Приближенные алгоритмы и не только

Саютин Дмитрий

Дома, 17 марта 2020 г.

Приближенный Partition: жадное решение

- Есть предметы $\{a_1, a_2, \ldots, a_n\}$, разложить в две коробки так, чтобы суммарный вес был как можно более близок.
- Жадное решение: раскладываем жадно предметы начиная от больших.

Приближенный Partition: LDM

- Пусть нужно разложить предметы $a_1 \leqslant a_2 \leqslant \ldots \leqslant a_{n-1} \leqslant a_n$.
- Если разложим как-то предметы $\{a_1, a_2, \ldots a_{n-2}, a_n - a_{n-1}\}$, то можно восстановить разложение исходных предметов.
- LDM = Largest Difference Method, или алгоритм Кармаркар-Карпа.
- Удобно реализовывать через кучу.

Приближенный Partition: LDM: пример

$$\{1, 5, 6, 8\}$$

$$\{1, 5, 6, 8\}$$

$$\{1, 2, 5\}$$

$$\{1, 2, 5\}$$

$$\{1, 3\}$$

$$\{1, 3\}$$

$$\{2\}$$

Приближенный Partition: LDM

- Если $D = |\sum a_i - \sum b_i|$, а все предметы случайно выбраны в $[0, 1]$, то утверждается, что:
- Обычная жадность: $E(D) = \Theta(\frac{1}{n})$.
- LDM: $E(D) = n^{-\Theta(\log(n))}$.

Приближенный Bin Packing

- Напоминание: есть предметы $x_1, \ldots, x_n \in [0, 1]$, раскладываем в минимальное число рюкзаков вместимости 1.

- Отличить ответ 2 от ответа 3 уже NP-трудно, решим тогда трудную задачу Partition.
- Значит полиномиального 1.5-OPT решения не бывает.

Приближенный Bin Packing: First Fit

- Перебираем предметы в произвольном порядке
- Внутри ищем первый по номеру рюкзак, в который влезет.

- Получили 2-OPT решение.

Приближенный Bin Packing: простое наблюдение

- При анализе приближенных алгоритмов нужно оценивать оптимальных ответ относительно нашего, ну или хотя бы оптимальный ответ снизу.
- Тут это сделать легко:
- $OPT \geqslant \sum x_i$

Приближенный Bin Packing: First Fit

- Перебираем предметы в произвольном порядке
- Внутри ищем первый по номеру рюкзак, в который влезет.

- Получили 2-OPT решение.
- Пусть нам понадобилось m рюкзаков. Тогда первые $m-1$ из них заполнены $> \frac{1}{2}$.
- Тогда $OPT \geqslant \sum a_i > (m-1)\frac{1}{2}$.
- Тогда $m - 1 < 2OPT$, значит $m \leqslant 2OPT$.

Приближенный Bin Packing: FFD и Best Fit

- First Fit Decreasing: то же самое, но в порядке убывания веса предмета.
- Best Fit [Decreasing]: пробуем запихнуть туда, куда влезет, и чтобы места осталось по-меньше.

- Известно, что FFD и BFD это $\frac{11}{9}OPT + \frac{6}{9}$, при чём эта оценка “сильная” (tight),
- Иначе говоря, она достигается на некотором примере.

Приближенный BinPacking: PTAS-схема

Хотим получить что-то вроде $(1+\delta)$-OPT решения.

Сделаем три шага:

- Пусть все $a_i \geqslant \varepsilon$, а различных a_i не более K. Решить точно.
- Отбросим условие про “не более K”. Получим приближенное решение.
- Отбросим условие про “$a_i \geqslant \varepsilon$”. Решение станет ещё немного хуже.

