

Алгоритмы для NP-трудных задач

Лекция 1: Обзор

А. Куликов

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



- 1 Р и NP неформально

План лекции

- 1 P и NP неформально
- 2 Точные алгоритмы
 - Задача выполнимости
 - Задача о максимальном разрезе

Цель первых двух лекций

Привести несколько красивых алгоритмов, не особо вдаваясь в определения и доказательства. Все формальности будут дальше в курсе.

План лекции

- 1 P и NP неформально
- 2 Точные алгоритмы
 - Задача выполнимости
 - Задача о максимальном разрезе

Классы P и NP

Классы P и NP

- **Задача поиска** (search problem) задаётся алгоритмом C , который получает на вход условие I и кандидата на решение S и имеет время работы, ограниченное некоторым полиномом от $|I|$. S называется **решением** (solution), если и только если $C(S, I) = \text{true}$.

Классы P и NP

- **Задача поиска** (search problem) задаётся алгоритмом C , который получает на вход условие I и кандидата на решение S и имеет время работы, ограниченное некоторым полиномом от $|I|$. S называется **решением** (solution), если и только если $C(S, I) = \text{true}$.
- NP — класс всех задач поиска. Другими словами, NP — класс всех задач, решение для которых может быть быстро **проверено**.

Классы P и NP

- **Задача поиска** (search problem) задаётся алгоритмом C , который получает на вход условие I и кандидата на решение S и имеет время работы, ограниченное некоторым полиномом от $|I|$. S называется **решением** (solution), если и только если $C(S, I) = \text{true}$.
- NP — класс всех задач поиска. Другими словами, NP — класс всех задач, решение для которых может быть быстро **проверено**.
- P — класс задач поиска, решение для которых может быть быстро **найдено**.

Классы P и NP

- **Задача поиска** (search problem) задаётся алгоритмом C , который получает на вход условие I и кандидата на решение S и имеет время работы, ограниченное некоторым полиномом от $|I|$. S называется **решением** (solution), если и только если $C(S, I) = \text{true}$.
- NP — класс всех задач поиска. Другими словами, NP — класс всех задач, решение для которых может быть быстро **проверено**.
- P — класс задач поиска, решение для которых может быть быстро **найдено**.
- $P \neq NP$? Другими словами, существуют ли задачи, решение для которых может быть быстро проверено, но не может быть быстро найдено? Это один из самых важных и самых сложных открытых вопросов Theoretical Computer Science.

Сведения

Сведения

- Говорим, что задача A **сводится к** (reduces to) задаче B , и пишем $A \rightarrow B$, если по эффективному алгоритму для задачи B можно построить эффективный алгоритм для задачи A .

Сведения

- Говорим, что задача A **сводится к** (reduces to) задаче B , и пишем $A \rightarrow B$, если по эффективному алгоритму для задачи B можно построить эффективный алгоритм для задачи A .
- По-другому: если A решить сложно и $A \rightarrow B$, то и B решить сложно. То есть B не может быть сильно проще A .

Сведения

- Говорим, что задача A **сводится к** (reduces to) задаче B , и пишем $A \rightarrow B$, если по эффективному алгоритму для задачи B можно построить эффективный алгоритм для задачи A .
- По-другому: если A решить сложно и $A \rightarrow B$, то и B решить сложно. То есть B не может быть сильно проще A .
- Задача поиска называется **NP-полной** (NP-complete), если к ней сводятся все задачи поиска. То есть это универсальный притягивающий объект в классе NP.

Сведения

- Говорим, что задача A **сводится к** (reduces to) задаче B , и пишем $A \rightarrow B$, если по эффективному алгоритму для задачи B можно построить эффективный алгоритм для задачи A .
- По-другому: если A решить сложно и $A \rightarrow B$, то и B решить сложно. То есть B не может быть сильно проще A .
- Задача поиска называется **NP-полной** (NP-complete), если к ней сводятся все задачи поиска. То есть это универсальный притягивающий объект в классе NP.
- Удивительно (на первый взгляд), что такие задачи вообще существуют.

P vs NP

P vs NP

- Большинство исследователей считают, что $P \neq NP$.

