Задача о вершинном покрытии взвешенного графа

Кириленко Елена, 594

1 Формулировка

Дан взвешенный граф $G = (V, E)$ с весовой функцией $w : V \to R_+$. Нужно найти подмножество вершин, которое будет являтся вершинным покрытием, притом минимального веса. Другими словами, необходимо найти такое $U \subset V$: $\forall (u, v) \in E$: либо $u \in U$, либо $v \in U$, и его суммарный вес $\sum_{u \in U} w(u)$ - минимален.
Заметим также, что данная задача является NP-полной так как к ней можно свести задачу об обычном вершинном покрытии (решить с помощью нее), а задача о взвешенном покрытии, как известно, является NP-полной. Таким образом, решить задачу о вершинном покрытии взвешенного графа за приемлемое время не получится. Наша задача же найти алгоритм дающий 2-приближение, работающий за полиномиальное время.

2 Определения

- Дан граф $G = (V, E)$. Его вершинное покрытие U- множество вершин, которое покрывает все ребра, то есть $\forall (u, v) \in E$: либо $u \in U$, либо $v \in U$.
- Вес множества $w(U) = \sum_{u \in U} w(u)$.
- Пусть U оптимальное решение задачи взвешенного вершинного покрытия. Тогда U' является 2-приближением если U' - вершинное покрытие и $w(U') \leq 2w(U)$

3 Описание алгоритмов

Будем рассматривать два похожих алгоритма.

3.1 Алгоритм 1

1) Если не существует такого ребра $e = (u, v) \in E$, что $w(u) > 0$ и $w(v) > 0$ то просто возьмем все вершины с нулевым весом. Такое множество будет

является вершинным покрытием, так как в данном случае у каждого ребра хотя бы одна вершина имеет нулевой вес. При этом вес полученного множества минимален и равен 0.
2) Иначе возьмем ребро $e = (u, v) \in E$ такое, что $w(u) > 0$ и $w(u) > 0$
Рассмотрим функцию δ:
$\delta(u) = min(w(u), w(v))$
$\delta(v) = min(w(u), w(v))$
$\delta(x) = 0,\ if\ x \notin e$
Теперь запустим рекурсивно алгоритм для исходного графа G с новой весовой функцией $w' = w - \delta$

3.2 Алгоритм 2

1) Добавим все вершины нулевого веса (если такие есть) в покрытие и удалим эти вершины из графа вместе с инцидентными ребрами.
2) Если в графе нет ребер, то есть все вершины изолированы, то вернем пустое покрытие.
3) Иначе, найдем максимальное $c > 0$ такое, что $\forall v \in V : w(v) \geq c * deg(v)$
Рассмотрим функцию δ
$\delta(v) = c * deg(v)$
Теперь запустим рекурсивно алгоритм для графа G с новой весовой функцией $w' = w - \delta$

4 Обоснование алгоритма 1

Утверждение 1

Пусть $A_{w-\delta}$ - 2-приближение задачи для графа G с весовой функцией $w - \delta$
Тогда $A_{w-\delta}$ также является 2-приближением задачи для графа G с весовой функцией w

Доказательство

Пусть $U_w, U_\delta, U_{w-\delta}$ - оптимальное решение задачи для графа G с весовыми функциями $w, \delta, w - \delta$ соответственно.

Возьмем вершины u и v из определения функции δ.
Заметим, что V/v является вершинным покрытием и $\delta(U_\delta) \leq \delta(V/v)$ так как U_δ - оптимальное вершинное покрытие для δ
$\Rightarrow \delta(U_\delta) \leq \delta(V/v) = min(w(u), w(v))$
Также заметим, что $\forall B \subset V :\ \delta(B) \leq \delta(V) = 2min(w(u), w(v)) = 2\delta(U_\delta)$, так как при использовании весовой фукции δ только 2 вершины имеют

ненулевой вес равный $min(w(u), w(v))$.

