

с/к “Эффективные алгоритмы”

Лекция 18: Задача полуопределённого программирования

А. Куликов

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



План лекции

- 1 Задача полуопределённого программирования

План лекции

- 1 Задача полуопределённого программирования
- 2 Задача о максимальном разрезе

План лекции

- 1 Задача полуопределённого программирования
- 2 Задача о максимальном разрезе
- 3 Задача о раскраске графа

План лекции

- 1 Задача полуопределённого программирования
- 2 Задача о максимальном разрезе
- 3 Задача о раскраске графа

Положительно полуопределенные матрицы

Определение

Симметрическая матрица называется **положительно полуопределённой** (positive semidefinite), если все её собственные числа неотрицательны.

Обозначение: $A \succeq 0$.

Положительно полуопределенные матрицы

Определение

Симметрическая матрица называется **положительно полуопределённой** (positive semidefinite), если все её собственные числа неотрицательны. Обозначение: $A \succeq 0$.

Факт

Пусть $A \in \mathbb{R}^{n \times n}$ — симметрическая матрица. Следующие условия эквивалентны:

Положительно полуопределенные матрицы

Определение

Симметрическая матрица называется **положительно полуопределённой** (positive semidefinite), если все её собственные числа неотрицательны. Обозначение: $A \succeq 0$.

Факт

Пусть $A \in \mathbb{R}^{n \times n}$ — симметрическая матрица. Следующие условия эквивалентны:

- $A \succeq 0$;

Положительно полуопределенные матрицы

Определение

Симметрическая матрица называется **положительно полуопределённой** (positive semidefinite), если все её собственные числа неотрицательны. Обозначение: $A \succeq 0$.

Факт

Пусть $A \in \mathbb{R}^{n \times n}$ — симметрическая матрица. Следующие условия эквивалентны:

- $A \succeq 0$;
- квадратичная форма $x^T A x$ неотрицательна для любого $x \in \mathbb{R}^n$;

Положительно полуопределенные матрицы

Определение

Симметрическая матрица называется **положительно полуопределённой** (positive semidefinite), если все её собственные числа неотрицательны. Обозначение: $A \succeq 0$.

Факт

Пусть $A \in \mathbb{R}^{n \times n}$ — симметрическая матрица. Следующие условия эквивалентны:

- $A \succeq 0$;
- квадратичная форма $x^T A x$ неотрицательна для любого $x \in \mathbb{R}^n$;
- существуют вектора $u_1, \dots, u_n \in \mathbb{R}^m$, такие что $a_{ij} = u_i^T u_j$; или же $A = U^T U$ для некоторой U ;

Положительно полуопределенные матрицы

Определение

Симметрическая матрица называется **положительно полуопределённой** (positive semidefinite), если все её собственные числа неотрицательны. Обозначение: $A \succeq 0$.

Факт

Пусть $A \in \mathbb{R}^{n \times n}$ — симметрическая матрица. Следующие условия эквивалентны:

- $A \succeq 0$;
- квадратичная форма $x^T A x$ неотрицательна для любого $x \in \mathbb{R}^n$;
- существуют вектора $u_1, \dots, u_n \in \mathbb{R}^m$, такие что $a_{ij} = u_i^T u_j$; или же $A = U^T U$ для некоторой U ;
- A является неотрицательной линейной комбинацией матриц типа $x x^T$;

Положительно полуопределенные матрицы

Определение

Симметрическая матрица называется **положительно полуопределённой** (positive semidefinite), если все её собственные числа неотрицательны. Обозначение: $A \succeq 0$.

Факт

Пусть $A \in \mathbb{R}^{n \times n}$ — симметрическая матрица. Следующие условия эквивалентны:

- $A \succeq 0$;
- квадратичная форма $x^T A x$ неотрицательна для любого $x \in \mathbb{R}^n$;
- существуют вектора $u_1, \dots, u_n \in \mathbb{R}^m$, такие что $a_{ij} = u_i^T u_j$; или же $A = U^T U$ для некоторой U ;
- A является неотрицательной линейной комбинацией матриц типа $x x^T$;
- определитель любого симметрического минора A неотрицателен.