P vs NP

- Большинство исследователей считают, что $P \neq NP$.
- Есть, впрочем, и другие мнения:

<http://www.win.tue.nl/~gwoegi/P-versus-NP.htm>

P vs NP

- Большинство исследователей считают, что $P \neq NP$.
- Есть, впрочем, и другие мнения:
<http://www.win.tue.nl/~gwoegi/P-versus-NP.htm>
- В предположении $P \neq NP$ не существует полиномиальных алгоритмов для NP-трудных задач.

Мотивация

Мотивация

- Многим приложениям требуется решать NP-трудные задачи, даже несмотря на то, что решения могут быть найдены только для весьма маленьких размеров входов.

Мотивация

- Многим приложениям требуется решать NP-трудные задачи, даже несмотря на то, что решения могут быть найдены только для весьма маленьких размеров входов.
- Лучшее понимание NP-трудных задач.

Мотивация

- Многим приложениям требуется решать NP-трудные задачи, даже несмотря на то, что решения могут быть найдены только для весьма маленьких размеров входов.
- Лучшее понимание NP-трудных задач.
- Новые интересные комбинаторные и алгоритмические задачи.

Мотивация

- Многим приложениям требуется решать NP-трудные задачи, даже несмотря на то, что решения могут быть найдены только для весьма маленьких размеров входов.
- Лучшее понимание NP-трудных задач.
- Новые интересные комбинаторные и алгоритмические задачи.
- Общая теория.

План лекции

- 1 Р и NP неформально
- 2 Точные алгоритмы
 - Задача выполнимости
 - Задача о максимальном разрезе

Точные алгоритмы для NP-трудных задач

Алгоритмы, находящие точное решение для данной задачи за время c^n для достаточно малой константы c .

Мотивация

Представим, что у нас есть алгоритм сложности 1.7^n для некоторой задачи, который за “разумное” время позволяет решать примеры этой задачи размера не более n_0 .

Мотивация

Представим, что у нас есть алгоритм сложности 1.7^n для некоторой задачи, который за “разумное” время позволяет решать примеры этой задачи размера не более n_0 .

- The “hardware” approach: возьмем в 10 раз более быстрый компьютер. Теперь мы можем решать примеры размера $n_0 + 4$.

Мотивация

Представим, что у нас есть алгоритм сложности 1.7^n для некоторой задачи, который за “разумное” время позволяет решать примеры этой задачи размера не более n_0 .

- The “hardware” approach: возьмем в 10 раз более быстрый компьютер. Теперь мы можем решать примеры размера $n_0 + 4$.
- The “brainware” approach: придумаем алгоритм сложности 1.3^n . Это позволит нам решать примеры размера $2 \cdot n_0$.

Мотивация

Представим, что у нас есть алгоритм сложности 1.7^n для некоторой задачи, который за “разумное” время позволяет решать примеры этой задачи размера не более n_0 .

- The “hardware” approach: возьмем в 10 раз более быстрый компьютер. Теперь мы можем решать примеры размера $n_0 + 4$.
- The “brainware” approach: придумаем алгоритм сложности 1.3^n . Это позволит нам решать примеры размера $2 \cdot n_0$.

Другими словами, уменьшение основания экспоненты времени работы алгоритма увеличивает размер решаемых за данное время примеров **в константное число раз**, в то время как использование более быстрого компьютера способно увеличить размер лишь **на константу**.

План лекции

- 1 Р и NP неформально
- 2 Точные алгоритмы
 - Задача выполнимости
 - Задача о максимальном разрезе

Пример

Формула

$$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$

невыполнима. То есть переменным x, y, z нельзя присвоить истинностные значения так, чтобы значение формулы стало истиной.

Задача выполнимости

Определение

Задача пропозициональной выполнимости (Boolean satisfiability problem, SAT): определить, выполнима ли данная формула в конъюнктивной нормальной форме, то есть существует ли набор булевых значений переменным формулы, выполняющий формулу. Такой набор называют **выполняющим** (satisfying assignment), а формулу, для которой такой набор существует, — **выполнимой** (satisfiable).

Хэммингово расстояние между наборами

Хэммингово расстояние между наборами

- Набор истинностных значений переменным формулы — это последовательность битов длины n , где n — количество переменных формулы. Всего, таким образом, есть 2^n различных наборов.

Хэммингово расстояние между наборами

- Набор истинностных значений переменным формулы — это последовательность битов длины n , где n — количество переменных формулы. Всего, таким образом, есть 2^n различных наборов.

