идеи алгоритма для задачи выполнимости формулы константной плотности

- $F = F' \wedge (a \vee C_1) \wedge \dots \wedge (a \vee C_p) \wedge (\neg a \vee D_1) \wedge \dots \wedge (\neg a \vee D_q)$
- $F[a = 0] = F' \wedge \underbrace{C_1 \wedge \dots \wedge C_p}_{F_0}$, $F[a = 1] = F' \wedge \underbrace{D_1 \wedge \dots \wedge D_q}_{F_1}$
- Допустим, что F_0 сравнительно мала, так что её выполнимость можно проверить за полиномиальное время.
- Если F_0 невыполнима, тогда невыполнима и $F[a = 0]$ и не нужно расщепляться вообще (можно просто присвоить значение 1 переменной a).
- В противном случае найдём набор α , выполняющий F_0 .
- Тогда мы можем рекурсивно вызваться для $F[a = 0]$ и если получен ответ "невыполнима", тогда мы знаем, что α не может быть расширен до выполняющего набора F' . Таким образом, нет необходимости рассматривать расширения α в ветке $F[a = 1]$.

Основные идеи: иллюстрация

проверяем выполнимость F_0

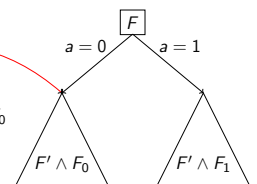


Основные идеи: иллюстрация

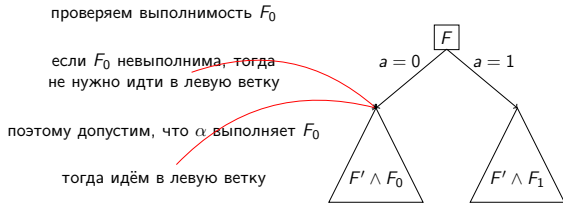
проверяем выполнимость F_0

если F_0 невыполнима, тогда не нужно идти в левую ветку

поэтому допустим, что α выполняет F_0

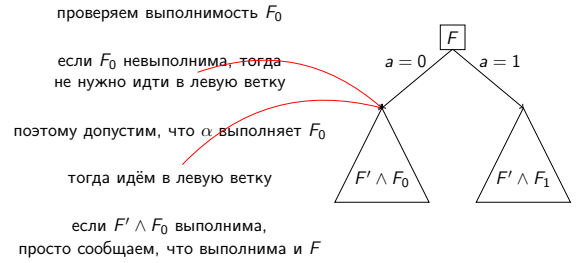


Основные идеи: иллюстрация



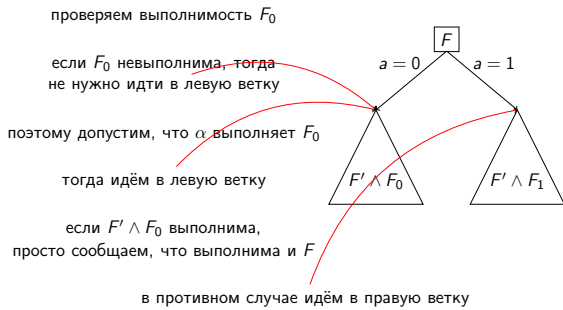
15 / 19

Основные идеи: иллюстрация



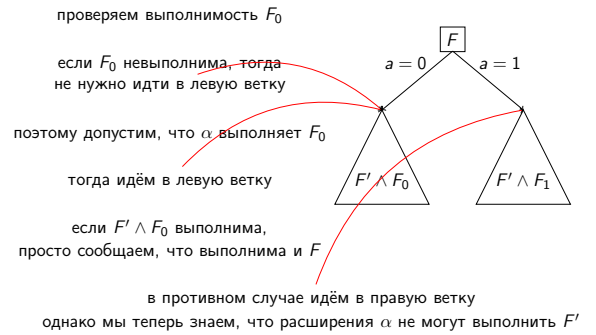
15 / 19

Основные идеи: иллюстрация



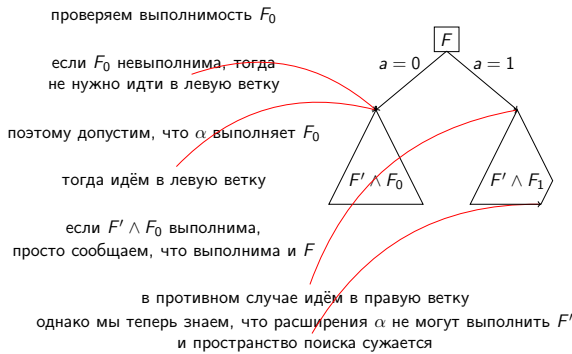
15 / 19

Основные идеи: иллюстрация



15 / 19

Основные идеи: иллюстрация



15 / 19

Алгоритм для выполнимости формул константной плотности

Пусть дана формула F с не более чем Δn кловзами.

- Если F содержит литерал a , такой что a и $\neg a$ встречаются хотя бы d раз (константа $d = d(\Delta)$ будет определена позже), расщепимся на $F[a=0]$ и $F[a=1]$.
- В противном случае выберем d^- -литерал a .
 - Пусть $F[a=0] = F' \wedge F_0$, $F[a=1] = F' \wedge F_1$.
 - Поскольку a является d^- -литералом, F_0 содержит не более d кловзов и мы можем проверить выполнимость F_0 за полиномиальное время.
 - Если F_0 невыполнима, тогда невыполнима и $F[a=0]$ и расщепляться не нужно (можно просто присвоить значение 1 переменной a).
 - В противном случае находим набор $\alpha = \{l_1, \dots, l_t\}$, выполняющий F_0 .

15 / 19

Продолжение описания алгоритма

- Напомним, что $a - d^-$ -литерал, $F[a=0] = F' \wedge F_0$, $F[a=1] = F' \wedge F_1$, $\alpha = \{l_1, \dots, l_t\}$ — выполняющий набор F_0 .
- Рекурсивно вызовем для $F[a=0]$.
- Если $F[a=0]$ выполнима, то выполнима и F .
- В противном случае $F[a=0]$ невыполнима и мы заключаем, что α не может быть расширен до выполняющего набора F' (ибо $F[a=0] = F' \wedge F_0$). Значит, в ветке $F[a=1]$ можно расщепиться так: $F[a=1, l_1=0]$, $F[a=1, l_1=1, l_2=0]$, \dots , $F[a=1, l_1=1, l_2=1, \dots, l_{t-1}=1, l_t=0]$.

17 / 19

Анализ времени работы алгоритма

- для оценки времени работы используется следующая мера сложности

$$\gamma(F) = n(F) + w \cdot m(F)$$

- расщепление по (d^+, d^-) -литералу даёт число расщепления не более $\tau(1 + wd, 1 + wd) = 2^{\frac{1}{1+wd}}$
- расщепление по d^- -литералу даёт число расщепления не более $\tau(1, 2, \dots, d+1)$
- положим d равным произвольной константе, большей Δ
- пусть w — константа, такая что $2^{\frac{1}{1+wd}} = \tau(1, 2, \dots, d+1)$
- тогда время работы алгоритма не превосходит $= 2^{\frac{n+wm}{1+wd}} < 2^{\frac{n+wdn}{1+wd}} = c^n$, где $c = 2^{\frac{1+wd}{1+wd}} < 2$ — константа

18 / 19

Заключение

- Итак, мы построили алгоритм для задачи выполнимости формулы константной плотности со временем работы c^n , где $c < 2$ — константа, а n — количество переменных.
- Напомним, что в общем случае задачи выполнимости (без ограничений на длину клона и количество клонов) таких алгоритмов неизвестно.