Общий вид задачи полуопределённого программирования

Общий вид задачи полуопределённого программирования

минимизировать $c^T x$

при условии $x_1 A_1 + \dots + x_n A_n - B \succeq 0$

где A_1, \dots, A_n, B симметрические матрицы размера $m \times m$, $c \in \mathbb{R}^n$.

Сходства с задачей линейного программирования

Замечания

Сходства с задачей линейного программирования

Замечания

- Если матрицы A_1, \dots, A_n, B являются диагональными, то получаем задачу линейного программирования.

Сходства с задачей линейного программирования

Замечания

- Если матрицы A_1, \dots, A_n, B являются диагональными, то получаем задачу линейного программирования.
- Поэтому задачу полуопределенного программирования можно рассматривать как обобщение задачи линейного программирования.

Сходства с задачей линейного программирования

Замечания

- Если матрицы A_1, \dots, A_n, B являются диагональными, то получаем задачу линейного программирования.
- Поэтому задачу полуопределенного программирования можно рассматривать как обобщение задачи линейного программирования.
- Есть, однако, и отличия.

Пример

Пример

минимизировать x_1 при условии $\begin{pmatrix} x_1 & 1 \\ 1 & x_2 \end{pmatrix} \succeq 0$

Пример

Пример

минимизировать x_1 при условии $\begin{pmatrix} x_1 & 1 \\ 1 & x_2 \end{pmatrix} \succeq 0$

Замечания

Условие положительной полуопределенности сводится к выполнению условий $x_1, x_2 \geq 0$ и $x_1 x_2 \geq 1$. Супремум 0 , таким образом, не достигается.

Факт

Факт

Существует алгоритм, находящий для данной задачи полуопределённого программирования решение с аддитивной ошибкой ϵ за время, полиномиальное от размера входных данных и от $\log \frac{1}{\epsilon}$.

План лекции

- 1 Задача полуопределённого программирования
- 2 Задача о максимальном разрезе
- 3 Задача о раскраске графа

Задача о максимальном разрезе

Определение

Задача о максимальном разрезе (maximum cut problem, MAX-CUT) заключается в нахождении по данному взвешенному неориентированному графу такого разбиения вершин на две части (раскраски в белый и чёрный цвета), при котором общий вес рёбер, соединяющих вершины разных частей, был бы максимален.

Задача о максимальном разрезе

Определение

Задача о максимальном разрезе (maximum cut problem, MAX-CUT) заключается в нахождении по данному взвешенному неориентированному графу такого разбиения вершин на две части (раскраски в белый и чёрный цвета), при котором общий вес рёбер, соединяющих вершины разных частей, был бы максимален.

Факт

Известно, что нахождение приближения, лучшего чем $16/17 \approx 0.94$, является NP-трудной задачей.

Формулировка в виде задачи целочисленного линейного программирования

Формулировка в виде задачи целочисленного линейного программирования

Формулировка в виде задачи целочисленного линейного программирования

Формулировка в виде задачи целочисленного линейного программирования

- Сопоставим каждой вершине $v_i \in V$ число y_i , где

$$y_i = \begin{cases} -1, & v_i \text{ — белая вершина,} \\ 1, & v_i \text{ — чёрная вершина.} \end{cases}$$

Формулировка в виде задачи целочисленного линейного программирования

Формулировка в виде задачи целочисленного линейного программирования

- Сопоставим каждой вершине $v_i \in V$ число y_i , где

$$y_i = \begin{cases} -1, & v_i \text{ — белая вершина,} \\ 1, & v_i \text{ — чёрная вершина.} \end{cases}$$

- Тогда ребро (v_i, v_j) принадлежит разрезу тогда и только тогда, когда $y_i y_j = -1$.

Формулировка в виде задачи целочисленного линейного программирования

Формулировка в виде задачи целочисленного линейного программирования

- Сопоставим каждой вершине $v_i \in V$ число y_i , где

$$y_i = \begin{cases} -1, & v_i \text{ — белая вершина,} \\ 1, & v_i \text{ — чёрная вершина.} \end{cases}$$

- Тогда ребро (v_i, v_j) принадлежит разрезу тогда и только тогда, когда $y_i y_j = -1$.
- Величина же разреза вычисляется как

$$f(M) = \sum_{i < j} \frac{1 - y_i y_j}{2} w_{ij}.$$

Релаксация: оптимизация на сфере большей размерности

Релаксация: оптимизация на сфере большей размерности

Релаксация: оптимизация на сфере большей размерности

Релаксация: оптимизация на сфере большей размерности

- Будем искать

$$\max_{v_1, \dots, v_n \in S^n} g((v_1, \dots, v_n)),$$

где

$$g((v_1, \dots, v_n)) = \sum_{i < j} \frac{1 - v_i^T v_j}{2} w_{ij},$$

а S^n — n -мерная единичная сфера.