Определение

Хэммингово расстояние между наборами

- Набор истинностных значений переменным формулы — это последовательность битов длины n , где n — количество переменных формулы. Всего, таким образом, есть 2^n различных наборов.

Определение

- **Хэммингово расстояние** двух наборов — количество переменных, которым эти наборы присваивают разные значения.

Хэммингово расстояние между наборами

- Набор истинностных значений переменным формулы — это последовательность битов длины n , где n — количество переменных формулы. Всего, таким образом, есть 2^n различных наборов.

Определение

- **Хэммингово расстояние** двух наборов — количество переменных, которым эти наборы присваивают разные значения.
- Для набора $t \in \{0, 1\}^n$ и числа d под **Хэмминговым шаром** $\mathcal{H}(t, d)$ (с центром в t и радиуса d) будем понимать множество всех наборов, находящихся на расстоянии не более чем d от t .

Поиск в шаре

Поиск в шаре

Для формулы F в 3-КНФ поиск выполняющего набора в шаре $\mathcal{H}(t, d)$ может быть осуществлен за время 3^d :

Поиск в шаре

Поиск в шаре

Для формулы F в 3-КНФ поиск выполняющего набора в шаре $\mathcal{H}(t, d)$ может быть осуществлен за время 3^d :

- если t не выполняет F , возьмем произвольный невыполненный клон $C = (x_1 \vee x_2 \vee x_3)$

Поиск в шаре

Поиск в шаре

Для формулы F в 3-КНФ поиск выполняющего набора в шаре $\mathcal{H}(t, d)$ может быть осуществлен за время 3^d :

- если t не выполняет F , возьмем произвольный невыполненный клон $C = (x_1 \vee x_2 \vee x_3)$
- рассмотрим три набора, полученных из t изменением значений переменных x_1 , x_2 и x_3

Поиск в шаре

Поиск в шаре

Для формулы F в 3-КНФ поиск выполняющего набора в шаре $\mathcal{H}(t, d)$ может быть осуществлен за время 3^d :

- если t не выполняет F , возьмем произвольный невыполненный клоз $C = (x_1 \vee x_2 \vee x_3)$
- рассмотрим три набора, полученных из t изменением значений переменных x_1 , x_2 и x_3
- хотя бы один из них будет ближе к выполняющему набору

$\sqrt{3}^n$ -Алгоритм

Алгоритм

SIMPLE-3-SAT(F)

$\sqrt{3}^n$ -Алгоритм

Алгоритм

SIMPLE-3-SAT(F)

- проверить, есть ли выполняющий набор в шарах $\mathcal{H}(0^n, n/2)$, $\mathcal{H}(1^n, n/2)$

$\sqrt{3}^n$ -Алгоритм

Алгоритм

SIMPLE-3-SAT(F)

- проверить, есть ли выполняющий набор в шарах $\mathcal{H}(0^n, n/2)$, $\mathcal{H}(1^n, n/2)$

Лемма

Время работы алгоритма SIMPLE-3-SAT есть $\sqrt{3}^n$, где $n = n(F)$ — число переменных формулы F .

$\sqrt{3}^n$ -Алгоритм

Алгоритм

SIMPLE-3-SAT(F)

- проверить, есть ли выполняющий набор в шарах $\mathcal{H}(0^n, n/2)$, $\mathcal{H}(1^n, n/2)$

Лемма

Время работы алгоритма SIMPLE-3-SAT есть $\sqrt{3}^n$, где $n = n(F)$ — число переменных формулы F .

Доказательство

Понятно, что если выполняющий набор есть, то он содержится в одном из рассмотренных двух шаров.

План лекции

- 1 Р и NP неформально
- 2 Точные алгоритмы
 - Задача выполнимости
 - Задача о максимальном разрезе

3-клика

Определение

Задача о 3-клике (3-clique problem) заключается в проверке наличия 3-клики (то есть полного подграфа на трех вершинах) во входном графе.

3-клика

Определение

Задача о 3-клике (3-clique problem) заключается в проверке наличия 3-клики (то есть полного подграфа на трех вершинах) во входном графе.

Алгоритм

MATRIX-CLIQUE(G)

3-клика

Определение

Задача о 3-клике (3-clique problem) заключается в проверке наличия 3-клики (то есть полного подграфа на трех вершинах) во входном графе.