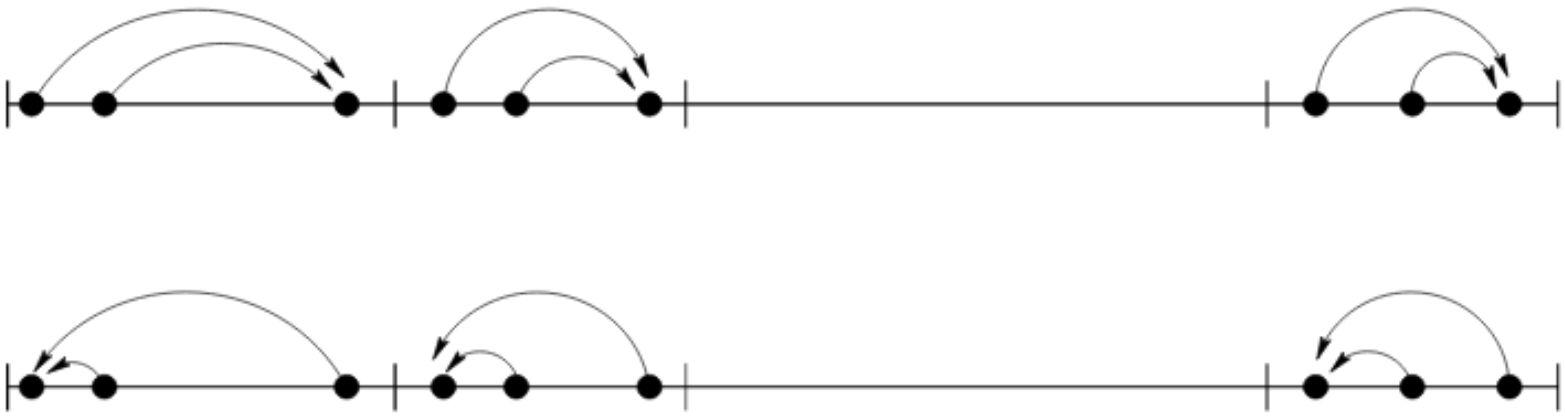
Приближенный BinPacking: PTAS 1/3

- Все $a_i \geqslant \varepsilon$, различных не более K
- Если $a_i \geqslant \varepsilon$, то в один рюкзак влезет $\leqslant 1/\varepsilon =: M$ предметов.
- Если типов a_i не более K, то сколько различных рюкзаков может быть?
- Не более $\binom{M+K}{K} =: R$ (разбиваем на $K+1$ неотрицательных упорядоченных слагаемых).
- Тогда количество способов хоть как-то разложить предметы не более $\binom{n+R-1}{R-1}$ (не более n рюкзаков, на R слагаемых).
- Хорошая новость: ε, K, M, R константы, $\binom{n+R-1}{R-1} = poly(n)$, можно просто перебрать и решить точно.

Приближенный BinPacking: PTAS 2/3

- Все $a_i \geqslant \varepsilon$, ~~различных не более K~~
- Пусть $K := \lfloor \frac{1}{\varepsilon^2} \rfloor$.
- Отсортируем a_i, разобьём на K групп примерно равного размера. В каждой группе будет не более $\lfloor n\varepsilon^2 \rfloor =: Q$ элементов.
- Пусть A исходная задача, в задаче A^+ округлим все предметы вверх (до наибольшего предмета в этой группе), в A^- вниз.

Приближенный BinPacking: PTAS 2.1/3



Сверху A^+, снизу A^-.

Сразу заметим, что $OPT(A^-) \leqslant OPT(A) \leqslant OPT(A^+)$.

Картинка из книги V. Vazirani, Approximation Algorithms, 2004.

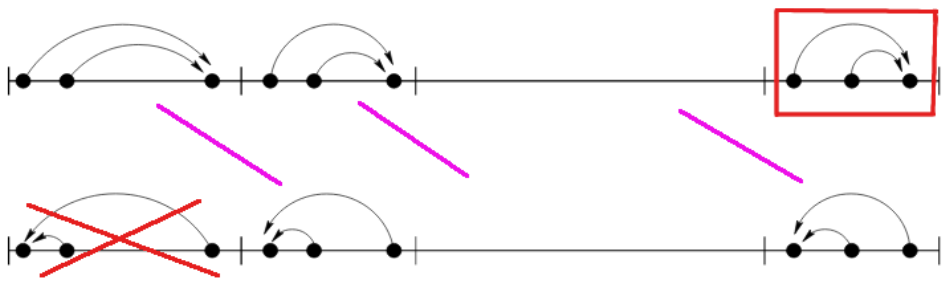
Приближенный BinPacking: PTAS 2.2/3

- Все $a_i \geqslant \varepsilon$, ~~различных не более K~~
- Пусть $K := \lfloor \frac{1}{\varepsilon^2} \rfloor$.
- Отсортируем a_i, разобьём на K групп примерно равного размера. В каждой группе будет не более $\lfloor n\varepsilon^2 \rfloor =: Q$ элементов.
- Пусть A исходная задача, в задаче A^+ округлим все предметы вверх (до наибольшего предмета в этой группе), в A^- вниз.