Релаксация: оптимизация на сфере большей размерности

Релаксация: оптимизация на сфере большей размерности

- Будем искать

$$\max_{v_1, \dots, v_n \in S^n} g((v_1, \dots, v_n)),$$

где

$$g((v_1, \dots, v_n)) = \sum_{i < j} \frac{1 - v_i^T v_j}{2} w_{ij},$$

а S^n — n -мерная единичная сфера.

- Ясно, что максимальное значение является верхней оценкой на величину максимального разреза.

Вероятностное округление

Вероятностное округление

Вероятностное округление

Вероятностное округление

- Рассмотрим решение $(v_j)_j$ новой задачи.

Вероятностное округление

Вероятностное округление

- Рассмотрим решение $(v_i)_i$ новой задачи.
- Проведем через начало координат случайным образом гиперплоскость T размерности $n - 1$, которая разделит пространство на два полупространства A и B . После чего определим y_i так:

$$y_i = \begin{cases} 1, & v_i \in A, \\ -1, & v_i \in B. \end{cases}$$

Вероятностное округление

Вероятностное округление

- Рассмотрим решение $(v_i)_i$ новой задачи.
- Проведем через начало координат случайным образом гиперплоскость T размерности $n - 1$, которая разделит пространство на два полупространства A и B . После чего определим y_i так:

$$y_i = \begin{cases} 1, & v_i \in A, \\ -1, & v_i \in B. \end{cases}$$

- Иначе говоря, выберем случайным образом вектор $u \in S^n$ и построим y_i так:

$$y_i = \begin{cases} 1, & u^T v_i \geq 0, \\ -1, & u^T v_i < 0. \end{cases}$$

Оценка оптимальности

Лемма

Пусть $(v_i)_i$ — (приближённое) решение задачи, R' — значение g на этом наборе. Пусть также $(y_i)_i$ — набор, полученный с помощью алгоритма, описанного выше, из $(v_i)_i$ и $U' = f((y_i)_i)$. Тогда

$$E[U'] \geq 0.87R'.$$

Оценка оптимальности

Лемма

Пусть $(v_i)_i$ — (приближённое) решение задачи, R' — значение g на этом наборе. Пусть также $(y_i)_i$ — набор, полученный с помощью алгоритма, описанного выше, из $(v_i)_i$ и $U' = f((y_i)_i)$. Тогда

$$E[U'] \geq 0.87R'.$$

Доказательство

Необходимо доказать, что

$$E \sum_{i < j} \frac{1 - y_i y_j}{2} w_{ij} \geq 0.87 \sum_{i < j} \frac{1 - v_i^T v_j}{2} w_{ij}.$$

Доказательство (продолжение)

Доказательство

Доказательство (продолжение)

Доказательство

- Заметим, что $y_i y_j = -1$ тогда и только тогда, когда v_i и v_j лежат в разных полупространствах относительно нашей гиперплоскости T .

Доказательство (продолжение)

Доказательство

- Заметим, что $y_i y_j = -1$ тогда и только тогда, когда v_i и v_j лежат в разных полупространствах относительно нашей гиперплоскости T .
- $P(y_i \neq y_j) = \frac{\theta_{ij}}{\pi}$, где θ_{ij} — угол между векторами v_i и v_j .