Алгоритм

MATRIX-CLIQUE(G)

- построить матрицу смежности A графа G

3-клика

Определение

Задача о 3-клике (3-clique problem) заключается в проверке наличия 3-клики (то есть полного подграфа на трех вершинах) во входном графе.

Алгоритм

MATRIX-CLIQUE(G)

- построить матрицу смежности A графа G
- посчитать A^3

3-клика

Определение

Задача о 3-клике (3-clique problem) заключается в проверке наличия 3-клики (то есть полного подграфа на трех вершинах) во входном графе.

Алгоритм

MATRIX-CLIQUE(G)

- построить матрицу смежности A графа G
- посчитать A^3
- вернуть “да”, если на диагонали построенной матрицы есть хотя бы одна единица

3-клика

Определение

Задача о 3-клике (3-clique problem) заключается в проверке наличия 3-клики (то есть полного подграфа на трех вершинах) во входном графе.

Алгоритм

MATRIX-CLIQUE(G)

- построить матрицу смежности A графа G
- посчитать A^3
- вернуть “да”, если на диагонали построенной матрицы есть хотя бы одна единица
- вернуть “нет”

Анализ алгоритма

Лемма

Алгоритм выдает правильный ответ за время $O(n^\omega)$, где $\omega \approx 2.376$ — экспонента перемножения матриц (matrix multiplication exponent).

Анализ алгоритма

Лемма

Алгоритм выдает правильный ответ за время $O(n^\omega)$, где $\omega \approx 2.376$ — экспонента перемножения матриц (matrix multiplication exponent).

Доказательство

Анализ алгоритма

Лемма

Алгоритм выдает правильный ответ за время $O(n^\omega)$, где $\omega \approx 2.376$ — экспонента перемножения матриц (matrix multiplication exponent).

Доказательство

- важное свойство матрицы смежности: $A^k[i, j]$ есть количество путей длины k из вершины i в вершину j в графе G (легко доказать по индукции)

Анализ алгоритма

Лемма

Алгоритм выдает правильный ответ за время $O(n^\omega)$, где $\omega \approx 2.376$ — экспонента перемножения матриц (matrix multiplication exponent).

Доказательство

- важное свойство матрицы смежности: $A^k[i, j]$ есть количество путей длины k из вершины i в вершину j в графе G (легко доказать по индукции)
- 3-клика — это путь длины 3 из вершины в саму себя

Анализ алгоритма

Лемма

Алгоритм выдает правильный ответ за время $O(n^\omega)$, где $\omega \approx 2.376$ — экспонента перемножения матриц (matrix multiplication exponent).

Доказательство

- важное свойство матрицы смежности: $A^k[i, j]$ есть количество путей длины k из вершины i в вершину j в графе G (легко доказать по индукции)
- 3-клика — это путь длины 3 из вершины в саму себя
- для вычисления A^3 требуется два умножения матриц

Задача о максимальном разрезе

Определение

Задача о максимальном разрезе (maximum cut problem, MAX-CUT) заключается в нахождении такого разбиения вершин графа на две части, при котором количество ребер, концы которых принадлежат разным частям, максимально.

Задача о максимальном разрезе

Определение

Задача о максимальном разрезе (maximum cut problem, MAX-CUT) заключается в нахождении такого разбиения вершин графа на две части, при котором количество ребер, концы которых принадлежат разным частям, максимально.

Основные идеи сведения максимального разреза к 3-клике

Дан граф G на n вершинах.

Задача о максимальном разрезе

Определение

Задача о максимальном разрезе (maximum cut problem, MAX-CUT) заключается в нахождении такого разбиения вершин графа на две части, при котором количество ребер, концы которых принадлежат разным частям, максимально.

Основные идеи сведения максимального разреза к 3-клике

Дан граф G на n вершинах.

- Построим трехдольный граф H на $3 \cdot 2^{n/3}$ вершинах со следующим свойством: исходный граф G имеет разрез веса w тогда и только тогда, когда H содержит 3-клику веса w .

Задача о максимальном разрезе

Определение

Задача о максимальном разрезе (maximum cut problem, MAX-CUT) заключается в нахождении такого разбиения вершин графа на две части, при котором количество ребер, концы которых принадлежат разным частям, максимально.

Основные идеи сведения максимального разреза к 3-клике

Дан граф G на n вершинах.

