Дискретная оптимизация
Сложные комбинаторные задачи

Роланд Хильдебранд

Институт Вычислительной Математики им. Г.И. Марчука РАН

7 апреля 2026 г.

План

сложные комбинаторные задачи

- **задача об упаковке**
- покрытие подмножествами
- покраска графа
- наибольшая клика

Задача об упаковке

пусть заданы

- список объектов $1, \ldots, n$ с объёмами $w_i > 0$
- ёмкости с вместимостью $W \geq \max_i w_i$

задача об упаковке состоит в том, чтобы найти упаковку всех объектов в *наименьшее* число ёмкостей, не нарушая ограничения на вместимость

это NP сложная задача

более того, сложно решить задачу с гарантией

$$\hat{N} \leq \left(\frac{3}{2} - \epsilon\right) N_{opt}$$

Задача об упаковке

действительно, пусть

$$w_1, \ldots, w_n \in \mathbb{N}_+, \ \sum_i w_i = \bar{w}, \ W = \frac{\bar{w}}{2} \geq \max_i w_i$$

если объекты делятся на два подмножества с равными суммарными весами, то $N_{opt} = 2$, иначе $N_{opt} = 3$

почему $N_{opt} \leq 3$

- пусть $N_{opt} > 2$
- не более, чем в одной ёмкости суммарный вес объектов $\leq \frac{W}{2}$ (иначе их можно сложить вместе)
- в двух ёмкостях суммарный вес $> 2 \cdot \frac{W}{2} = W$
- оставшиеся объекты имеют суммарный вес $< W$ и умещаются в 3-ю ёмкость

Задача об упаковке

алгоритм, требующий линейное время: следующий подходящий (next fit — NF)

- рассматриваем объекты один за другим в произвольном порядке
- нумеруем ёмкости индексами $1, \ldots, n$
- кладём текущий объект в текущую ёмкость, если он входит, и в следующую, если не входит

после помещения объекта достаточно вычесть его объём из остаточного объёма в текущей ёмкости — n вычитаний и n сравнений

Задача об упаковке

гарантия на качество:

- пусть N — число ёмкостей в полученном решении, N_{opt} — наименьшее число ёмкостей
- пусть в ёмкости k — объекты с суммарным весом W_k
- в двух соседних ёмкостях веса суммарно больше, чем W

получаем

$$W_1 + 2W_2 + \cdots + 2W_{N-1} + W_N > (N-1)W$$

отсюда

$$N_{opt}\,W \geq \sum_{k=1}^{N} W_k > \frac{N-1}{2}W$$

в итоге

$$N \leq 2N_{opt}$$

гарантированный фактор качества решения — 2

Задача об упаковке

жадные алгоритмы:

первый/лучший подходящий — first/best fit decreasing (FFD/BFD)

- ранжируем объекты по объёму в низходящем порядке
- нумеруем ёмкости индексами $1, \ldots, n$
- начиная с наибольшего, кладём объект в первую ёмкость, в которую он вмещается / в ёмкость, в которую он вмещается с минимальным остаточным объёмом

Задача об упаковке

гарантия на качество алгоритма FFD/BFD

если N — количество использованных в решении ёмкостей, а N_{opt} — оптимальное число, то

$$N \leq \left\lfloor \frac{11}{9} N_{opt} + \frac{6}{9} \right\rfloor$$

пример, показывающий неулучшаемость оценки

пример

- ёмкость $W = 60$
- $6k + 4$ объектов с весами 31,17,16, соответственгно (всего $18k + 12$ объектов)
- $12k + 8$ объектов с весом 13

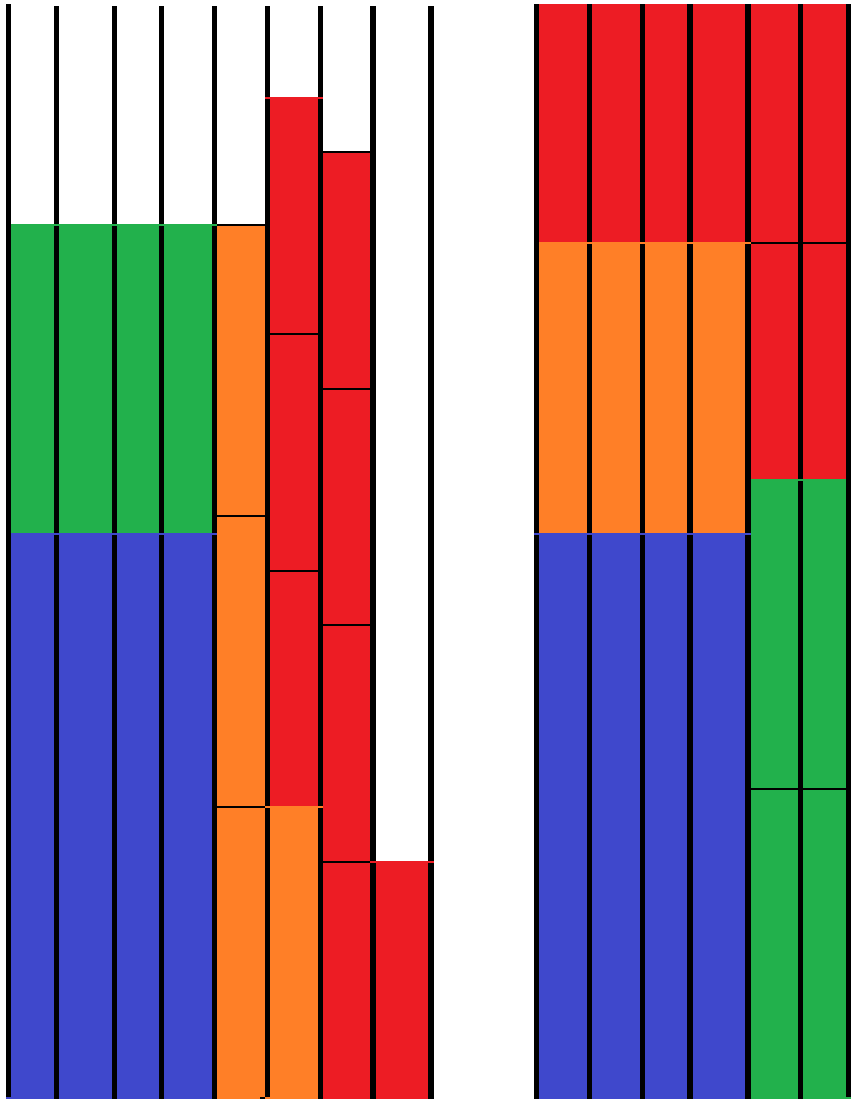
FFD/BFD используют $11k + 8$ ёмкостей
оптимальное количество ёмкостей — $9k + 6$

Задача об упаковке

пример для $k = 0$

слева: решение, выданное жадным алгоритмом FFD/BFD с 8 ёмкостями

справа: оптимальное решение с 6 ёмкостями



Задача об упаковке

Как доказывать гарантии качества алгоритма?