Теперь посмотрим на $A_{w-\delta}$:
Из условия следует: $(w-\delta)(A_{w-\delta}) \leq 2(w-\delta)(U_{w-\delta})$
$\delta(A_{w-\delta}) \leq 2\delta(U_\delta)$ - из ранее доказанного.
Также заметим, что $\delta(U_\delta) \leq \delta(U_w)$, так как U_δ - оптимальное решение для весовой функции δ
Аналогично $(w-\delta)(U_{w-\delta}) \leq (w-\delta)(U_w)$

Тогда получим:
$w(A_{w-\delta}) = (w-\delta+\delta)(A_{w-\delta}) = (w-\delta)(A_{w-\delta}) + \delta(A_{w-\delta}) \leq 2(w-\delta)(U_{w-\delta}) + 2\delta(U_\delta) \leq 2(w-\delta)(U_w) + 2\delta(U_w) = 2w(U_w)$
Таким образом : $w(A_{w-\delta}) \leq 2w(U_w)$, то есть $A_{w-\delta}$ - 2-приближение задачи для графа G с весовой функцией w.

Этим утверждением мы свели задачу к поиску 2-приближения для весовой функции $w-\delta$. Но у этой задачи суммарный вес всех вершин строго меньше, так как $\delta(u) = \delta(v) > 0 \Rightarrow (w-\delta)(u) < w(u)$ и $(w-\delta)(v) < w(v)$. Значит рано или поздно задача сведется к поиску 2-приближения для весовой функции, где у каждого ребра есть вершина веса 0. В этом случае мы уже не выбираем функцию δ , а решаем полученную задачу алгоритмом без рекурсивного перехода из пункта 1.
Таким образом, алгоритм рано или поздно завершится и мы в итоге получим 2-приблежение. То есть корректность алгоритма доказана.

5 Обоснование алгоритма 2

Утверждение 1

Пусть U - некоторое вершинное покрытие.
Тогда $c|E| \leq \delta(U) \leq 2c|E|$

Доказательство

Так как U - вершинное покрытие, то для каждого ребра хотя бы одна его вершина принадлежит $U \Rightarrow \sum_{v \in U} deg(v) \geq |E|$
Тогда $\delta(U) = \sum_{v \in U} \delta(v) = \sum_{v \in U} c * deg(v) \geq c|E|$
Заметим, что $\delta(V) = \sum_{v \in V} \delta(v) = \sum_{v \in V} c * deg(v) = c \sum_{v \in V} deg(v) = 2c|E|$
Так как $U \subset V$, то $\delta(U) \leq \delta(V) = 2c|E|$
Таким образом, $c|E| \leq \delta(U) \leq 2c|E|$

Утверждение 2

Пусть $A_{w-\delta}$ - 2-приближение задачи для графа G с весовой функцией $w-\delta$
Тогда $A_{w-\delta}$ также является 2-приближением задачи для графа G с весовой функцией w

Доказательство

Пусть $U_w, U_\delta, U_{w-\delta}$ - оптимальное решение задачи для графа G с весовыми функциями $w, \delta, w-\delta$ соответственно.

Из предыдущего утверждения имеем : $c|E| \leq \delta(U_\delta)$.
Так как U_δ - оптимальное вершинное покрытие для δ, то $\delta(U_\delta) \leq \delta(A_{w-\delta})$
Также из предыдущего утверждения имеем $\delta(A_{w-\delta}) \leq 2c|E|$
То есть получаем, что $\delta(A_{w-\delta}) \leq 2c|E| \leq 2\delta(U_\delta)$

Также заметим, что $(w-\delta)(A_{w-\delta}) \leq 2(w-\delta)(U_{w-\delta})$, так как $A_{w-\delta}$ - 2-приближение для весовой функции δ
$(w-\delta)(U_{w-\delta}) \leq (w-\delta)(U_w)$, так как $U_{w-\delta}$ - оптимальное вершинное покрытие для $w-\delta$
Аналогично $(\delta)(U_\delta) \leq (\delta)(U_w)$, так как U_δ - оптимальное вершинное покрытие для δ

Тогда получим
$w(A_{w-\delta}) = (w-\delta+\delta)(A_{w-\delta}) = (w-\delta)(A_{w-\delta}) + \delta(A_{w-\delta}) \leq 2(w-\delta)(U_{w-\delta}) + 2\delta(U_\delta) \leq 2(w-\delta)(U_w) + 2\delta(U_w) = 2w(U_w)$
Таким образом, $w(A_{w-\delta}) \leq 2w(U_w)$

Как и в первом алгоритме, этим утверждением мы свели задачу к поиску 2-приближения для весовой функции $w-\delta$. Но у этой задачи строго меньше вершин и ребер, так как на каждой итерации вес хотя бы одной вершины зануляется и мы добавляем ее в вершинное покрытие и удаляем из графа со всеми инцидентными ребрами. Значит рано или поздно задача сведется к поиску 2-приближения для графа без вершин или ребер. В этом случае мы уже не выбираем функцию δ , а решаем полученную задачу алгоритмом без рекурсивного перехода.
Таким образом, алгоритм рано или поздно завершится и мы в итоге получим 2-приблежение. То есть корректность алгоритма доказана.