- Построим трехдольный граф H на $3 \cdot 2^{n/3}$ вершинах со следующим свойством: исходный граф G имеет разрез веса w тогда и только тогда, когда H содержит 3-клику веса w .
- Используем быстрое умножение матриц для поиска 3-клики.

Задача о максимальном разрезе

Определение

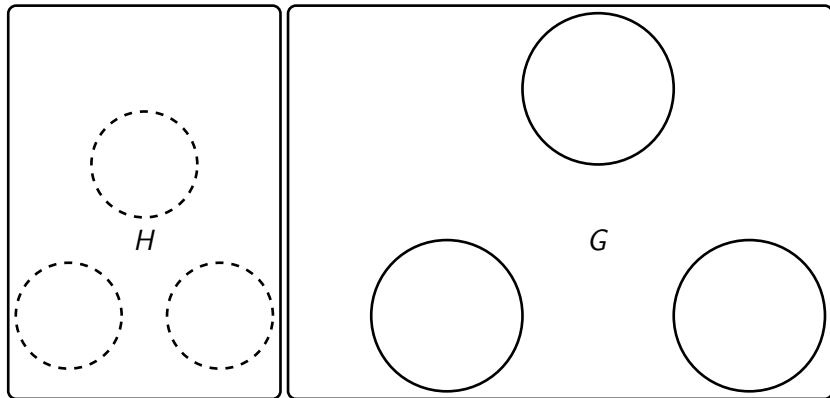
Задача о максимальном разрезе (maximum cut problem, MAX-CUT) заключается в нахождении такого разбиения вершин графа на две части, при котором количество ребер, концы которых принадлежат разным частям, максимально.

Основные идеи сведения максимального разреза к 3-клике

Дан граф G на n вершинах.

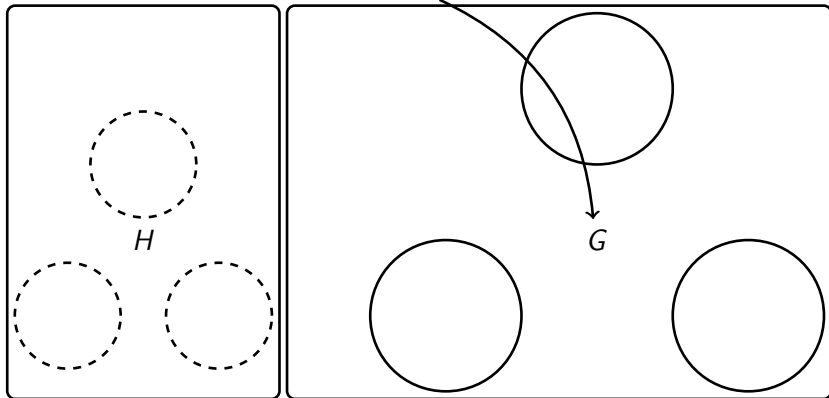
- Построим трехдольный граф H на $3 \cdot 2^{n/3}$ вершинах со следующим свойством: исходный граф G имеет разрез веса w тогда и только тогда, когда H содержит 3-клику веса w .
- Используем быстрое умножение матриц для поиска 3-клики.
- Сложность: $2^{\omega n/3} \approx 1.732^n$.