Доказательство (продолжение)

Доказательство

- Заметим, что $y_i y_j = -1$ тогда и только тогда, когда v_i и v_j лежат в разных полупространствах относительно нашей гиперплоскости T .
- $P(y_i \neq y_j) = \frac{\theta_{ij}}{\pi}$, где θ_{ij} — угол между векторами v_i и v_j .

$$E \sum_{i < j} \frac{1 - y_i y_j}{2} w_{ij} = \sum_{i < j} \frac{\theta_{ij}}{\pi} w_{ij}.$$

Доказательство (продолжение)

Доказательство

- Заметим, что $y_i y_j = -1$ тогда и только тогда, когда v_i и v_j лежат в разных полупространствах относительно нашей гиперплоскости T .
- $P(y_i \neq y_j) = \frac{\theta_{ij}}{\pi}$, где θ_{ij} — угол между векторами v_i и v_j .

$$E \sum_{i < j} \frac{1 - y_i y_j}{2} w_{ij} = \sum_{i < j} \frac{\theta_{ij}}{\pi} w_{ij}.$$

$$\sum_{i < j} \frac{1 - v_i^T v_j}{2} w_{ij} = \sum_{i < j} \frac{1 - \cos \theta_{ij}}{2} w_{ij}.$$

Доказательство (продолжение)

Доказательство

- Заметим, что $y_i y_j = -1$ тогда и только тогда, когда v_i и v_j лежат в разных полупространствах относительно нашей гиперплоскости T .
- $P(y_i \neq y_j) = \frac{\theta_{ij}}{\pi}$, где θ_{ij} — угол между векторами v_i и v_j .

$$E \sum_{i < j} \frac{1 - y_i y_j}{2} w_{ij} = \sum_{i < j} \frac{\theta_{ij}}{\pi} w_{ij}.$$

$$\sum_{i < j} \frac{1 - v_i^T v_j}{2} w_{ij} = \sum_{i < j} \frac{1 - \cos \theta_{ij}}{2} w_{ij}.$$

- Осталось заметить, что $\min_{\theta} \frac{2}{\pi} \frac{\theta}{1 - \cos \theta} \approx 0.87$. □

Формулировка в виде задачи полуопределённого программирования

Задача полуопределённого программирования

$$\begin{array}{ll} \text{максимизировать} & \sum_{i < j} \frac{1-x_{ij}}{2} w_{ij} \\ \text{при условии} & X = \{x_{ij}\}_{i,j=1}^n \succeq 0; \forall i, x_{ii} = 1 \end{array}$$

Формулировка в виде задачи полуопределённого программирования

Задача полуопределённого программирования

$$\begin{array}{ll} \text{максимизировать} & \sum_{i < j} \frac{1-x_{ij}}{2} w_{ij} \\ \text{при условии} & X = \{x_{ij}\}_{i,j=1}^n \succeq 0; \forall i, x_{ii} = 1 \end{array}$$

Замечания

Найдя оптимальную матрицу X для данной задачи, найдем набор векторов u_1, \dots, u_n , для которого $X_{ij} = u_i^T u_j$ (поскольку на диагонали X стоят 1 , все найденные вектора будут единичными).

План лекции

- 1 Задача полуопределённого программирования
- 2 Задача о максимальном разрезе
- 3 Задача о раскраске графа

Задача о раскраске графа

Определение

Задача о раскраске графа (graph coloring problem) заключается в нахождении по данному графу раскраски его вершин с использованием минимального количества цветов так, чтобы концы любого ребра были покрашены в разные цвета.

Задача о раскраске графа

Определение

Задача о раскраске графа (graph coloring problem) заключается в нахождении по данному графу раскраски его вершин с использованием минимального количества цветов так, чтобы концы любого ребра были покрашены в разные цвета.

Замечания

Задача о раскраске графа

Определение

Задача о раскраске графа (graph coloring problem) заключается в нахождении по данному графу раскраски его вершин с использованием минимального количества цветов так, чтобы концы любого ребра были покрашены в разные цвета.

Замечания

- Задача является NP-трудной.

Задача о раскраске графа

Определение

Задача о раскраске графа (graph coloring problem) заключается в нахождении по данному графу раскраски его вершин с использованием минимального количества цветов так, чтобы концы любого ребра были покрашены в разные цвета.

Замечания

- Задача является NP-трудной.
- Даже нахождение правильной раскраски в 3 цвета данного 3-раскрашиваемого графа является NP-трудной.

Задача о раскраске графа

Определение

Задача о раскраске графа (graph coloring problem) заключается в нахождении по данному графу раскраски его вершин с использованием минимального количества цветов так, чтобы концы любого ребра были покрашены в разные цвета.