идея: минимальный контрпример

пусть $\{w_1, \ldots, w_n\}$, W — данные задачи, нарушающей оценку, и n **минимально**

тогда либо можно построить пример с меньшим n, либо верно некое свойство

повторными рассуждениями накапливается набор противоречивых свойств

[Yue, Zhang 1995] представили алгоритм с гарантией

$$\hat{N} \leq \left\lfloor \frac{71}{60} N_{opt} \right\rfloor + 1.$$

Задача об упаковке

отдельно рассматриваем объекты с весами в интервалах $(0, \frac{1-w_{\min}}{5}]$, $(\frac{1-w_{\min}}{5}, \frac{1}{3}]$, $(\frac{1}{3}, \frac{1}{2}]$, $(\frac{1}{2}, 1]$, где $W = 1$, $w_{\min} = \min_i w_i$

1. отсортировать объекты по объёмам w_i в убывающем порядке
2. разделить их на классы A, B, C, D согласно вышеуказанным интервалам объёмов
3. для $i = 1, \ldots, |A|$, положить объект i в ёмкость i
4. в цикле по $i = |A|, \ldots, 1$, если это возможно, упаковать два объекта из C в ёмкость i следующим образом:
   1. положить в ёмкость i наименьший оставшийся объект из C
   2. доложить наибольший из оставшихся объектов из C, которые ещё вмещаются
   3. заменить ранее положенный наименьший объект на наибольший из оставшихся объектов из C, которые ещё вмещаются
5. в цикле по оставшимся объектам, каждому объекту сопоставить с соответствие первую из ёмкостей, в которую объект войдёт

Задача об упаковке

формулировка ЦЛП

пусть

- $x_{ij} \in \{0, 1\}$ индицирует, что ёмкость i содержит объект j
- $y_i \in \{0, 1\}$ индицирует, что ёмкость i используется

тогда получаем задачу

$$\min_{X,y} \langle 1, y \rangle : \; X^T 1 = 1, \; Xw \leq Wy,$$

$$X \in \{0, 1\}^{n \times n}, \; y \in \{0, 1\}^n$$

страдает от симметрий, можно добавить неравенства

$$y_i \geq y_{i+1}$$

Задача об упаковке

другая формулировка ЦЛП

пусть w_i, W — целые

объекты можно разделить на не более, чем W весовых классов

имеется конечное число возможностей собрать ёмкость (u_i объектов веса i, $\sum_i iu_i \leq W$)

можно оптимизировать по количеству ёмкостей с определённым шаблоном заполнения

$$\min_{x \in \mathbb{N}^n} \langle 1, x \rangle : \quad Ux = v$$

U_{ij} — количество объектов веса i в шаблоне j
x_j — количество ёмкостей шаблона j
v_i — количество объектов веса i

Задача об упаковке

алгоритм сложности $O(n)$, гарантирующий результат с асимптотической точностью до множителя $1 + \epsilon$
[Fernandez de la Vega, Luecker 1981]

- мелкие объекты рассматриваются только в конце и добавляются в ёмкости к более крупным или в новые ёмкости
- крупные объекты делятся на весовые классы, их вес округляется вверх до границы класса
- так как классов — конечное число, то есть конечное число шаблонов заполнения ёмкости крупными объектами (не зависящее от n)
- субоптимальные количества ёмкостей, заполненные по разным шаблонам, можно найти решением линейной релаксации
- дробные остатки в решении ЛП пакуются отдельно

План

сложные комбинаторные задачи

- задача об упаковке
- **покрытие подмножествами**
- покраска графа
- наибольшая клика

Покрытие подмножествами

пусть U — конечное множество (вселенная), $S \subset 2^U$ — система подмножеств

задача *покрытия подмножествами* состоит в нахождении подсистемы $C \subset S$ подмножеств наименьшей мощности (покрытия) такой, что

$$U = \bigcup_{M \in C} M$$

Покрытие подмножествами

жадный алгоритм

- начинает с $C = \emptyset$
- итеративно добавляет в C подмножество $M \in S$, покрывающее максимальное число ещё не покрытых элементов вселенной U

пусть C_{opt} — наименьшее (оптимальное) покрытие, C — построенное жадным алгоритмом покрытие

можно показать, что $\frac{|C|}{|C_{opt}|} \leq H(\max_{M \in S} |M|)$ где $H(k) = \sum_{j=1}^{k} j^{-1} \approx \log\ k$ — k-ое гармоническое число

работает хорошо, если каждое множество $M \in S$ — маленькое

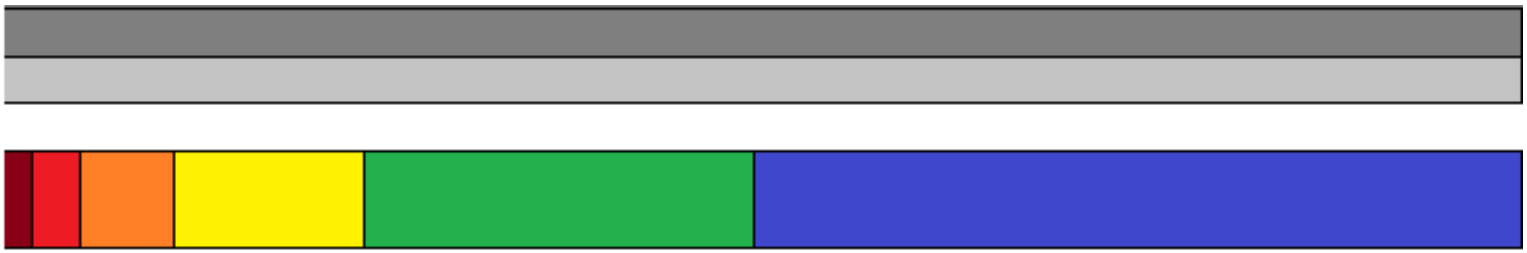
Покрытие подмножествами

пример

- пусть U — множество $2^{k+1} - 2$ точек, расположенных на решётке размера $2 \times (2^k - 1)$
- два подмножества из системы S определяются как строки (мощности $2^k - 1$)
- k подмножеств в S определены как объединения столбцов с индексами $2^j, \ldots, 2^{j+1} - 1$, $j = 0, \ldots, k - 1$

оптимальное покрытие использует два множества, $|C_{opt}| = 2$

жадный алгоритм строит покрытие k подмножествами, $|C| = k$



сверху: оптимальное покрытие

снизу: покрытие, найденное жадным алгоритмом

Покрытие подмножествами

задачу можно сформулировать в виде ЦЛП

пусть $A \in \{0,1\}^{|U| \times |S|}$ — бинарная матрица, определённая по формуле

$$A_{jM} = \begin{cases} 1, & j \in M; \\ 0, & j \notin M \end{cases}$$

тогда задачу можно сформулировать в виде

$$\min_x \langle x, 1 \rangle : \quad Ax \geq 1, \; x \in \{0,1\}^{|S|}$$

здесь x_M — бинарная переменная, индицирующая принадлежность $M \in S$

линейная релаксация имеет вид

$$\min_x \langle x, 1 \rangle : \quad Ax \geq 1, \; x \in [0,1]^{|S|}$$

Покрытие подмножествами

пусть x^* — решение линейной релаксации, $N^* = \langle 1, x^* \rangle$ — нижняя граница на N_{opt}