Вспомогательный граф



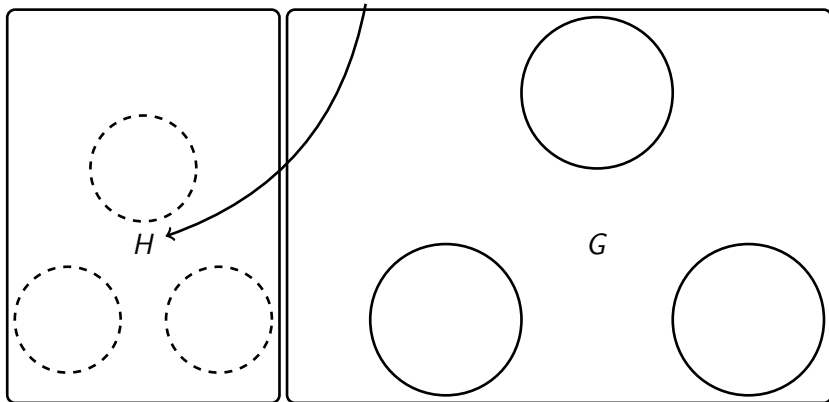
Вспомогательный граф

это три части вершин исходного графа G



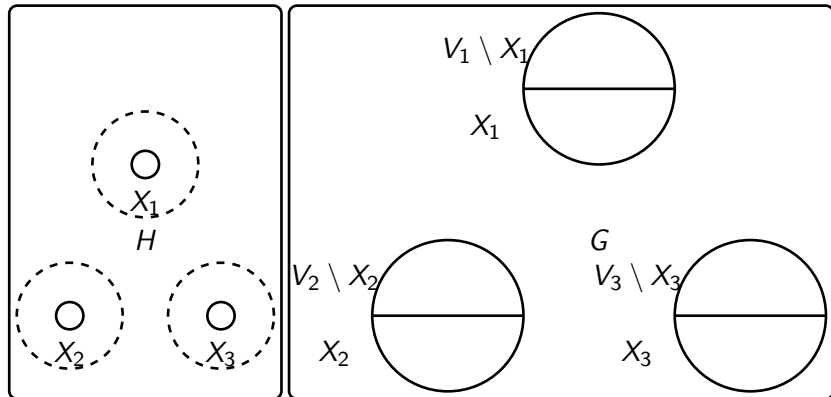
Вспомогательный граф

это три доли вершин вспомогательного огромного графа H



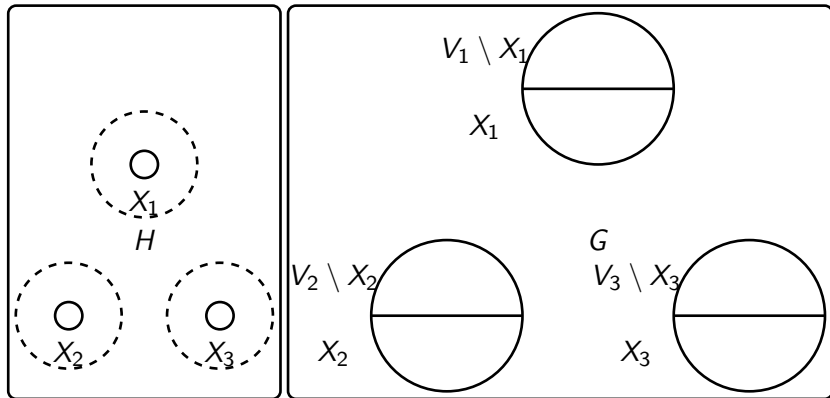
Вспомогательный граф

для каждого $X_i \subseteq V_i$ создаём вершину в соответствующей доле H



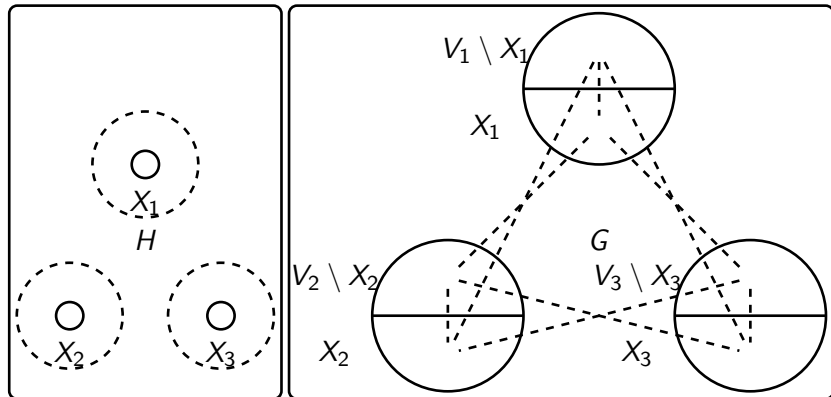
Вспомогательный граф

чему равен вес разреза $X_1 \cup X_2 \cup X_3$ в G ?

