Параметризованные алгоритмы выделения плотных компонент в графах

Иван Близнец Николай Карпов

27 июня 2016 г.

Аннотация

Плотными называются графы на n вершинах, у которых реберная связность больше чем $\frac{n}{2}$. В нашей работе мы рассматриваем задачи редактирования графов до графа, состоящего из плотных компонент связности, или до графа, в котором существует плотная компонента. Представлен новый алгоритм для задачи редактирования графа путем удаления ребер до набора плотных компонент с временем работы $\mathcal{O}(3^k poly(n))$, где k это максимальное число удаляемых ребер. Для задачи выделения плотной компоненты связности размера ровно s из графа, путем удаления не более чем k ребер, найден алгоритм с субэкспоненциально зависящим от k временем работы $2^{\mathcal{O}(k^{2/3}\log k)} poly(n)$. Для задачи выделения в графе плотной компоненты найден алгоритм с временем работы $2^{\mathcal{O}(\sqrt{k}\log k)} poly(n)$, где k это количество ребер, которые лежат вне плотной компоненты.

1 Введение

Выделение в графе подграфов, которые имеют плотную структуру внутри себя и слабо связаны с остальным графом, является ключевой идеей кластеризации для моделей, представимых в виде графа, более подробно это рассмотрено в статьях [Alb05, New10, SUS07, SST04]. Хартув и Шамир [HS00] предложили алгоритм, который выдает разбиение на так называемые *плотные* графы. Там же введено ограничение на связность следующим образом: обозначим за $\lambda(G)$ реберную связность графа G, равную минимальному числу ребер, которое нужно удалить, чтобы граф стал несвязным. Тогда граф $G = (V, E)$ называется плотным, если $\lambda(G) > \frac{|V|}{2}$. Существует эквивалентное определение [Cha66], в котором плотным графом называется тот граф, у которого степень любой вершины больше половины от числа вершин графа. Алгоритм Хартува и Шамира [HS00] выдает такое разбиение множества вершин, что индуцированный граф на каждой части является плотным. Однако, алгоритм не гарантирует, что количество ребер между различными частями разбиения минимизируется.

Предыдущие результаты. Следующая задача была хорошо изучена Хафнером и соавторами[HKLN14].

HIGHLY CONNECTED DELETION
Вход: Неориентированный граф $G = (V, E)$.
Вопрос: Найти размер минимального множества ребер $E' \subseteq E$ такого, что любая компонента связности в $G' = (V, E \setminus E')$ плотная.

В[HS00] приведен точный алгоритм для задачи с временем $\mathcal{O}(3^n m)$, где $n = |V|$, $m = |E|$. Более того, изучна параметризованная сложность задачи относительно параметра k, равного размеру искомого множества E'. Там же приведен алгоритм с временем работы $\mathcal{O}(81^k k^2 + n^2 mk \log n)$ и построено ядро размера $\mathcal{O}(k^{1.5})$, получены алгоритмы, хорошо работающие на практике.

Далее в [HKS15] была исследована задача

HIGHLY CONNECTED SUBGRAPH
Вход: Неориентированный граф $G = (V, E)$.
Вопрос: Найти множество вершин S размера s такое, что $G[S]$ плотный граф.

Предложен алгоритм с временем работы $\mathcal{O}(4^k n^2 + (sn + k)nm)$, где k это размер множества $E(S, V \setminus S)$. Там же изучена параметризованная сложность данной задачи и при других параметризациях, например, размером вершинного покрытия графа.

Для задачи

SEEDED HIGHLY CONNECTED EDGE DELETION
Вход: Граф $G = (V, E)$, множество вершин $S \subseteq V$ и целые неотрицательные a и k.
Вопрос: Существует ли множество ребер $E' \subseteq E$ размера не более k такое, что $G - E'$ состоит из изолированных вершин и плотной компоненты C такой, что $S \subseteq V(C)$ и $|V(C)| = |S| + a$.

построен алгоритм с временем работы $\mathcal{O}^*(2^{4k^{0.75}})$.

Наши результаты. Для задачи HIGHLY CONNECTED DELETION улучшен алгоритм[HKLN14] с временем работы $81^k poly(n, m)$ до алгоритма, требующего $3^k poly(n, m)$ времени, так же построен алгоритм с временем работы $2^n poly(n, m)$. Для задачи HIGHLY CONNECTED SUBGRAPH построен алгоритм с временем работы $2^{\mathcal{O}(k^{2/3} \log k)} poly(n, m)$. Для задачи SEEDED HIGHLY CONNECTED EDGE DELETION улучшена субэкспоненциальная зависимость от k с $\mathcal{O}^*(2^{\mathcal{O}(k^{0.75})})$ до $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k} \log k)})$.

