Лекция 9

Приближенные алгоритмы для задач о рюкзаке, покрытии множествами, раскраске графа и задачи о коммивояжере в метрическом пространстве

(Конспект: А. Куликов)

9.1 Приближенный алгоритм для задачи о рюкзаке

Определение 9.1 (Задача о рюкзаке). Из заданных предметов нужно выбрать такие, чтобы их суммарный вес был не более W, а стоимость – наибольшей. Точнее – заданы два множества, содержащие по n натуральных чисел: $w_1, \ldots, w_n$ и $p_1, \ldots, p_n$. Необходимо найти такое множество I, содержащееся в $[1..n]$, что $\sum\limits_{i \in I} w_i \le W$, и $\sum\limits_{i \in I} p_i$ максимально. НУО, $\max_{i \in [1..n]} w_i \le W$.

Как известно, в общем случае эта задача является $\mathcal{NP}$-полной. Так что будем строить приближенный алгоритм. Приведем так называемый псевдополиномиальный алгоритм для рюкзака, который будет полиномиальным от длины входа и $\max\limits_{i \in [1..n]} p_i$.

Пусть $S = \sum\limits_{i \in [1..n]} p_i$. Введем функцию w: $w(k, p)$:=минимальный объем, необходимый для того, чтобы уложить предметы с номерами, не превосходящими $k \in [1..n]$, общей стоимостью не менее $p \leq S$ (если такого набора предметов нет, то приравняем функцию $+\infty$).

Вычислять эту функцию будем индуктивно. В цикле по k от 1 до n вычисляем $w(k, p)$ для каждого p от 1 до S следующим образом:

$$w(k_0 + 1, p) = \min\{w(k_0, p), w(k_0, p - p_{k_0+1}) + w_{k_0+1}\}.$$

Ясно, что этот алгоритм работает не более, чем nS, т.е. не более $n^2 \max p_i$.

Таким образом, мы могли бы решить нашу задачу за полиномиальное время, если бы p_i были достаточно маленькими.

Введем обозначения. Пусть задано ϵ, определяющее, с какой точностью мы хотим найти ответ. Обозначим A_ϵ общую стоимость набора предметов, который находится алгоритмом, который мы построим и про который покажем, что он дает $(1-\epsilon)$-приближение; $P = \max p_i$, $K_\epsilon = \frac{P}{(1+1/\epsilon)n}$.

Поделим все p_i на K_ϵ и округлим: $p'_i = \lceil p_i/K_\epsilon \rceil$. Теперь запустим наш псевдополиномиальный алгоритм для полученного набора чисел. Заметим, что $\max p'_i \leq nP(1 + 1/\epsilon)/P = O(n(1 + 1/\epsilon))$. Таким образом, мы потратили время $\mathrm{poly}(n, 1/\epsilon)$.

Заметим теперь следующее неравенство: $A_\epsilon \geq A_0 - K_\epsilon n$, где A_0 — оптимальная стоимость предметов. Действительно, псевдополиномиальный алгоритм находит точное решение, поэтому отклонение в стоимости могло появиться только при округлении. При округлении мы могли потерять не более 1 на каждом предмете, которая домножилось на K_ϵ при обратном переходе от p'_i к p_i; итого, потеряли не более $K_\epsilon n$.

Далее,

$$\frac{A_\epsilon}{A_0} \geq \frac{A_0 - K_\epsilon n}{A_0} = 1 - \frac{Pn}{nA_0(1 + 1/\epsilon)} \geq 1 - \frac{1}{1 + 1/\epsilon} = \frac{1}{\epsilon + 1}.$$

В последнем неравенстве мы воспользовались очевидным фактом: $A_0 \geq P$ (действительно, ведь в рюкзак можно просто положить самый дорогой предмет). Итак, $A_\epsilon \geq (1/(1 + \epsilon))A_0 \geq (1 - \epsilon)A_0$.

9.2 Приближенный алгоритм для покрытия множествами

Пусть $U = \bigcup\limits_{i \in I} S_i$. Необходимо найти $I' \subseteq I$, так чтобы $U = \bigcup_{i \in I'} S_i$ и $|I'|$ было бы минимальным. Для приближенного решения этой задачи используем жадный алгоритм. Пусть мы уже выбрали множества

$S_{i_1}, \ldots, S_{i_k}$. На следующем шаге выбираем то множество, которое покрывает максимальное количество еще не покрытых элементов. Каждому элементу x из U присвоим вес c_x, который вычислим следующим образом: пусть x впервые покрыт нашим алгоритмом на шаге k; вместе с ним на этом шаге покрыто, очевидно, множество $S_{i_k} \setminus \bigcup\limits_{j \le k-1} S_{i_j})$, пусть p_k — мощность этого множества; тогда c_x положим равным $1/p_k$. В результате, на каждом шаге нашего алгоритма мы покрываем множество веса 1.