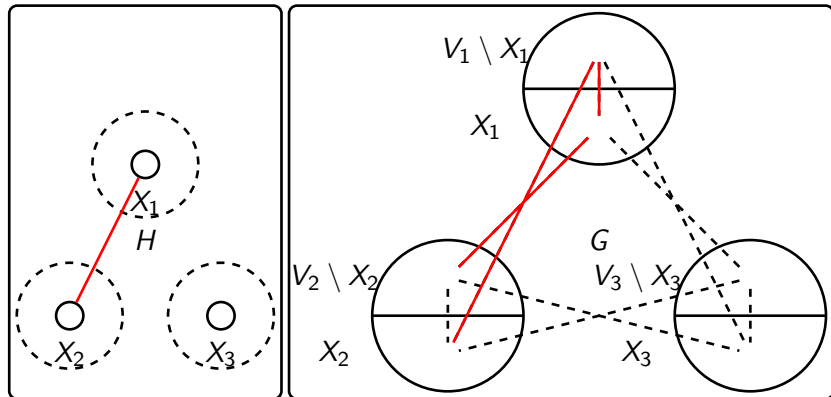


Вспомогательный граф

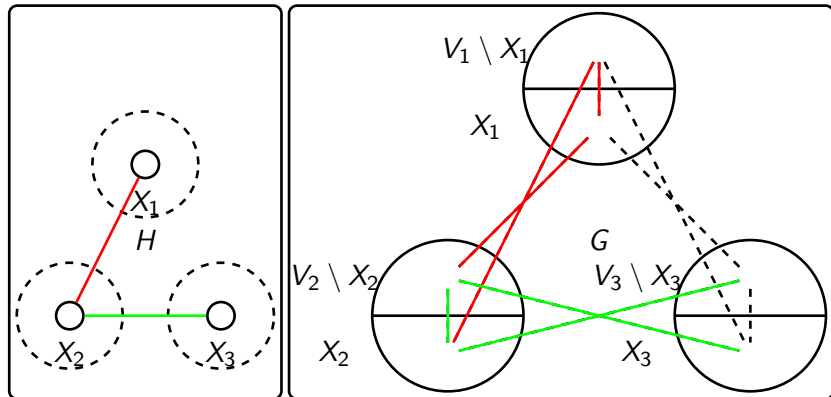
чему равен вес разреза $X_1 \cup X_2 \cup X_3$ в G ?



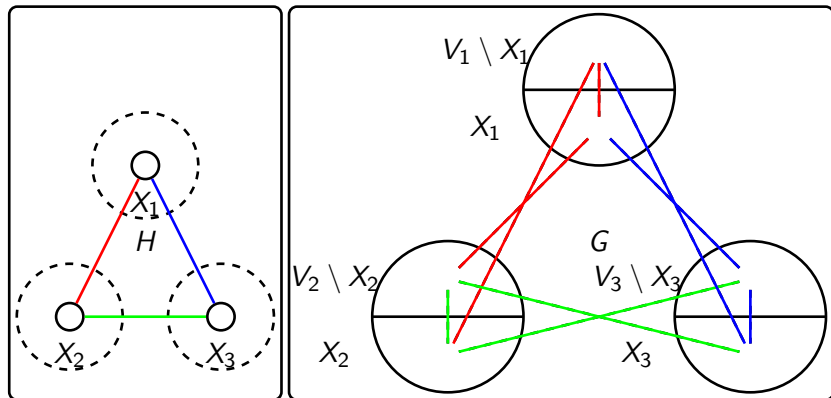
Вспомогательный граф



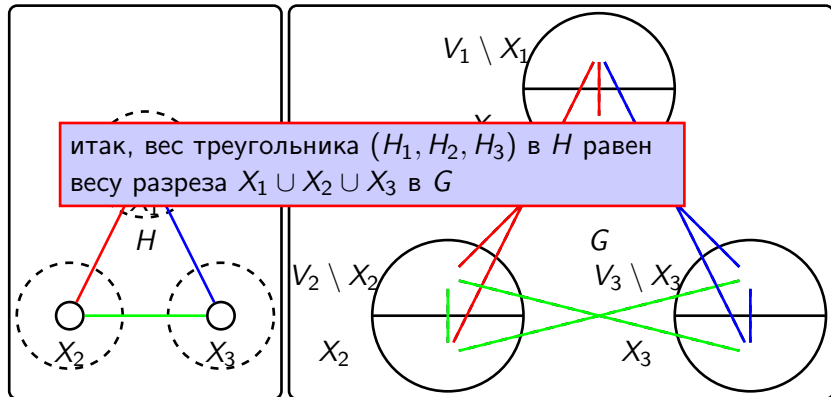
Вспомогательный граф



Вспомогательный граф



Вспомогательный граф



Алгоритм

Алгоритм

MATRIX-MAX-CUT(G)