рандомизированный алгоритм:

- фиксируем $\lambda \geq 1$
- для каждого $M \in S$, добавляем M в покрытие с вероятностью $\min(1, \lambda x^*_M)$
- проверяем, получилось ли покрытие
- выбираем лучшее из найденных покрытий

при $\lambda = \log(2n)$ вероятность того, алгоритм произведёт покрытие со значением $\hat{N} = \langle 1, \hat{x} \rangle$ не большим, чем $2\log(2n) \cdot N^*$, строго положительна

Покрытие подмножествами

- усреднённое значение не превосходит $\lambda \cdot N^*$, и поэтому вероятность события $\hat{N} > 2\lambda N^*$ строго меньше, чем $\frac{1}{2}$
- для любого $k \in U$, вероятность события $(A\hat{x})_k = 0$ равна $\prod_{M:\, k \in M} \max(0, 1 - \lambda x^*_M) < \prod_{M:\, k \in M} e^{-\lambda x^*_M} = e^{-\lambda} = \frac{1}{2n}$

вероятность того, что реализуется по крайней мере одно из событий, строго меньше, чем $\frac{1}{2} + n \cdot \frac{1}{2n} = 1$

алгоритм даёт решение со значением, не превышающим

$$\hat{N} \leq 2\log(2n)N^* \leq 2\log(2n)N_{opt}$$

фактор гарантии качества $\frac{\hat{N}}{N_{opt}} \leq 2\log(2n)$

Покрытие подмножествами

пусть каждый элемент $u \in U$ входит в не более, чем k подмножеств $M \in S$

в каждой строке A не более, чем k единиц

тогда в силу $Ax \geq 1$ в каждую сумму

$$\sum_M A_{jM} x_M$$

есть вклад элемента $x^*_M \geq \frac{1}{k}$

тогда

$$C = \{M \mid x^*_M \geq \frac{1}{k}\}$$

есть покрытие мощности

$$|C| \leq k \cdot \langle 1, x^* \rangle$$

но $|C_{opt}| \geq \langle 1, x^* \rangle$, и

$$\frac{|C|}{|C_{opt}|} \leq k$$

План

сложные комбинаторные задачи

- задача об упаковке
- покрытие подмножествами
- **покраска графа**
- наибольшая клика

Покраска графа

пусть $G = (V, E)$ — ненаправленный граф

Определение

Покраской графа G называют отображение $f : V \to \mathbb{N}_+$ такое, что для любого ребра $(v_i, v_j) \in E$ справедливо $f(v_i) \neq f(v_j)$. Минимальной (или оптимальной) покраской называют покраску f такую, что $f[V] = \{1, \ldots, N\}$ и N при этом — минимально возможное.

значения покраски называют *цветами*
прообразы $f^{-1}(k)$ называют *цветовыми классами*

задача о *минимальной покраске графа* состоит в построении покраски с наименьшим количеством цветов

Определение

Наименьшее количество цветов, необходимое для покраски графа G, называется хроматическим числом графа и обозначается через $\chi(G)$.

Покраска графа

задача возникает в приложениях, в которых нужно разделить некоторое множество на непересекающиеся подмножества, соблюдая ограничения, наложенные на запрет нахождения некоторых пар элементов в обном подмножестве
граф в этом случае называется *конфликтным*

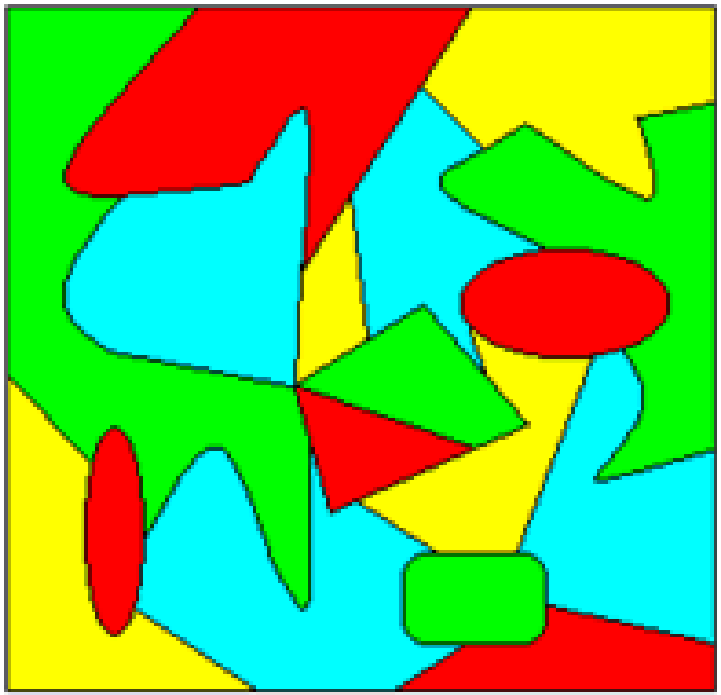
примеры

- расписания экзаменов (студент не может посещать два экзамена одновременно)
- распределение заданий по персоналу (конфликт вызывается пересечением заданий по времени)
- назначение переменных регистрам процессора (конфликт если две переменные используются одновременно)
- расписание соревнований (конфликт если та же команда участвует в двух играх одновременно)

Покраска графа

Теорема (Appel, Haken 1977)

Планарный граф можно покрасить в 4 цвета.



более простое доказательство в [Robertson 1997]

но: покраска планарного графа — NP сложная задача

покраска в 4 цвета может быть неоптимальной

Покраска графа

необходимые условия оптимальности

в оптимальной покраске

- для каждой пары цветов i, j существует ребро, соединяющее вершины цветов i, j (иначе цветовые классы можно объединить)
- для каждого цвета i существует вершина, покрашенная в цвет i, и смежная с вершинами, покрашенными во **все** другие цвета (иначе цветовой класс i можно распределить по другим цветовым классам)