6 Имплементация алгоритма 1

```
#принимает на вход массив весов вершин и массив пар ребер
def algorithm_1(weights, edges):
    for (u, v) in edges:
        delta = min(weights[u], weights[v])
        if delta > 0:
            weights[u] -= delta
            weights[v] -= delta
            return algorithm_1(weights, edges)
    return np.arange(len(weights))[weights == 0]
```

7 Имплементация алгоритма 2

```
#принимает на вход массив весов вершин и матрицу смежности
def algorithm_2(weights, adj_matrix):
    eps = 1e-5

    #для вершин с нулевым весом удаляем все инцендентные ей ребра
    for i, w in enumerate(weights):
        if w <= eps:
            weights[i] = 0
            adj_matrix[i, :] = np.zeros_like(adj_matrix[i, :])
            adj_matrix[:, i] = np.zeros_like(adj_matrix[:, i])

    #если не осталось ни одного ребра,то завершаем алгоритм
    #вершинным покрытием будут вершины, получившие вес 0
    if np.sum(adj_matrix) == 0:
        return np.arange(len(weights))[weights == 0]

    #иначе находим с такое, что для любой вершины v: w[v] >= c deg[v]
    else:
        c = max(weights)
        for i, w in enumerate(weights):
            deg = sum(adj_matrix[i])
            if w > 0 and deg > 0 and deg * c > 1.0 * w:
                c = 1.0 * w / deg
        #изменяем веса
        for i in range(len(weights)):
            deg = sum(adj_matrix[i])
            weights[i] -= c * deg

        return algorithm_2(weights, adj_matrix)
```

8 Оценка времени работы

Покажем, что приведенные выше алгоритмы действительно работают ха полиномиальное время.

Алгоритм 1

Цикл for в худшем случае проходит по всем ребрам, что O(|E|) операций. Рекурсивных запусков не больше чем число вершин, то есть O(|V|), так как

после каждого запуска число вершин веса 0 строго увеличивается.
Значит в итоге время работы равно O(|V||E|). Так что первый алгоритм работает за полиномиальное время.

Алгоритм 2

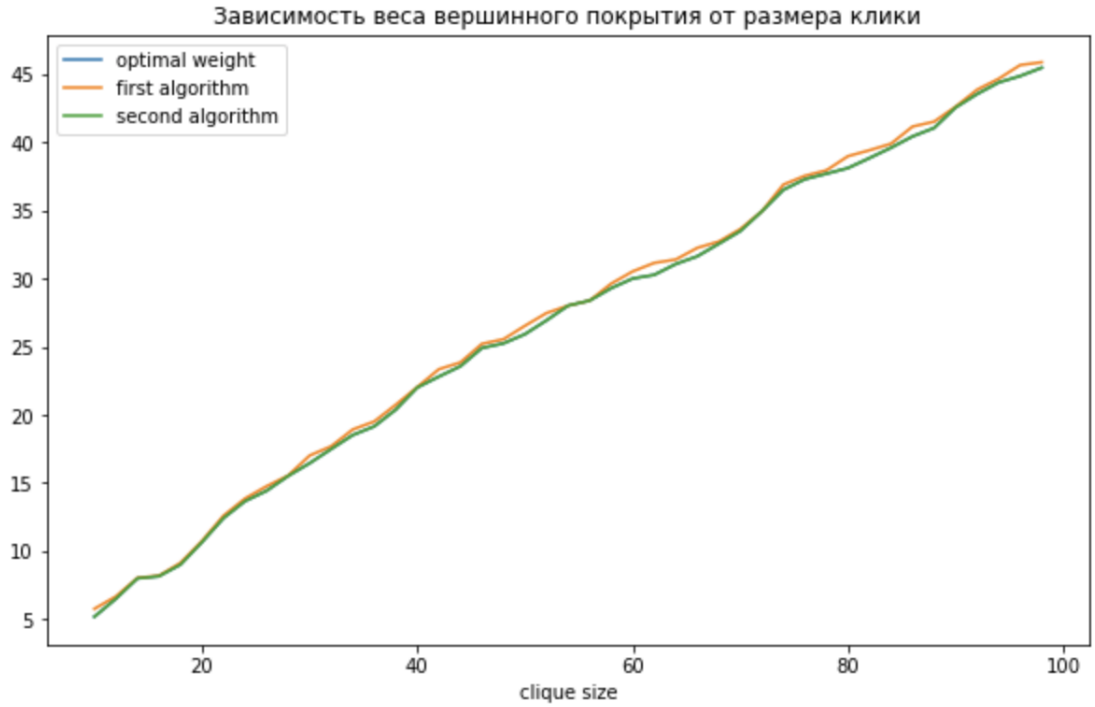
После каждого рекурсивного запуска число вершин нулевого веса строго увеличивается. Значит рекурсивных запусков O(|V|). А внутри каждого запуска есть циклы по всем вершинам, где на каждой итерации либо зануляется строка матрицы смежности, либо считается степень вершины. Обе операции выполняются за O(|V|) операций, что вместе с циклом дает $O(|V|^2)$. Итоговая сложность - $O(|V|^3)$. Так что второй алгоритм тоже работает за полиномиальное время.