Лемма 9.1.

$$\sum_{x \in S_i} c_x \le H_{|S_i|} (= 1 + \frac{1}{2} + ... + \frac{1}{|S_i|}).$$

Доказательство. Пусть u_k – количество элементов S_i, покрытых на k-ом шаге. На каждом шаге мы покрываем $u_k - u_{k-1}$ элементов множества S_i, тогда общий покрытый вес на шаге k равен $(u_k - u_{k-1})/p_k$. Просуммируем по всем шагам и оценим

$$\sum_k (u_k - u_{k-1})/p_k \le \sum_k (u_k - u_{k-1})/u_k \le \sum_k (H_{u_k} - H_{u_{k-1}}) = H_{|S_i|}$$

(поясним последнее неравенство: $H_b - H_a = 1/(a+1) + ... + 1/b \ge (b-a)/b$, т.к. в получившейся последовательности ровно $b-a$ дробей, знаменатель каждой из которых не превосходит b; осталось положить $b = u_k$, $a = u_{k-1}$). □

Теперь посчитаем, сколько мы нашли множеств. Для этого достаточно просто просуммировать веса всех элементов:

$$\sum_{x \in U} c_x \le \sum_{i \in I_{opt}} \sum_{x \in S_i} c_x \le \sum_{i \in I_{opt}} H_{|s_i|} \le |I_{opt}| H_{|U|},$$

т.е. приведенный алгоритм является $H_{|U|}$-оптимальным.

9.3 Приближенный алгоритм для раскраски графа

Предположим, что мы хотим покрасить вершины графа правильным образом (то есть так, чтобы концы любого ребра были покрашены в разные цвета).

Замечание 9.1. Ясно, что любой граф можно покрасить в $\delta + 1$ цвет, где δ — максимальная степень вершин этого графа.

Будем теперь рассматривать 3-раскрашиваемый граф. (Задача выяснения, можно ли покрасить граф в три цвета правильным образом, является $\mathcal{NP}$-полной.)

Рассмотрим произвольную вершину A и ее окрестность. В силу того, что рассматриваемый граф является 3-раскрашиваемым, окрестность вершины A можно покрасить в два цвета. Заметим, что покрасить граф в два цвета очень легко (а в нашем случае это, как мы выяснили, возможно): покрасим произвольную вершину в какой-нибудь из цветов; далее всех соседей уже покрашенных вершин будем красить в противоположные цвета (так нужно будет сделать для каждой компоненты связности). Теперь будем поступать следующим образом: если в графе есть вершина степени не менее $\sqrt{n}$, то красим ее окрестность в новые два цвета и выкидываем все эти вершины. Повторив не более $\sqrt{n}$ таких операций мы получим граф, содержащий лишь вершины степени менее $\sqrt{n}$. Его мы покрасим в новые $\sqrt{n}$ цветов (это сделать можно по замечанию, сделанному выше). Итого, мы использовали $3\sqrt{n}$ цветов для раскраски исходного графа.

Теперь зададимся целью покрасить граф в $\delta^{1/3}$ цветов. Назовем граф векторно 3-раскрашиваемым, если каждой его вершине можно сопоставить единичный вектор (из пространства $\mathbb{R}^n$), так что для любого ребра (i, j) будет выполняться равенство $v_i \cdot v_j = -1/2$. Ясно, что любой 3-раскрашиваемый граф является векторно 3-раскрашиваемым (достаточно три цвета заменить на такие три вектора плоскости, что угол между каждыми двумя из них будет $2\pi/3$).

Итак, найдем какой-нибудь набор векторов, удовлетворяющий указанному свойству. Пусть r – случайный вектор пространства $\mathbb{R}^n$ (быть может, не единичный). Обозначим: $U = \{i | r \cdot v_i \geq c\}$ (константу c мы определим позже). Если в этом множестве есть ребра, то выкинем по одной вершине от каждого ребра. Пусть $n' = |U|$, $m' = |\{(i,j) \in E | i, j \in U\}|$.

Найдем теперь $\mathbf{E}(n' - m')$ (число $n' - m'$ будет размером полученного множества; само множество будет независимым (определяли в лекции 9)). $\mathbf{E}n' = n \cdot P\{$вершина попадет в множество $U\} = n \cdot P(c)$, где $P(c) = P\{v_i \cdot r \geq c\} = \int_c^\infty \phi(x)dx$, а $\phi(x) = (1/\sqrt{2\pi})e^{-x^2/2}$ — плотность нормального распределения.

$$\mathbf{E}m' = \sum_{(i,j)\in E} P\{v_i \cdot r \geq c,\ v_j \cdot r \geq c\} \leq \sum_{(i,j)\in E} P\{(v_i + v_j) \cdot r \geq 2c\} = P(2c) \cdot |E|,$$

так как $|v_i + v_j|^2 = (v_i)^2 + (v_j)^2 + 2\, v_i \cdot v_j = 1$.

Итак, $\mathbf{E}(n' - m') \geq P(c) \cdot n - P(2c) \cdot (n\delta)/2 = n(P(c) - (\delta/2)P(2c))$.