каждый цветовой класс — независимое подмножество

каждая клика покрашена в попарно разные цвета

следствие: $\chi(G) \geq \omega(G)$

Покраска графа

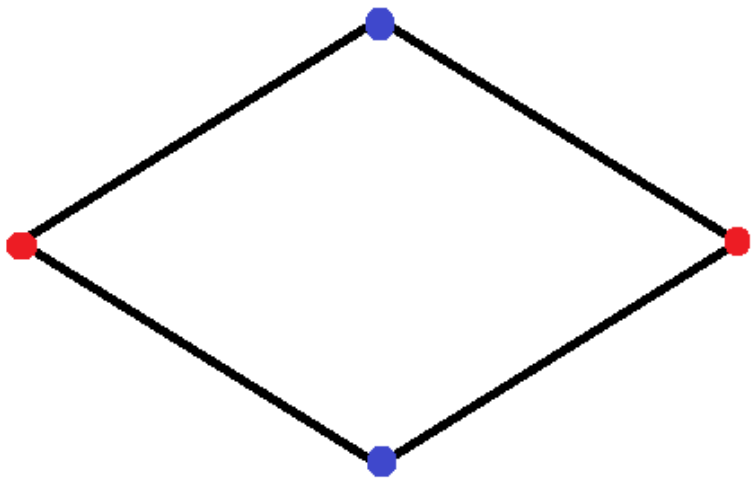
Теорема

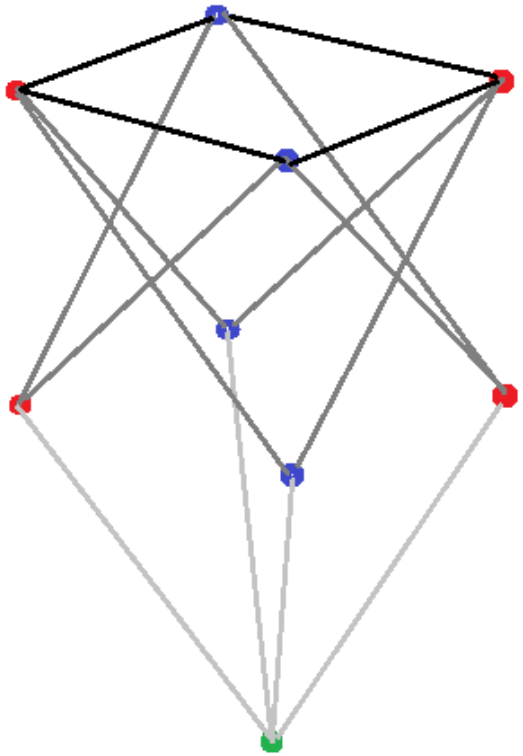
Для каждого $k \geq 2$ существует граф G, для которого

$$\omega(G) = 2, \qquad \chi(G) = k.$$

конструкция Мычельского строит из графа G_{k-1} без треугольников и с оптимальной раскраской в $k-1$ цветов граф G_k без треугольников и с оптимальной раскраской в k цветов

Покраска графа





Покраска графа

степень вершины $v \in V$ в графе $G = (V, E)$ — число смежных с v вершин

пусть $\Delta(G)$ — максимальная степень вершины в графе G

тогда $\chi(G) \leq \Delta(G) + 1$

теорема Брукса

Теорема

Пусть G — граф. Имеем $\chi(G) = \Delta(G) + 1$ тогда и только тогда, когда граф G

- *полный граф*
- *нечётный цикл*

Покраска графа

жадный алгоритм

- упорядочить вершины
- одну за одной красить вершины в цвет с минимальным индексом, не вызывающим конфликт

Лемма

Пусть уже есть покраска f графа. Пусть вершины упорядочены по индексу цвета в этой покраске, $f(v_i) \leq f(v_j)$ для всех $i < j$. Тогда жадный алгоритм, запущенный на данном порядке вершин, построит покраску f' с количеством цветов, не превышающих количество цветов покраски f.

Покраска графа

в частности, если f — оптимальная покраска, то f' также будет оптимальной

Следствие

Существует порядок вершин графа, на котором жадный алгоритм построит оптимальную покраску.

однако, жадный алгоритм может дать неоптимальный результат даже на простых графах (например, двудольных)

Покраска графа

Лемма

Жадный алгоритм, применённый к хордальному графу, вершины которого расположены в совершенном порядке исключения, повёрнутом вспять, выдаст минимальную раскраску. Для такого графа G имеем $\omega(G) = \chi(G)$.

на этапе покраски вершины v

- уже покрашенные вершины, смежные с v, образуют клику C_v
- индекс цвета v не превосходит $|C_v| + 1$
- для некоторого v — $\{v\} \cup C_v$ — наибольшая клика
- нужно не менее $\omega(G)$ цветов

Покраска графа

алгоритм степени насыщения (DSatur)

жадного типа: красит одну вершину за другой

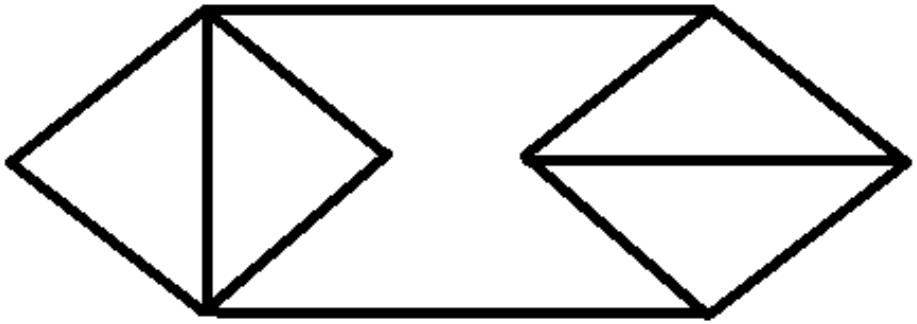
на каждом шаге

- выбирает непокрашенную вершину с максимальным числом цветов среди покрашенных смежных вершин
- если таких несколько, выбирает ту с максимальным числом непокрашенных смежных вершин
- красит выбранную вершину в цвет минимального индекса, не вызывающий конфликт

