Задача SETCOVER

Андрей Степанов

19 декабря 2015 г.

Формулировка задачи

Сформулируем задачу о покрытии множествами SETCOVER. Пусть дано конечное множество $\mathcal{U}$, называемое вселенной и конечная система его подмножеств $\mathcal{F} = \{S_1, \ldots, S_n\}$. Без ограничения общности можно считать, что $\mathcal{U} = \{1, \ldots, m\}$. В задаче поиска требуется найти минимальное по мощности покрытие $\mathcal{G} \subset \mathcal{F}$, такое что

$$\mathcal{U} = \bigcup_{S \in \mathcal{G}} S$$

В задаче распознавания требуется проверить, если ли такое покрытие мощности $\leq k$.

NP-полнота задачи SETCOVER

Сначала напомним формулировку задачи VERTEXCOVER. Пусть дан неориентированный граф $G = (V, E)$ и число k. Требуется проверить, есть ли в графе вершинное покрытие размера $\leq k$. Вершинным покрытием называется такое $V' \subset V$, что для любого ребра $\{u, v\} \in E$ выполнено, что $u \in V'$ или $v \in V'$. Известно, что данная задача является **NP**-полной [1].

Теорема 1. *Задача* SETCOVER *является* **NP***-полной.*

Доказательство. Докажем сначала **NP**-трудность задачи SETCOVER, сведя к ней **NP**-полную задачу VERTEXCOVER.

Пусть дан граф $G = (V, E)$ и число k. Пусть $|V| = n, |E| = m$. Положим $\mathcal{U} = E$, $\mathcal{F} = \{A_v : v \in V\}$, где $A_v = \{e \in E : v \in e\}$ – множество рёбер, покрываемых вершиной v. Ясно, что такая сводимость является полиномиальной, чтобы построить множество A_v, достаточно пройтись по всем рёбрам графа, которых m. Значит, чтобы построить $\mathcal{F}$ нужно $O(nm)$ времени, а на построение $\mathcal{U}$ нужно $O(m)$ времени. Суммарно это $O(|G|)$. Докажем, что эта сводимость корректна.

[1]Это показано, например в [1]

Пусть в графе G нашлось вершинное покрытие $V', |V'| \leq k$. Тогда $\mathcal{G} = \{A_v : v \in V'\}$ будет покрытием множества $\mathcal{U}$ (по определению вершинного покрытия V') мощности $\leq k$ (так как $|\mathcal{G}| = |V'| \leq k$).

Наоборот, пусть нашлось покрытие $\mathcal{G} = \{A_v : v \in V'\}$ множества $\mathcal{U}$ размера $\leq k$. Тогда $|V'| = |\mathcal{G}| \leq k$ и вершины из V' покрывают все рёбра графа G по определению покрытия $\mathcal{G}$.

Итак, VERTEXCOVER $\leq_p$ SETCOVER. Осталось показать, что SETCOVER $\in$ **NP**. Действительно, для задачи SETCOVER легко построить верификатор $V(x, s)$. Сертификатом будет служить само покрытие. Действительно, покрытие является подмножеством входа программы, то есть сертификат имеет полиномиальную длину от размера входа. Задачей верификатора будет проверить, действительно ли s является покрытием вселенной $\mathcal{U}$. Для этого нужно для каждого элемента вселенной проверить, лежит ли он в каком-либо множестве в покрытии. Это можно сделать за полиномиальное время. □

Жадный алгоритм и $\log n$ приближение

Опишем жадный алгоритм GREEDY решения задачи о покрытии. В каждый момент времени будем выбирать множество $A \in \mathcal{F}$, которое покрывает как можно больше ещё не покрытых элементов в множестве $\mathcal{U}$, и добавлять множество A в покрытие. Поскольку этот алгоритм пытается максимизировать покрытие только на данном шаге, он называется "жадным". Так как "не смотрит в будущее".

Приведём реализацию этого алгоритма на языке Python:

```python
def greedy(universe, sets)
    set_cover = []
    while (universe):
        best_set = []
        best_set_coverage = 0
        for s in sets:
            coverage = 0
            for elem in s:
                if (elem in universe):
                    coverage = coverage + 1
            if (coverage >= best_set_coverage):
                best_set = s
        if (best_set == []):
            raise RuntimeError("no solution")
        for elem in best_set:
            if (elem in universe):
                universe.remove(elem)
        sets.remove(best_set)
        set_cover.append(best_set)
    return set_cover
```

Оказывается, жадный алгоритм даёт $\log n$-приближение, в смысле, который сформулирован в следующей теореме.