- *Lm*: $OPT(A^+) \leqslant (1+\varepsilon)OPT(A)$.
- Тогда можно просто решить A^+ как делали раньше и ошибёмся не сильно.

Приближенный BinPacking: PTAS 2.3/3

- Возьмём решение A^- и перекроим его в решение для A^+. Для этого из A^- выкинем Q самых мелких предметы, в A^+ выкинем Q самых больших предметов в отдельные коробки, а оставшиеся предметы сматчим по диагонали.



- Итого $OPT(A^+) \leqslant OPT(A^-) + Q$
- Значит $OPT(A^+) \leqslant OPT(A) + Q$
- Ещё знаем, что $OPT(A) \geqslant n\varepsilon$, значит ошиблись не более чем $Q \leqslant \varepsilon OPT(A)$.

Приближенный BinPacking: PTAS 3/3

- ~~Все $a_i \geqslant \varepsilon$, различных не более K~~
- Пусть у нас есть $(1+\varepsilon)$-OPT решение задачи без предметов $< \varepsilon$.
- Давайте досыпем малые предметы методом First-Fit.
- Если количество бакетов не изменилось, то всё ещё $(1+\varepsilon)$-OPT
- оцениваем OPT просто по OPT решению задачи без малых предметов
- Иначе: во всех корзинах кроме, возможно последней, хотя бы $(1-\varepsilon)$.
- Пусть m число корзин в нашем решении.
- Тогда: $(m-1)(1-\varepsilon) \leqslant \sum a_i \leqslant OPT$, значит $m \leqslant (1-\varepsilon)^{-1} OPT + 1 \leqslant (1+2\varepsilon) OPT + 1$.
- (Примечание: $(1-\varepsilon)^{-1} \leqslant (1+2\varepsilon)$ верно в предположении $\varepsilon \in (0; 0.5]$).

Приближенный BinPacking: PTAS резюме

Алгоритм

- Удаляем предметы $< \varepsilon$.
- Разбиваем предметы на несколько групп, в каждой округляем наверх.
- Перебором за полином от n решаем оставшуюся задачу.
- Досыпаем малые предметы.

- Получаем $(1 + 2\varepsilon)$-OPT $+1$.

Задача о надстроке

- Даны строки $s_1, s_2, \ldots, s_n$.
- Найти самую короткую строку T, такую что все s_i входят как подстроки.
- WLOG ни одна строка не является подстрокой другой.

- **Упражнение:** решить точно за $\mathcal{O}^*(2^n)$.

- Первая попытка решения: будем итеративно находить две строки с самым большим наложением и заменять их на их надстроку.
- Утверждается, что получится 3.5-OPT.

Задача о надстроке: из статьи

Algorithm 1 Greedy Algorithm (GA)

Input: set of strings $\mathcal{S}$.
Output: a superstring for $\mathcal{S}$.

1: **while** $\mathcal{S}$ contains at least two strings **do**
2: extract from $\mathcal{S}$ two strings with the maximum overlap
3: add to $\mathcal{S}$ the shortest superstring of these two strings
4: return the only string from $\mathcal{S}$

Greedy Conjecture. *For any set of strings $\mathcal{S}$, $\mathrm{GA}(\mathcal{S})$ constructs a superstring that is at most twice longer than an optimal one.*

Blum et al. [BJL+91] prove that the Greedy Algorithm returns a 4-approximation of SCS, and Kaplan and Shafrir [KS05] improve this bound to 3.5. A slight modification of the Greedy Algorithm gives a 3-approximation of SCS [BJL+91], and other greedy algorithms are studied from theoretical [BJL+91, RC18] and practical perspectives [RBT04, CJR18].

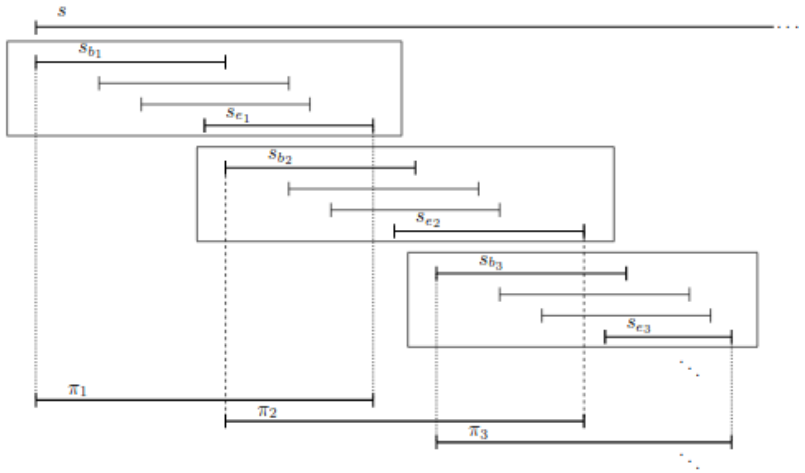
It is known that the Greedy Conjecture holds for the case when all input strings have length at most 3 [TU88, CR18b], and it was recently shown to hold in the case of strings of length 4 [KSS15].