производит оптимальную покраску на двудольных графах и циклах

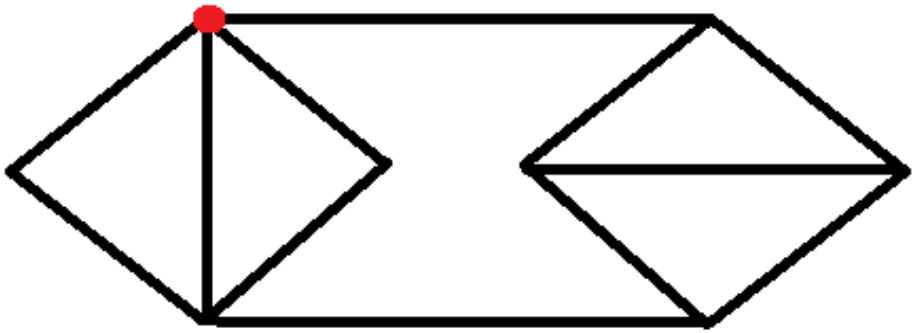
сложность алгоритма — $O(n^2)$

Покраска графа

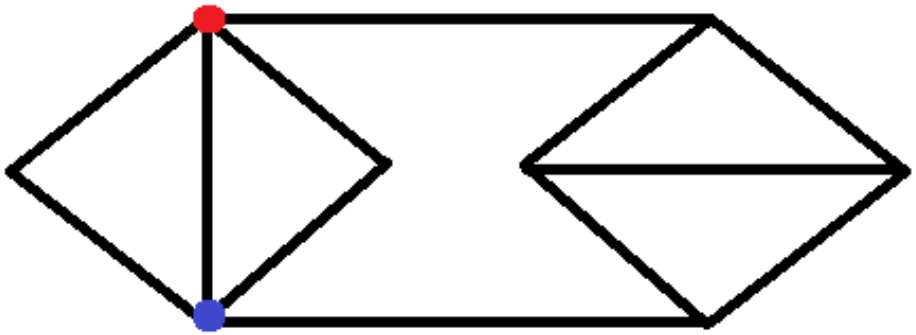


контр-пример: DSatur не находит оптимальную покраску

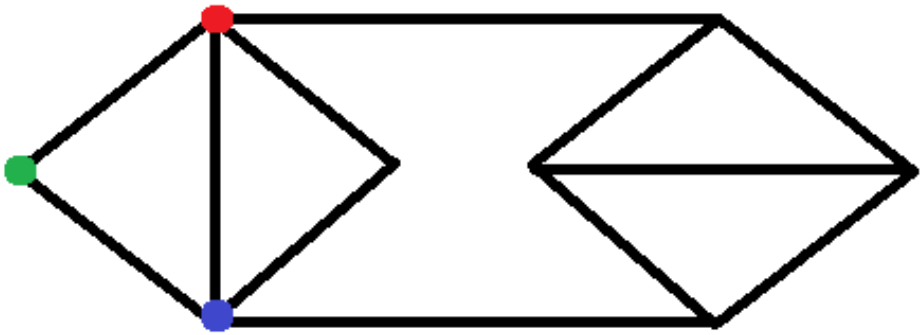
Покраска графа



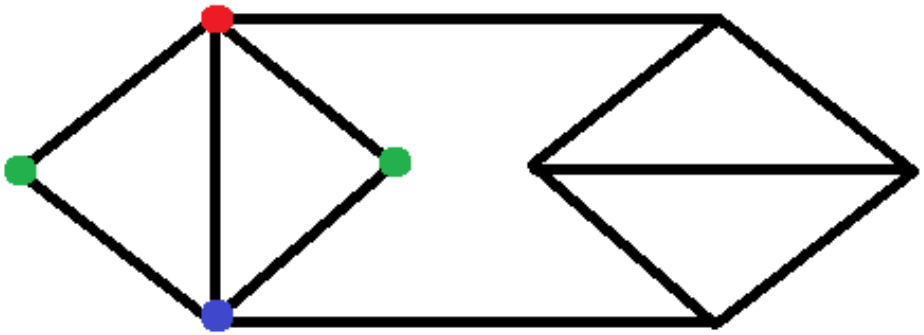
Покраска графа



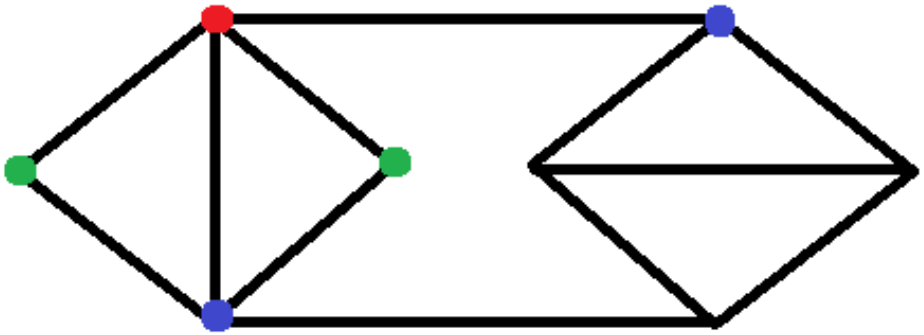
Покраска графа



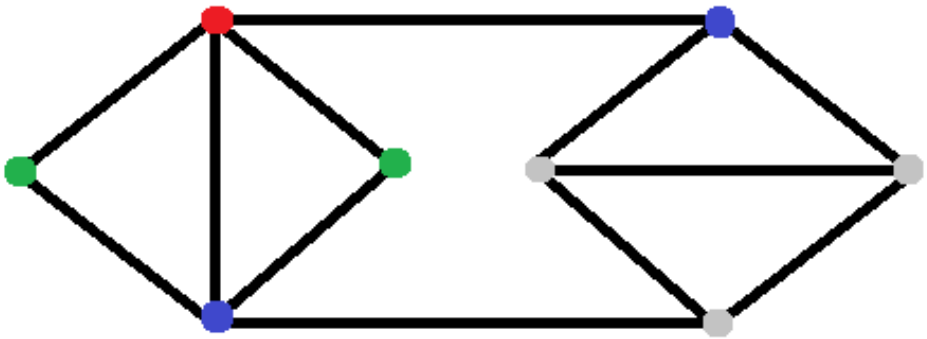
Покраска графа



Покраска графа

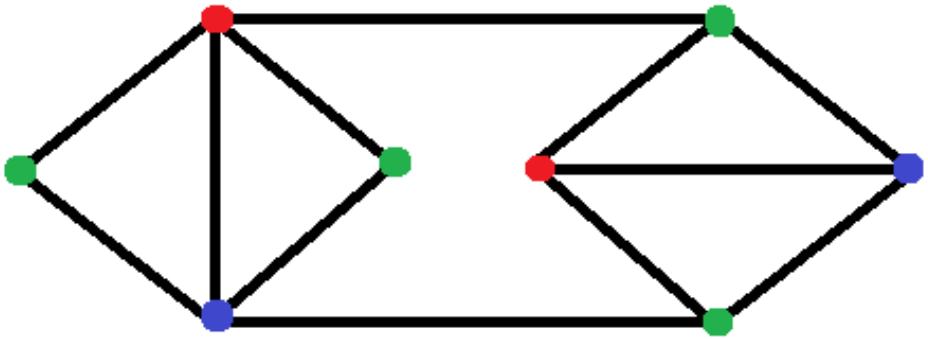


Покраска графа



серые вершины образуют клику, которая не может быть покрашена без привлечения нового цвета

Покраска графа



оптимальная покраска в 3 цвета

Покраска графа

алгоритм RLF (recursive largest first):
на каждом шаге строит цветовой класс S_i, который затем удаляется из графа
построение S_i

- поочерёдное прибавление вершин
- на каждом шаге выбирается вершина v с максимальной степенью насыщения (количество смежных с вершинами из S_i вершин, которые смежны с v)
- если таких несколько, выбирается та с минимальным количеством смежных вершин во всём остаточном графе

выдаёт оптимальный результат на циклах и двудольных графах
сложность $O(n \cdot |E|)$.