Теперь воспользуемся таким фактом:

$$\left(\frac{1}{x}-\frac{1}{x^3}\right)\cdot\phi(x)\le N(x)\le\frac{1}{x}\phi(x).$$

Тогда

$$N(c)/N(2c)\ge((1/c-1/c^3)\cdot e^{-c^2/2})/(1/2c)\cdot e^{-2c^2}\ge 2(1-1/c^2)\cdot e^{3c^2/2}.$$

Хотим, чтобы получившееся число было не меньше, чем δ. Для этого достаточно взять $c=\sqrt{2/3\ln\delta}$ (тогда $c=O(\sqrt{\ln\delta})$). Итого,

$$\mathbf{E}(n'-m')\ge n\cdot N(c)-\frac{1}{2}N(c)=\frac{N(c)n}{2}\ge$$
$$\frac{1}{2}\cdot\frac{1}{\sqrt{2\pi}}\cdot(\frac{1}{c}-\frac{1}{c^3})\cdot e^{-c^2/2}=\omega(n/(\delta^{1/3}\sqrt{\ln\delta})).$$

Таким образом, на каждом шаге мы будем получать независимое множество размера $\omega(n/(\delta^{1/3}\sqrt{\ln\delta}))$. Ясно, что за $O(\delta^{1/3}\sqrt{\ln\delta}\log n)$ шагов мы покрасим весь граф (на каждом шаге мы красим найденное независимое множество в новый цвет и выкидываем покрашенные вершины) в $O(\delta^{1/3}\text{polylog}(n))$ цветов.

Задача 9.1. Пользуясь полученными фактами, показать, что за полиномиальное время граф может быть покрашен в $O(n^{1/4}\text{polylog}(n))$ цветов.

9.4 Приближенный алгоритм для задачи о коммивояжере в метрическом пространстве

Имеется полный граф G с расстояниями, удовлетворяющими неравенству треугольника. Необходимо обойти все вершины графа, вернувшись в начальную и пройдя при этом как можно меньше, то есть найти гамильтонов цикл минимального веса.

Предъявим 2-оптимальный алгоритм. Найдем сначала минимальное остовное дерево T графа G. Теперь продублируем каждое ребро наденного дерева T и в полученном графе найдем эйлеров цикл (цикл, проходящий по всем ребрам графа ровно по одному разу). И наконец, преобразуем эйлеров цикл в гамильтонов следующим образом: вершины в гамильтоновом цикле будут идти в порядке их первого появления в эйлеровом при его обходе с произвольной вершины. (Другими словами, едем по эйлерову циклу и те вершины, в которых мы еще не были, записываем в гамильтонов цикл. Когда же встретим такую вершину, просто ее перепрыгнем.) Видно, что все эти операции могут быть выполнены за полиномиальное (даже, в большинстве случаев, за линейное) время.

Покажем, что предъявленный алгоритм действительно 2-оптимальный. Пусть W_T — вес минимального остовного дерева, W_{opt} — вес оптимального гамильтонова цикла. Ясно, что $W_T \leq W_{opt}$, так как при выкидывании любого ребра из гамильтонова цикла мы получаем остовное дерево. Каждое ребро построенного гамильтонова цикла заменяет какой-то путь эйлерова цикла, длина которого не превосходит длины этого ребра (по неравенству треугольника). Таким образом, вес построенного гамильтонова цикла не превосходит $2W_T$, а значит, не превосходит и $2W_{opt}$, чтд.

Теперь улучшим наш алгоритм до 3/2-оптимального. Вместо того, чтобы дублировать каждое ребро остовного дерева, поступим следующим образом: найдем минимальное паросочетание всех вершин дерева T нечетной степени (ясно, что таких вершин четное количество). Добавив ребра найденного паросочетания в дерево T, получим эйлеров граф (то есть такой, в котором степени всех вершин четны). Найдем в этом графе эйлеров цикл и преобразуем его в гамильтонов (как это сделать, было описано выше).

Задача 9.2. Найти минимальное паросочетание.

Докажем, что получившийся алгоритм является 3/2-оптимальным. Аналогично предыдущему доказательству вес построенного гамильтонова цикла будет не более $W_T + W_P$, где W_P — вес минимального паросочетания вершин нечетной степени дерева T. Остается показать, что $W_P \leq W_{opt}/2$. Пусть A — это множество всех вершин нечетной степени дерева T. Рассмотрим такой гамильтонов цикл множества A: в нем вершины множества A будут идти в такой последовательности, в какой они идут в оптимальном гамильтоновом цикле графа G. Ясно, что его вес будет не более W_{opt}. (Естественно, сам этот цикл мы не строим. Нам важно лишь то, что он существует.) Теперь разобьем множество вершин построенного гамильтонова цикла на четные и нечетные. Ясно, что мы получим два паросочетания, вес одного из которых будет меньше $W_{opt}/2$. Итак, мы показали, что существует паросочетание множества A веса не более $W_{opt}/2$, значит, и вес минимального паросочетания не превосходит $W_{opt}/2$, чтд.