Замечания

- Задача является NP-трудной.
- Даже нахождение правильной раскраски в 3 цвета данного 3-раскрашиваемого графа является NP-трудной.
- Проверка же 2-раскрашиваемости проста.

Задача о раскраске графа

Определение

Задача о раскраске графа (graph coloring problem) заключается в нахождении по данному графу раскраски его вершин с использованием минимального количества цветов так, чтобы концы любого ребра были покрашены в разные цвета.

Замечания

- Задача является NP-трудной.
- Даже нахождение правильной раскраски в 3 цвета данного 3-раскрашиваемого графа является NP-трудной.
- Проверка же 2-раскрашиваемости проста.
- Граф всегда можно покрасить в $\delta + 1$ цвет, где δ — максимальная степень графа.

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

- Рассмотрим произвольную вершину i и всех её соседей $N(i)$.

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

- Рассмотрим произвольную вершину i и всех её соседей $N(i)$.
- Ясно, что $N(i)$ можно покрасить в два цвета.

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

- Рассмотрим произвольную вершину i и всех её соседей $N(i)$.
- Ясно, что $N(i)$ можно покрасить в два цвета.
- Алгоритм: пока есть вершина степени хотя бы \sqrt{n} , покрасим её соседей в новые два цвета и выкинем их всех из графа; через не более чем \sqrt{n} таких операций останется граф степени менее \sqrt{n} , который мы покрасим в новые \sqrt{n} цветов.

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

Раскраска 3-раскрашиваемого графа в \sqrt{n} цветов

- Рассмотрим произвольную вершину i и всех её соседей $N(i)$.
- Ясно, что $N(i)$ можно покрасить в два цвета.
- Алгоритм: пока есть вершина степени хотя бы \sqrt{n} , покрасим её соседей в новые два цвета и выкинем их всех из графа; через не более чем \sqrt{n} таких операций останется граф степени менее \sqrt{n} , который мы покрасим в новые \sqrt{n} цветов.
- Потратим, таким образом, не более чем $3\sqrt{n}$ цветов.

Векторная 3-раскрасиваемость

Определение

Граф называется **векторно 3-раскрасиваемым** (vector 3-colorable), если каждой вершине графа можно приписать вектор так, что для любого ребра $(i, j) \in E$ выполнено равенство $v_i^T v_j = -1/2$ (и векторно 2-раскрасиваемым, если $v_i^T v_j = -1$).

Векторная 3-раскрасиваемость

Определение

Граф называется **векторно 3-раскрасиваемым** (vector 3-colorable), если каждой вершине графа можно приписать вектор так, что для любого ребра $(i, j) \in E$ выполнено равенство $v_i^T v_j = -1/2$ (и векторно 2-раскрасиваемым, если $v_i^T v_j = -1$).

Замечания

Векторная 3-раскрасиваемость

Определение

Граф называется **векторно 3-раскрасиваемым** (vector 3-colorable), если каждой вершине графа можно приписать вектор так, что для любого ребра $(i, j) \in E$ выполнено равенство $v_i^T v_j = -1/2$ (и векторно 2-раскрасиваемым, если $v_i^T v_j = -1$).

Замечания

- Любой 3-раскрасиваемый граф является векторно 3-раскрасиваемым. Обратное, вообще говоря, неверно.

Векторная 3-раскрасиваемость

Определение

Граф называется **векторно 3-раскрасиваемым** (vector 3-colorable), если каждой вершине графа можно приписать вектор так, что для любого ребра $(i, j) \in E$ выполнено равенство $v_i^T v_j = -1/2$ (и векторно 2-раскрасиваемым, если $v_i^T v_j = -1$).

Замечания

- Любой 3-раскрасиваемый граф является векторно 3-раскрасиваемым. Обратное, вообще говоря, неверно.
- А вот 2-раскрасиваемость эквивалентна векторной 2-раскрасиваемости.

Половинная раскраска

Определение

Половинной раскраской (half-coloring) графа называется правильная раскраска хотя бы половины вершин графа.

Половинная раскраска

Определение

Половинной раскраской (half-coloring) графа называется правильная раскраска хотя бы половины вершин графа.

Замечание

Если половинная раскраска графа в n^α цветов может быть найдена за полиномиальное время, то граф можно полностью раскрасить за полиномиальное время в

$$n^\alpha + \left(\frac{n}{2}\right)^\alpha + \dots = O(n^\alpha)$$

цветов.