Предварительные сведения. В работе рассматриваются только неориентированные графы $G = (V, E)$, через n и m обозначаются величины $|V|$ и $|E|$ соответственно. Через $G[S]$ обозначается граф, индуцированный множеством $S \subseteq V$. Пусть $N(v) = \{u \mid uv \in E\}$ – открытая окрестность вершины v, через $N[v] = N(v) \cup \{v\}$ будет обозначаться замкнутая окрестность вершины. Разрезом графа мы будем называть множество ребер E', при удалении которого из графа число компонент связности увеличивается. Разрезом между вершинами s и t мы будем называть множество ребер E', после удаления которого s и t будут находится в разных компонентах связности. Положим $E(S, T) = \{uv \mid uv \in E, u \in S, v \in T\}$.

2 Алгоритмы для задачи разбиения на кластеры

В данном разделе исследуется задача HIGHLY CONNECTED DELETION. Основным алгоритмическим инструментом для получения быстрого алгоритма для данной задачи является быстрый алгоритм для решения задачи о вычислении свертки подмножеств в кольце $(min, +)$. Бьеркланд и др.[BHKK07] показали, что если f, g – две функции определенные на подмножествах n элементного множества N и возвращающие значение из промежутка целых чисел от 0 до M, тогда функция $f * g$, определенная для всех $S \subseteq N$ как $(f * g)(S) = \min_{T \subseteq S}(f(T) + g(S \setminus T))$, может быть вычислена за время $\mathcal{O}(2^n n^2 M)$.

Теорема 1. Для задачи HIGHLY CONNECTED DELETION существует алгоритм с временем работы $O(2^n n^3 m)$.

Доказательство. Определим функцию f на подмножествах вершин V следующим образом:

$$f(S) = \begin{cases} |E(S, V \setminus S)| & \text{если } G[S] \text{ плотный граф} \\ \infty & \text{иначе} \end{cases}$$

Заметим, что $f^{*k}(V) = \min_{S_1 \sqcup \dots \sqcup S_k = V}(f(S_1) + \dots + f(S_k))$. Если $f^{*k}(V) < \infty$, то оно равно удвоенному минимальному числу ребер между частями разбиения множества V на k непересекающихся множеств, иначе такого разбиения (ровно на k частей) не существует. Если мы возьмем минимум по всем k от 1 до n величины $f^{*k}(V)$, то получим удвоенный минимальный размер множества E', после удаления которого из графа получается граф, в котором каждая компонента связности является плотным графом. Если же этот минимум равен бесконечности, то такого разбиения не существует. Финальный алгоритм работает следующим образом: вычисляется f, что можно сделать за время $\mathcal{O}(2^n(n+m))$, после чего с помощью алгоритма из [BHKK07] считаются все f^{*k} и находится k, на котором достигается минимум $f^{*k}(V)$. Далее зная все f^{*i} для $i \le k$, узнается S_k за время 2^n. Поскольку $k \le n$, то на нахождение всех S_i нам потребуется время $\mathcal{O}(n2^n)$. Осталось научиться считать все f^{*k} за разумное время. Единственная проблема, которая мешает нам это сделать, заключается в наличии ∞ в области значений наших функций. Однако, можно заменить ∞ на $2m + 1$, так как если значение конечно, то оно не больше $2m$. Таким образом необходимо вычислить n сверток функций. Каждая свертка выполняется за время $\mathcal{O}(2^n n^2 M)$, а M ограниченно $2m + 1$, поэтому имеем итоговое время $O(2^n n^3 m)$. □

Представленный в доказательстве алгоритм формализуется в виде псевдокода 1.

Далее мы докажем наш главный результат о параметризованной сложности задачи HIGHLY CONNECTED DELETION. В параметризованной версии задачи нам требуется ответить следующий вопрос : «Существует ли множество E' размера не более чем k такое, что после его удаления из графа все компоненты связности плотные?».

Теорема 2. Существует алгоритм для задачи HIGHLY CONNECTED DELETION с временем работы $\mathcal{O}^*(3^k)$.

Алгоритм 1 Точный алгоритм для задачи HIGHLY CONNECTED DELETION

```
function EXACT − CLUSTERING(G = (V, E))
    for S ⊆ V do
        if G[S] плотный then
            f[1][S] = |E(S, V \ S)|
        else
            f[1][S] = 2|E| + 1
        end if
    end for
    for i ∈ {2, . . . , n} do
        f[i] = f[i − 1] ∗ f[1]
        for S ⊆ V do
            if f[i][S] > 2|E| then
                f[i][S] = 2|E| + 1
            end if
        end for
    end for
    result = min_{i∈[n]}(f[i][V])
    if result > 2|E| then
        return ∞
    else
        return result/2
    end if
end function
```

Сначала опишем правила упрощения, которые позволят построить эквивалентный (т.е, с тем же ответом) экземпляр задачи меньшего размера. Далее потребуется более тонкий анализ структуры графа и разбор различных случаев, как финальный штрих нам потребуются алгоритмы, похожие на алгоритм из теоремы 1.