Задача о надстроке: 2-OPT через SetCover

- Пусть $t_{i,j,k}$ это взять s_i, s_j и надвинуть их друг на друга на k.
- abacabab ababc, $k = 2 \rightarrow$ abacababababc

- Построим все такие $t_{i,j,k}$, для каждой найдём всё входящие в неё подстроки, решим взвешенный Set Cover.
- В ответе тупо сконкатенируем всё, что взяли.
- Получится 2-OPT. Или $2ln(n)$-OPT, если решать Set-Cover жадностью.

Задача о надстроке: 2-OPT через SetCover

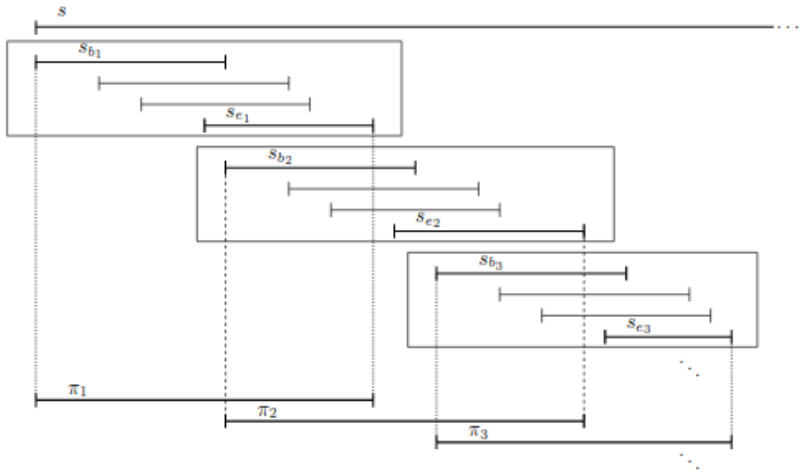


Наша цель получить решение SetCover хуже не более, чем в два раза чем OPT.

Рассмотрим OPT и как различные строки в него входят. Заметим, что эти отрезки не вложены.

Картинка из книги V. Vazirani, Approximation Algorithms, 2004.

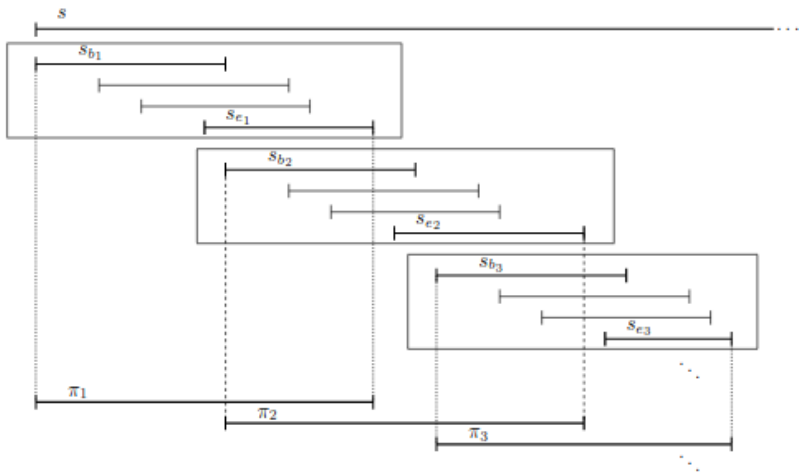
Задача о надстроке: 2-OPT через SetCover



Построим решение Set Cover так: отщепим в первое множество первую строку, а также все, которые с ней пересеклись. Аналогично отщеплять будем и все остальные.

Утверждение: Никакие три выбранных множества не накладываются (на картинке π_1 не накладывается с π_3). Из этого легко следует 2-OPT.