9 Проверка корректности работы алгоритмов

Будем проверять работу алгоритмов на кликах и деревьях разных размеров. Клики и деревья выбраны в связи с тем, что их легко сгенерировать и для них можно быстро найти оптимальное вершинное покрытие.
Код генерации случайных клик и деревьев и проверки с достаточно подробными объяснениями лежит во втором файле. Также в том файле находится проверка на то, что алгоритмы возвращают вершинные покрытия. Здесь же приведем только графики.

Проверка на кликах

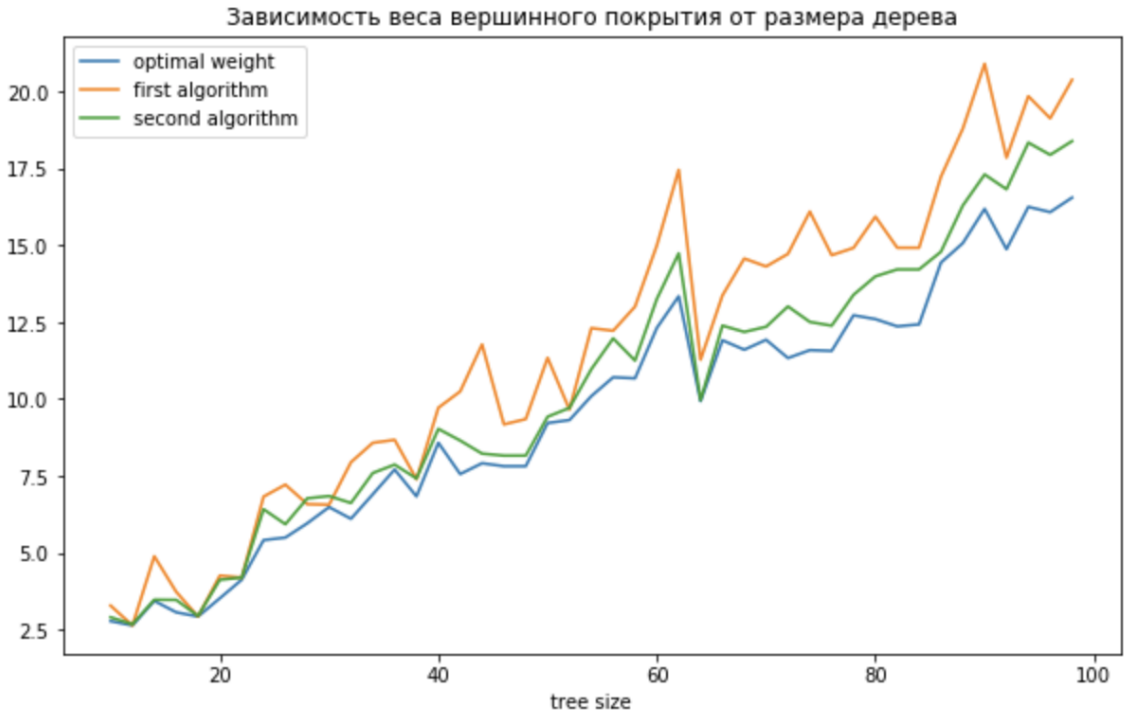
Проведем серию запусков на кликах со случайными весами вершин размера от 10 до 100 с шагом 2.
Первый график - график зависимости полученного в алгоритмах веса вершинного покрытия от размера клики. Также нарисована зависимость веса оптимального вершинного покрытия от размера клики.
Второй график - график зависимости веса покрытий, полученных в результате двух алгоритмов, поделенного на вес оптимального вершинного покрытия.