Алгоритм

Алгоритм

MATRIX-MAX-CUT(G)

- разбить множество вершин V на три равные части V_1, V_2, V_3

Алгоритм

Алгоритм

MATRIX-MAX-CUT(G)

- разбить множество вершин V на три равные части V_1, V_2, V_3
- построить вспомогательный трехдольный граф H : i -я доля T_i содержит все возможные подмножества V_i ; вес ребра между X_1 и X_2 равен

$$w(V_2 \setminus X_2, X_1) + w(V_1 \setminus X_1, X_1) + w(V_1 \setminus X_1, X_2)$$

(для остальных пар долей — аналогично)

Алгоритм

Алгоритм

MATRIX-MAX-CUT(G)

- разбить множество вершин V на три равные части V_1, V_2, V_3
- построить вспомогательный трехдольный граф H : i -я доля T_i содержит все возможные подмножества V_i ; вес ребра между X_1 и X_2 равен

$$w(V_2 \setminus X_2, X_1) + w(V_1 \setminus X_1, X_1) + w(V_1 \setminus X_1, X_2)$$

(для остальных пар долей — аналогично)

- для всех $1 \leq w_{12}, w_{13}, w_{23} \leq |E_G|$

Алгоритм

Алгоритм

MATRIX-MAX-CUT(G)

- разбить множество вершин V на три равные части V_1, V_2, V_3
- построить вспомогательный трехдольный граф H : i -я доля T_i содержит все возможные подмножества V_i ; вес ребра между X_1 и X_2 равен

$$w(V_2 \setminus X_2, X_1) + w(V_1 \setminus X_1, X_1) + w(V_1 \setminus X_1, X_2)$$

(для остальных пар долей — аналогично)

- для всех $1 \leq w_{12}, w_{13}, w_{23} \leq |E_G|$
 - оставить только ребра веса w_{ij} между долями T_i и T_j

Алгоритм

Алгоритм

MATRIX-MAX-CUT(G)

- разбить множество вершин V на три равные части V_1, V_2, V_3
- построить вспомогательный трехдольный граф H : i -я доля T_i содержит все возможные подмножества V_i ; вес ребра между X_1 и X_2 равен

$$w(V_2 \setminus X_2, X_1) + w(V_1 \setminus X_1, X_1) + w(V_1 \setminus X_1, X_2)$$

(для остальных пар долей — аналогично)

- для всех $1 \leq w_{12}, w_{13}, w_{23} \leq |E_G|$
 - оставить только ребра веса w_{ij} между долями T_i и T_j
 - проверить, есть ли в модифицированном графе H 3-клика

Алгоритм

Алгоритм

MATRIX-MAX-CUT(G)

- разбить множество вершин V на три равные части V_1, V_2, V_3
- построить вспомогательный трехдольный граф H : i -я доля T_i содержит все возможные подмножества V_i ; вес ребра между X_1 и X_2 равен

$$w(V_2 \setminus X_2, X_1) + w(V_1 \setminus X_1, X_1) + w(V_1 \setminus X_1, X_2)$$

(для остальных пар долей — аналогично)

- для всех $1 \leq w_{12}, w_{13}, w_{23} \leq |E_G|$
 - оставить только ребра веса w_{ij} между долями T_i и T_j
 - проверить, есть ли в модифицированном графе H 3-клика
 - если да, то в G есть разрез веса $w_{12} + w_{13} + w_{23}$

Алгоритм

Алгоритм

MATRIX-MAX-CUT(G)

- разбить множество вершин V на три равные части V_1, V_2, V_3
- построить вспомогательный трехдольный граф H : i -я доля T_i содержит все возможные подмножества V_i ; вес ребра между X_1 и X_2 равен

$$w(V_2 \setminus X_2, X_1) + w(V_1 \setminus X_1, X_1) + w(V_1 \setminus X_1, X_2)$$

(для остальных пар долей — аналогично)

- для всех $1 \leq w_{12}, w_{13}, w_{23} \leq |E_G|$
 - оставить только ребра веса w_{ij} между долями T_i и T_j
 - проверить, есть ли в модифицированном графе H 3-клика
 - если да, то в G есть разрез веса $w_{12} + w_{13} + w_{23}$
- вернуть максимальную найденную стоимость разреза

Спасибо за внимание!