Задача о надстроке: 2-OPT через SetCover



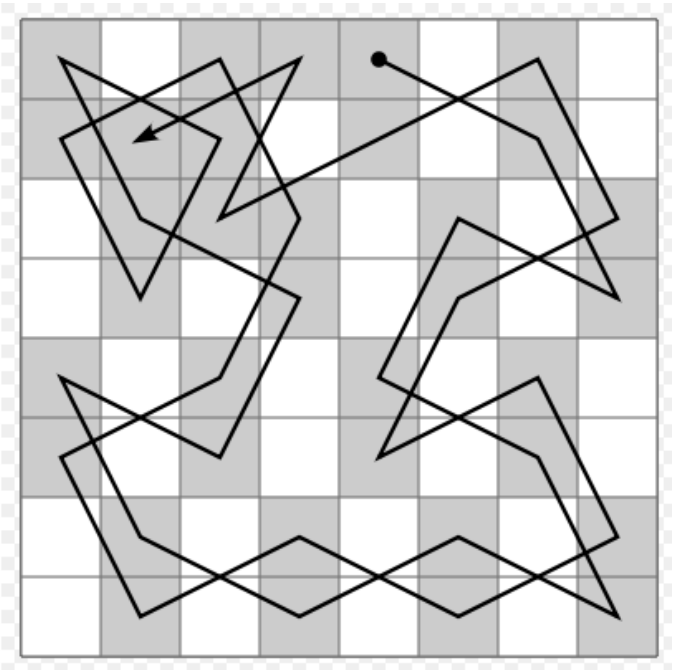
Утверждение: Никакие три выбранных множества не накладываются (на картинке π_1 не накладывается с π_3).

Если π_1 накладывается с π_3, то s_{b_3} пересекается с s_{e_1}, а значит тем более пересекается с s_{b_2}, а значит должна относится к π_2.

Закрываем долги

Гамильтонов путь

- Как бы искать Гамильтонов путь?
- Ну, вообще *сложно*.



- Давайте найдём гамильтонов путь конём.

Гамильтонов ход конём

- Правило Варнсдорфа: идти в вершину минимальной степени.
- Если выбор неоднозначен, можно рандомизировать, решает задачу для обычной доски с вероятностью близкой к 1.

- Применение к произвольному графу: запускать много раз, пробовать делать переход не в самую лучшую вершину, а в k лучших (перебор).
- Разумное отсечение: оставшаяся часть связна.

3-SAT: Random Walk

- Ищем решение 3-SAT
- Что уже было раньше: делаем перебор/random walk от всех нулей **и** всех единиц на глубину $n/2$, $\mathcal{O}(3^{n/2})$.
- Сейчас: получим решение за $\mathcal{O}(1.5^n)$.

- Решение: сгенерируем случайное назначение переменным X_0 и сделаем random walk из него.
- Если сделать n шагов, получится вероятность успеха $(2/3)^n$.

3-SAT: Random Walk

- Решение: сгенерируем случайное назначение переменным X_0 и сделаем random walk из него.
- Если сделать n шагов, получится вероятность успеха $(2/3)^n$.
- Какое-то решение есть. С вероятностью $\binom{n}{k}\frac{1}{2^n}$ мы начнём на расстоянии ровно k от него.
- С вероятностью $\geqslant 3^{-k}$ в него попадём.
- Итоговая вероятность:
  $\sum_k \binom{n}{k}\frac{1}{2^n}\frac{1}{3^k} = 2^{-n}(1+1/3)^n = (2/3)^n$.

Борувка

- Ищем минимальный остов. Выбираем из каждой вершины минимальное ребро, сжимаем.
- Что уже было: оценка $\mathcal{O}((V + E)\log(V))$.
- Сейчас: получим $\mathcal{O}(E\log(V^2/E))$.

Борувка

- Ищем минимальный остов. Выбираем из каждой вершины минимальное ребро, сжимаем.
- Lm. Борувка работает за $\mathcal{O}(E + V^2)$.
- Удаление кратных рёбер работает за $\mathcal{O}(V + E)$.
- После первого удаления кратных рёбер верно: $E \leqslant V^2$. Значит время можно оценить как $\mathcal{O}(V^2 + (V/2)^2 + \ldots) = \mathcal{O}(V^2)$.

- Lm. Борувка работает за $\mathcal{O}(E\lceil\log(V^2/E)\rceil)$
- V^2 уменьшается за фазу хотя бы в 4 раза. Спустя $\log_4(V^2/E)$ фаз количество вершин будет $\leqslant$ исходного числа рёбер. Значит оставшаяся часть за $\mathcal{O}(E)$ отработает.

Конец?

Конец.