Покраска графа

разложение сепарирующими кликами

Лемма

Пусть C — сепарирующая клика в графе $G = (V, E)$, V_i — компоненты связности индуцированного над $V \setminus C$ подграфа. Тогда

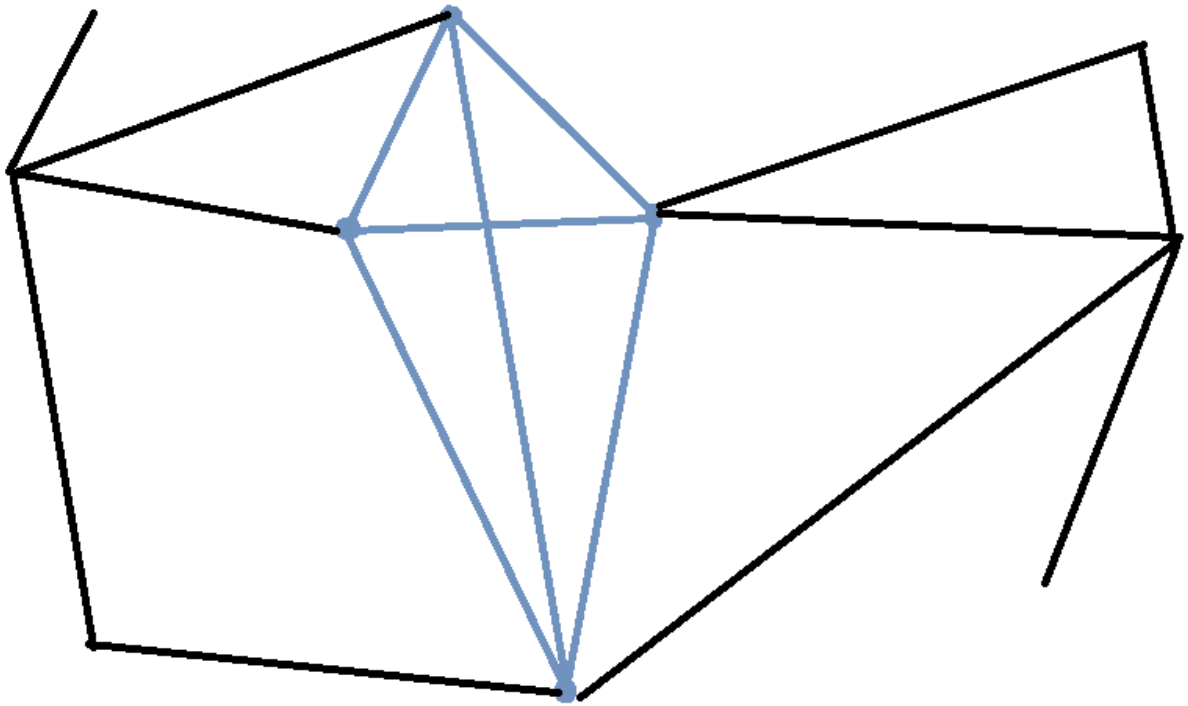
$$\chi(G) = \max_i \chi(V_i \cup C).$$

Более того, если есть покраски подграфов над $V_i \cup C$ с k_i цветами, то из них можно построить покраску G с $\max_i k_i$ цветами.

поэтому задачу покраски G можно свести к задаче покраски компонент $V_i \cup C$

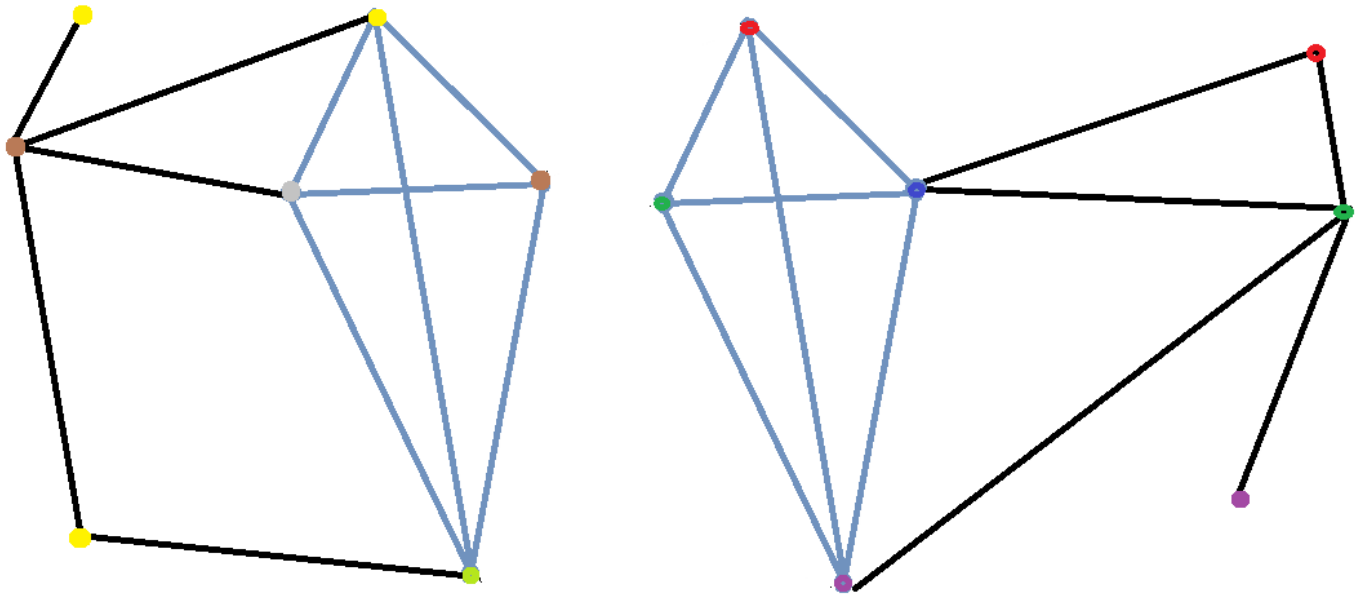
компоненты можно и далее разлагать на подкомпоненты