Алгоритм и его анализ

Алгоритм и его анализ

Алгоритм и его анализ

Алгоритм и его анализ

- Найдём необходимый набор векторов, решив соответствующую задачу полуопределённого программирования.

Алгоритм и его анализ

Алгоритм и его анализ

- Найдём необходимый набор векторов, решив соответствующую задачу полуопределённого программирования.
- Случайно выбранная гиперплоскость разделяет два вектора v_i, v_j для $(i, j) \in E$ с вероятностью $2/3$.

Алгоритм и его анализ

Алгоритм и его анализ

- Найдём необходимый набор векторов, решив соответствующую задачу полуопределённого программирования.
- Случайно выбранная гиперплоскость разделяет два вектора v_i, v_j для $(i, j) \in E$ с вероятностью $2/3$.
- Такая гиперплоскость, таким образом, может быть использована для покраски вершин в два цвета с мат. ожиданием количества одноцветных рёбер, равным одной трети общего количества рёбер.

Алгоритм и его анализ

Алгоритм и его анализ

- Найдём необходимый набор векторов, решив соответствующую задачу полуопределённого программирования.
- Случайно выбранная гиперплоскость разделяет два вектора v_i, v_j для $(i, j) \in E$ с вероятностью $2/3$.
- Такая гиперплоскость, таким образом, может быть использована для покраски вершин в два цвета с мат. ожиданием количества одноцветных рёбер, равным одной трети общего количества рёбер.
- Если же провести r случайных гиперплоскостей и покрасить вершины в соответствующие 2^r цветов, то ребро графа будет одноцветным с вероятностью $1/3^r$.

Алгоритм и его анализ

- Найдём необходимый набор векторов, решив соответствующую задачу полуопределённого программирования.
- Случайно выбранная гиперплоскость разделяет два вектора v_i, v_j для $(i, j) \in E$ с вероятностью $2/3$.
- Такая гиперплоскость, таким образом, может быть использована для покраски вершин в два цвета с мат. ожиданием количества одноцветных рёбер, равным одной трети общего количества рёбер.
- Если же провести r случайных гиперплоскостей и покрасить вершины в соответствующие 2^r цветов, то ребро графа будет одноцветным с вероятностью $1/3^r$.
- Тогда мат. ожидание количества одноцветных рёбер равно $|E|/3^r = \frac{dn}{2 \cdot 3^r} < n/4$, если $r \geq \log_3 d + 1$, где d — средняя степень графа.

Алгоритм и его анализ (завершение)

Алгоритм и его анализ (завершение)

Алгоритм и его анализ (завершение)

Алгоритм и его анализ (завершение)

- С большой вероятностью, таким образом, количество одноцветных рёбер будет меньше $n/2$.

Алгоритм и его анализ (завершение)

Алгоритм и его анализ (завершение)

- С большой вероятностью, таким образом, количество одноцветных рёбер будет меньше $n/2$.
- Сотрём цвет одного из концов каждого одноцветного ребра и получим половинную раскраску.

Алгоритм и его анализ (завершение)

Алгоритм и его анализ (завершение)

- С большой вероятностью, таким образом, количество одноцветных рёбер будет меньше $n/2$.
- Сотрём цвет одного из концов каждого одноцветного ребра и получим половинную раскраску.
- Использовали $2^r = O(d^{\log_2 3}) = O(d^{0.631})$ цветов.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Задача полуопределённого программирования заключается в минимизации (максимизации) линейной функции при условии положительной полуопределённости некоторой матрицы.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Задача полуопределённого программирования заключается в минимизации (максимизации) линейной функции при условии положительной полуопределённости некоторой матрицы.
- Данная задача решается за полиномиальное время с любой заданной заранее аддитивной ошибкой.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Задача полуопределённого программирования заключается в минимизации (максимизации) линейной функции при условии положительной полуопределённости некоторой матрицы.
- Данная задача решается за полиномиальное время с любой заданной заранее аддитивной ошибкой.
- Используя этот алгоритм, можно за полиномиальное время найти 0.87 -приближение для задачи о максимальном разрезе и покрасить в $d^{0.631}$ цветов 3 -раскрашиваемый граф.

Спасибо за внимание!