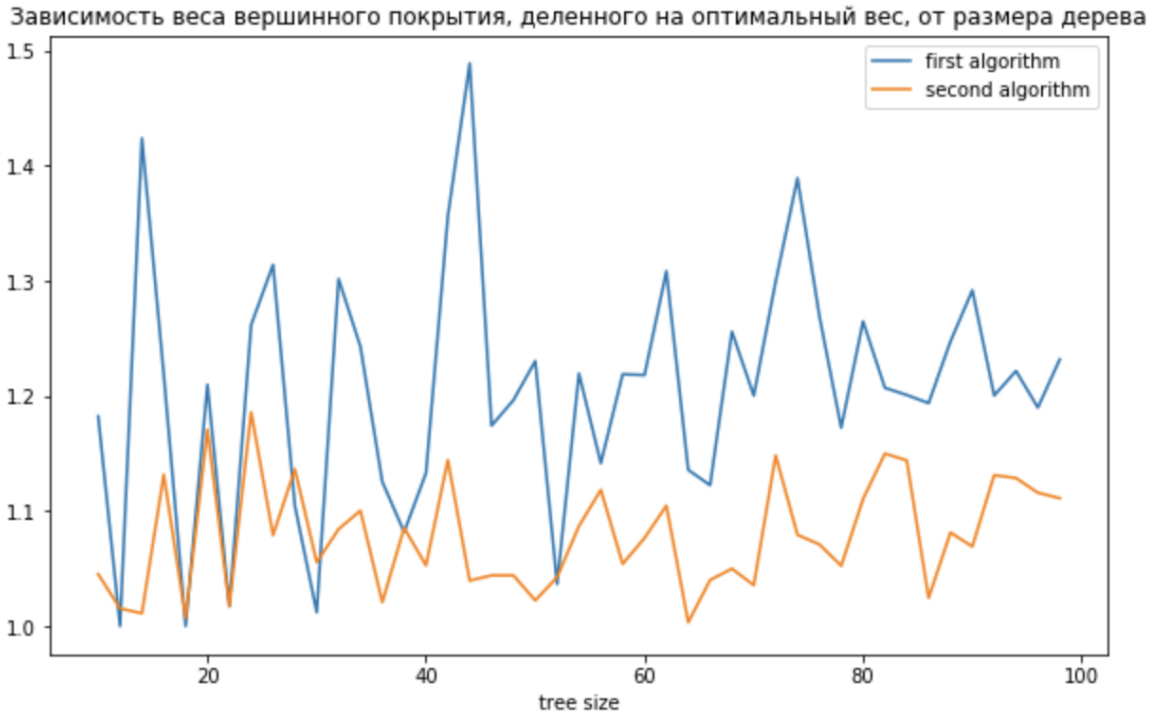

Зависимость веса вершинного покрытия от размера клики
optimal weight
first algorithm
second algorithm
45
40
35
30
25
20
15
10
5
20
40
60
80
100
clique size


Зависимость веса вершинного покрытия, деленного на оптимальный вес, от размера клики
first algorithm
second algorithm
1.10
1.08
1.06
1.04
1.02
1.00
20
40
60
80
100
clique size

Проверка на деревьях

Проведем серию запусков на деревьях со случайными весами вершин размера от 10 до 100 с шагом 2.
Первый график - график зависимости полученного в алгоритмах веса вершинного покрытия от размера дерева. Также нарисована зависимость веса оптимального вершинного покрытия от размера дерева.
Второй график - график зависимости веса покрытий, полученных в результате двух алгоритмов, поделенного на вес оптимального вершинного покрытия.





Из полученных графиков видно, что второй алгоритм всегда (по крайней мере всегда на этих примерах) дает более точный ответ. Также из графиков отношения найденных алгоритмами весов к оптимальному весу видим, что это отношение не превосходит 2, а это значит, что приведенные алгоритмы действительно находят 2-приближения.

10 Источники

https://en.wikipedia.org/wiki/Vertex_cover.
$http://ru.discrete-mathematics.org/fall2016/3/complexity/compl-book.pdf$