Покраска графа



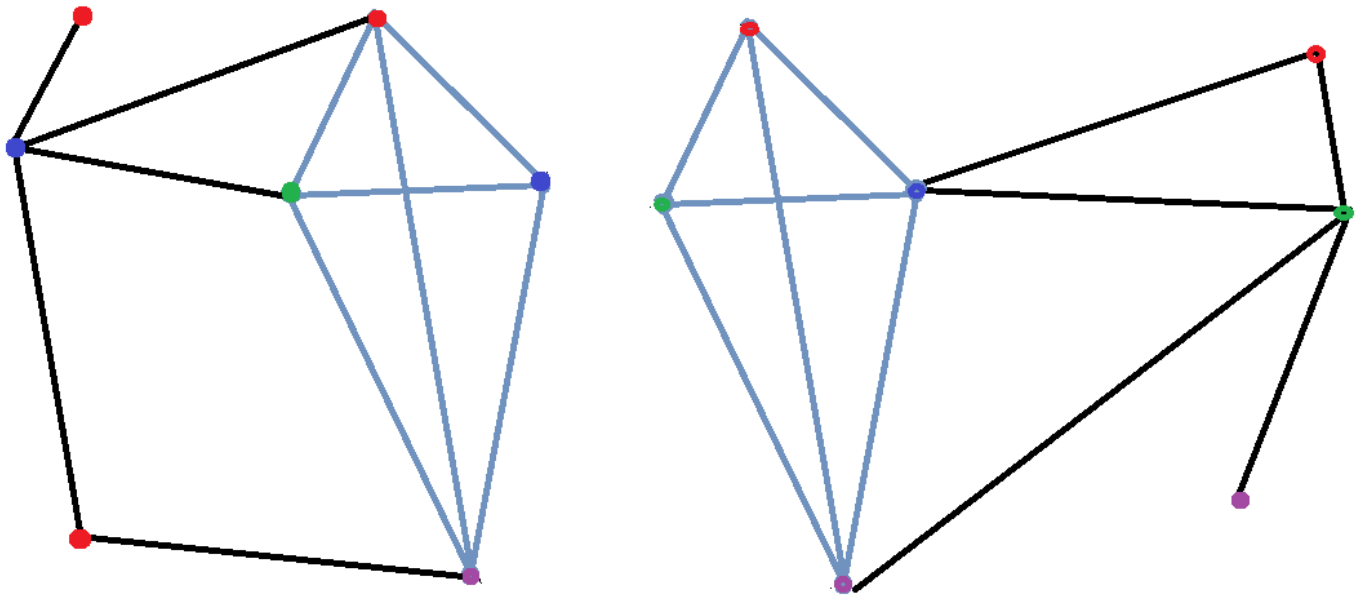
нашли сепарирующую клику

Покраска графа



покрасили компоненты

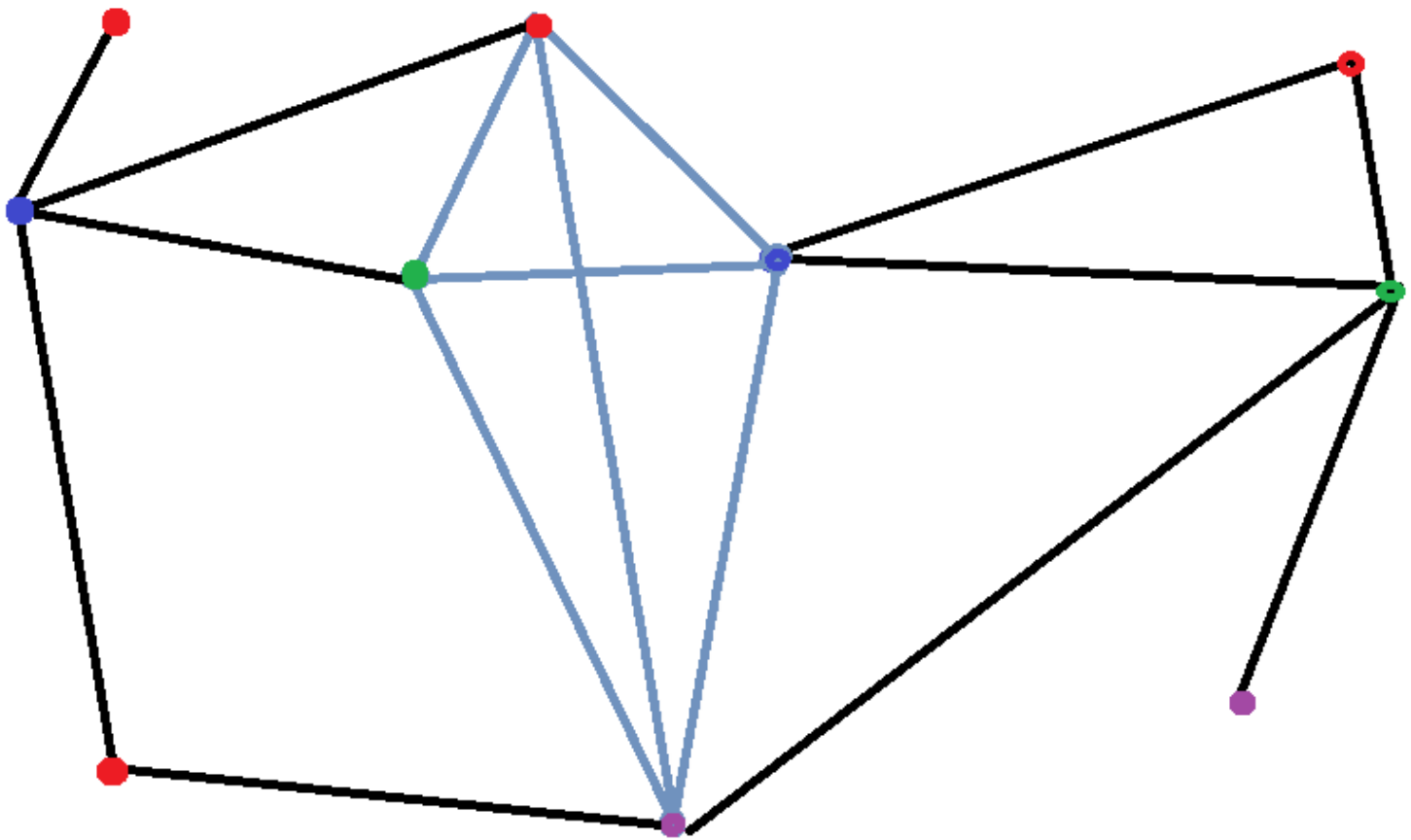
Покраска графа



поменяли цвета в компоненте так, чтобы они на клике совпадали

используем только цвета, встречающиеся в компоненте с максимальным количеством цветов

Покраска графа



объединили покраски компонент

Покраска графа

другие способы упрощения

если известна нижняя граница $\underline{\chi} \leq \chi(G)$, то можно удалить все вершины, смежные с не более, чем $\underline{\chi} - 1$ другими вершинами

пусть $u, v \in V$ такие, что $(u, v) \notin E$ и все смежные с u вершины также смежны с v

тогда $\chi(G) = \chi(G \setminus \{u\})$ — вершину u можно покрасить в тот же цвет, что и v

План

сложные комбинаторные задачи

- задача об упаковке
- покрытие подмножествами
- покраска графа
- **наибольшая клика**

Задача о наибольшей клике

пусть $G = (V, E)$ — ненаправленный граф

Определение

Кликой в G называют подмножество вершин $C \subset V$ такое, что $(v_i, v_j) \in E$ для всех $v_i, v_j \in C$.

задачей о *наибольшей клике* называют проблему нахождения клики максимальной мощности

Определение

Мощность наибольшей клики называется кликовым числом графа G и обозначается через $\omega(G)$.