Теорема 2. *Пусть OPT – оптимальный ответ для задачи* SETCOVER *на входе $(\mathcal{U}, \mathcal{F})$. Обозначим $n = |\mathcal{U}|$. Тогда алгоритм GREEDY запущенный на входе $(\mathcal{U}, \mathcal{F})$ выдаст ответ, по размеру не превосходящий $\log n * |OPT|$.*

Доказательство. Пусть $\mathcal{A} = \{A_1, \ldots, A_k\} \subset \mathcal{F}$ – ответ, выданный алгоритмом GREEDY, причём множества A_i пронумерованы в том порядке, в котором их выбирал алгоритм. Так как оптимальное решение использует $|OPT|$ множеств, то по принципу Дирихле, есть множество в $\mathcal{F}$, которое покрывает хотя бы $\frac{n}{|OPT|}$ точек вселенной $\mathcal{U}$. Поскольку жадный алгоритм выбирает каждый раз множество, которое покрывает как можно больше точек, то A_1 покрывает хотя бы $\frac{n}{|OPT|}$. Значит ещё не покрытых точек осталось не больше $n\Big(1 - \frac{1}{|OPT|}\Big)$. Далее, либо $A_1 \notin OPT$, либо $A_1 \in OPT$. В первом случае в OPT, а следовательно и в $\mathcal{F}$ всё ещё найдется по принципу Дирихле множество, покрывающее $\frac{1}{|OPT|}$-долю оставшихся точек. Во втором случае по тому же самому принципу Дирихле найдется множество в $OPT \setminus \{A_1\}$ покрывающее $\frac{1}{|OPT|-1}$-долю оставшихся точек, а значит и $\frac{1}{|OPT|}$-долю. В любом случае, в силу природы жадного алгоритма, A_2 покрывает хотя бы $\frac{1}{|OPT|}\Big(n - \frac{1}{|OPT}\Big)$ точек. Значит осталось $n\Big(1 - \frac{1}{|OPT|}\Big)^2$ точек. Тогда после $|OPT| \log n$ шагов по индукции получаем, что осталось

$$n\Big(1 - \frac{1}{|OPT|}\Big)^{|OPT|\log n} < n\Big(\frac{1}{e}\Big)^{\log n} = 1$$

точек, а значит, к этому времени алгоритм завершился и $k < |OPT| \log n$. □

Частные случаи и k-приближение

Рассмотрим следующий частный случай задачи SETCOVER. Пусть каждый элемент вселенной $\mathcal{U}$ присутствует не более чем в k подмножествах семейства $\mathcal{F}$. Оказывается, в этом случае можно построить полиномиальный алгоритм, дающий k-приближение, сведя задачу к задаче о линейном программировании (ЛП)

Действительно, сначала сведем задачу к задаче о целочисленном линейном программировании (ЦЛП). Пусть x_i – индикатор того, что S_i множество входит в покрытие. Тогда задача SETCOVER формулируется как задача ЦЛП в следующем виде

$$\begin{cases} \min x_1 + \ldots + x_n \\ \forall i : x_i \in \{0, 1\} \\ \forall u \in \mathcal{U} : \sum\limits_{j : u \in S_j} x_j \geq 1 \end{cases} \tag{1}$$

Заметим, что в каждой сумме

$$\Sigma_u = \sum_{j:u \in S_j} x_j$$

не более чем k слагаемых. Значит, если $\Sigma_u \geq 1$, то найдется $x_j \geq 1/k$, даже если отсутствуют ограничения на целочисленность переменных x_j. Это даёт нам право сформулировать следующей алгоритм LPSETCOVER.

Решшим задачу ЛП 1 без ограничения на целочисленность за полиномиальное время [2], то есть ту, где все $x_i \geq 0$. В каждой сумме есть хотя бы один $x_j >= 1/k$, округлим все такие x_j до 1, а все остальные положим равными 0. Понятно, что полученный набор $(x_1, \ldots, x_n)$ задает покрытие множества $\mathcal{U}$ в силу вышенаписанного утверждения.

Осталось понять, что в процессе округления мы каждый x_j из решения ЛП умножили на некоторое число, не превосходящее k, чтобы получить 0 или 1. Значит, верно следующее

$$|LPSETCOVER| \leq k * |LPOPT| \leq k * |ILPOPT| = k * |OPT|$$

, где $LPSETCOVER$ – ответ, выданный алгоритмом, $LPOPT$, $ILPOPT$, OPT – оптимальные ответы для задач линейного программирования, целочисленного линейного программирование и задачи SETCOVER соответственно. В

Список литературы

[1] D. Musatov *Lecture notes on computational complexity*
`http://ru.discrete-mathematics.org/fall2015/3/complexity/lecture-3-4-np-complete.pdf`

[2] Robert J. Vanderbei, Marc S. Meketon, Barry A. Freedman *A Modification of Karmarkar's Linear Programming Algorithm*
`http://www.princeton.edu/ rvdb/tex/myPapers/VanderbeiMeketonFreedman.pdf`

[3] Avrim Blum *Lecture notes on computer science*
`http://www.cs.cmu.edu/ avrim/451f12/lectures/lect1106.pdf`

[2] Алгоритм приведён в [2]