Задача о наибольшей клике

Определение

Пусть $G = (V, E)$ — нераправленный граф, $n = |V|$. Матрицей инцидентности графа G называют матрицу $A_G \in \mathcal{S}^n$, определённую как

$$(A_G)_{ij} = \begin{cases} 1, & (i,j) \in E \\ 0, & (i,j) \notin E \end{cases}$$

имеем

$$A_G + A_{\bar{G}} + I = 1$$

(матрица из всех единиц, $\bar{G}$ — комплементарный граф)

$$\alpha(\bar{G}) = \omega(G), \qquad \alpha(G) = \omega(\bar{G})$$

α — вершинное число независимости (мощность наибольшего независимого подмножества вершин)

Задача о наибольшей клике

формулировка ЦЛП

$$\max_{x \in \{0,1\}^n} \langle 1, x \rangle : \quad x_i + x_j \leq 1 \ \forall \ (i,j) \notin E$$

$x_i = 1$ если вершина i в клике

Лемма

Оптимальные вершины допустимого полиэдра линейной релаксации имеют только компоненты, равные $0, \frac{1}{2}, 1$.

- пусть $\hat{x} \notin \{0, \frac{1}{2}, 1\}^n$ — оптимальная вершина
- положим $I_- = \{i \mid \hat{x}_i \in (0, \frac{1}{2})\}$, $I_+ = \{i \mid \hat{x}_i \in (\frac{1}{2}, 1)\}$
- пусть $\delta \in \mathbb{R}^n$ задано $\delta_i = \pm 1$ для $i \in I_\pm$, $\delta_i = 0$ иначе
- точки $\hat{x} \pm \epsilon\delta$ допустимы для малых ϵ
- $\hat{x}$ не вершина если $I_- \cup I_+ \neq \emptyset$

Задача о наибольшей клике

Теорема (Nemhauser, Trotter)

Пусть x^* — оптимальная вершина в линейной релаксации. Тогда существует наибольшая клика C такая, что $I_1 = \{v_i \in V \mid x_i^* = 1\} \subset C$ и $\{v_i \in V \mid x_i^* = 0\} \cap C = \emptyset$. Более того, пусть $G_{1/2}$ — индуцированный подграф над множеством $\{v_i \in V \mid x_i^* = \frac{1}{2}\}$, и пусть C' — клика в нём. Тогда $C' \cup I_1$ — клика в G. Если C' — наибольшая клика в $G_{1/2}$, то $C' \cup I_1$ — наибольшая клика в G.

все целые вершины допустимого полиэдра релаксации определяют клики

значение $\langle 1, x^* \rangle$ релаксации — верхняя граница на $\omega(G)$

Задача о наибольшей клике

пусть $C \subset C'$ — клики

положим $\bar{C} = V \setminus C$

- $C' \setminus C$ — клика
- $C' \setminus C$ содержит только вершины из $\bar{C}$, смежные всем $v \in C$, пусть P — множество этих вершин
- $|C' \setminus C| \leq \chi(G_P)$, где G_P — индуцированный над P подграф

отсюда

$$|C| \leq |C'| \leq |C| + \chi(G_P)$$

любая покраска G_P даёт верхнюю границу на $|C'|$

Задача о наибольшей клике

метод ветвей и границ

- узлы дерева поиска индексированы кликами C
- корневой узел — $C = \emptyset$
- для текущей клики C, считаем $P = \{j \notin C \mid (i,j) \in E \ \forall \ i \in C\}$
- красим P $\bar{\chi}$ цветами
- ветвим по вершине в P (присоединяется к C или нет)
- верхняя граница: $|C| + \bar{\chi}$
- нижняя граница: мощность наибольшей доселе найденной клики

отсекаем ветку C если $|\hat{C}| \geq |C| + \bar{\chi}$

Задача о наибольшей клике

ветвление по вершине $v_1 \in P$ приводит к дочерним узлам

- $C \leftarrow C \cup \{v_1\}$, из P удаляются вершины, несмежные с v_1
- $P \leftarrow P \setminus \{v_1\}$

ускорение

- выбирать v_1 из малых цветовых классов
- двигать вершины из малых цветовых классов в большие

Совершенные графы

Определение

Граф $G = (V, E)$ называется совершенным если для любого подмножества $M \subset V$, индуцированный над M подграф G_M удовлетворяет условию $\chi(G_M) = \omega(G_M)$.

Теорема (Chudnovsky 2006)

Граф G — совершенный тогда и только тогда, когда ни G, ни $\bar{G}$ не имеют индуцированный нечётный цикл длины ≥ 5.

следствие: G — совершенный тогда и только тогда, когда $\bar{G}$ — совершенный

Совершенные графы

задачи, простые на совершенных графах:

- покраска графа
- наибольшая клика
- наибольшее независимое подмножество

примеры:

- хордальные графы
- двудольные графы

Задача о наибольшей клике

пусть C — наибольшая клика в хордальном графе

пусть $i \in C$ — вершина с минимальным индексом в порядке совершенного исключения

тогда $C = \{i\} \cup \{j > i \mid (i,j) \in E\}$

нужно только проверить клики данного вида и выбрать наибольшую из них

Задача о наибольшей клике

подход подграфов

- выбрать хордальный индуцированный подграф
- найти наибольшую клику в нём
- расширить подграф, сохраняя его хроматическое число — клика остаётся наибольшей
- использовать найденную клику как начальную точку для локального поиска

как найти максимальный индуцированный хордальный подграф

- запустить лексикографический поиск
- опускать вершины, нарушающие условие порядка совершенного исключения

алгоритм сложности $O(|V| + |E|)$

Задача о наибольшей клике

Лемма (Motzkin-Strauss)

Пусть $G = (V, E)$ — ненаправленный граф. Тогда кликовое число $\omega(G)$ зависит от оптимального значения c^* стандартной квадратичной программы

$$\max_{x \in \Delta} x^T A_G x$$

по формуле $c^* = \frac{\omega(G)-1}{\omega(G)}$. Эквивалентно

$$\omega(G) = \frac{1}{1 - c^*}.$$

Здесь $\Delta = \{x \in \mathbb{R}^n_+ \mid \langle 1, x \rangle = 1\}$ — стандартный симплекс.

Задача о наибольшей клике

- пусть C — наибольшая клика
- положим $x = \frac{1}{\omega(G)} 1_C$
- тогда $x^T A_G x = \frac{\omega(G)(\omega(G)-1)}{\omega(G)^2} = \frac{\omega(G)-1}{\omega(G)}$

с другой стороны, пусть x^* — решение квадратичной программы

- если $x_i x_j > 0$ и $(i,j) \notin E$, то подматрица над $\{i,j\}$ матрицы A_G — нулевая
- цена задачи аффинна по направлению $\delta = e_i - e_j$
- максимум достичается в конце интервала, направленного параллельно δ
- существует решение $\tilde{x}$, носитель которого задаёт клику

Задача о наибольшей клике

Лемма

Пусть r — число отрицательных собственных значений матрицы Q в стандартной квадратичной программе

$$\min_{x \in \Delta} x^T Q x.$$

Тогда оптимальная точка x^* имеет не более $r+1$ ненулевых элементов.

так как ненулевые элементы соответствуют наибольшей клике, получаем оценку

$$\omega(G) \leq r + 1$$

где $r = |\{i \mid \lambda_i(A_G) < 0\}|$