Доказательство. Доказательства дальнейших правил и лемм можно найти в работе [HKLN14].

Правило 1. Если есть плотная компонента связности C, то переходим к входу задачи $(G[V \setminus V(C)], k)$.

Лемма 1. Пусть G – плотный граф, а u, v – две различные вершины. Если $uv \in E$, тогда $|N(u) \cap N(v)| \geq 1$. Если $uv \notin E$, тогда $|N(u) \cap N(v)| \geq 3$.

Правило 2. Если есть две вершины u, v, которые соединены ребром и у них нет общих соседей, вернем ответ для входа $((V, E \setminus \{uv\}), k-1)$.

Определение 1. Пара вершин u и v *k-неразлучны*, если любой разрез между ними имеет размер больше k.

Правило 3. Пусть множество S – максимальное по включению множество попарно k-неразлучных вершин и его размер больше $2k$. Если $G[S]$ – не плотный, выдадим ответ «нет», иначе выдадим ответ для входа $(G[V \setminus S], k - |E(S, V \setminus S)|)$.

Лемма 2. Пусть G – плотный граф. Тогда диаметр G не больше 2.

Заметим, что все имеющиеся правила можно применить за полиномиальное время. Теперь с помощью лемм и правил мы готовы создать наш алгоритм. Заметим, что не умаляя общности, мы можем считать, что граф G связный и имеет диаметр 2. Действительно, если G несвязный, то задача распадается на независимые подзадачи для каждой компоненты связности, для которых можно найти минимальное число ребер, которое нужно удалить для разбиения на плотные компоненты, увеличивая в них параметр k' от 0 до k, пока не найдем минимальное k', при котором эту компоненту связности можно сделать плотной или определим, что все $k' \leq k$ не подходят. Если же диаметр G меньше 2, то граф является кликой, и ответ на задачу тривиален. По лемме 2, если у нас есть пара вершин на расстоянии больше 2, то в оптимальном ответе они лежат в разных кластерах. Значит, как минимум одно ребро лежащие на кратчайшем пути должно принадлежать E'. Имеется три возможности разрушить путь между этой парой вершин, удалив одно из трех ребер. Таким образом, имеем ветвление по трем возможностям разрушить этот путь, в каждой ветке перебора параметр k будет убывать на единицу. Предположим, что нет возможности ни разрушить путь, ни применить правило упрощения, ни разбить задачу на две независимые подзадачи, и текущий граф неплотный. Значит, имеем связный граф диаметра 2, неплотный, в нем нет множеств k-неразлучных вершин размера больше чем $2k$. Докажем, что если в нем размер E' не более k, тогда вершин в не более ck, где константа c будет определена позднее. Рассмотрим некоторое оптимальное разбиение на кластеры $C_1, \ldots, C_l$. Скажем, что вершина *недотрога*, если она несмежна в G ни с одной вершиной вне своего кластера. Будем говорить, что вершина *доступная*, в противном случае. Обозначим через U множество всех недотрог, через A – множество всех доступных вершин. Далее через $C(v)$ будем обозначать кластер C_i, в котором содержится вершина v.

Лемма 3. Пусть G – граф диаметра 2, тогда в оптимальном разбиении на кластеры $C_1, \ldots, C_l$ существует такой кластер C_i, что $U \subseteq C_i$.

Доказательство. Пусть есть две недотроги $u, v \in U$ и $C(v) \neq C(u)$. Заметим, что тогда длина кратчайшего пути между ними не меньше 3, так как путь должен содержать ребро из E' и два различных ребра, которые исходят из u, v в их кластеры $C(u)$ и $C(v)$ соответственно. Следовательно, множество U полностью лежит в одном кластере. $\square$

Лемма 4. Пусть $\{C_i\}$ – оптимальное разбиение на кластеры графа G, диаметр которого равен 2. Если $U \neq \emptyset$, тогда $|E'| \geq n - |C_i|$, где C_i это кластер содержащий U.

Доказательство. Заметим, что из любой вершины вне кластера ведет ребро в кластер C_i. Действительно, диаметр графа G равен 2, значит из недотроги $s \in U$ должен существовать путь в любую вершину длины не более 2. Следовательно, любая вершина вне $C(s)$ смежна с кластером $C(s)$, но таких вершин ровно $n - |C(s)|$, значит $|E'| \geq n - |C(s)|$. $\square$

Если ответ на задачу положителен, то $k \geq |E'| \geq \frac{|A|}{2}$, $n = |A| + |U|$, $|U| \leq 2k$, в частности $n \leq 4k$. Неравенство $|U| \leq 2k$ верно в силу правила 3, если бы размер кластера $C_i \supseteq U$ был больше $2k$, тогда мы бы могли его выделить и сократить экземпляр задачи, по 3, так как кластер из более чем $2k$ вершин k-неразлучное множество.

Для достижения финального результата нам потребуются два алгоритма, каждый из которых работает в разных предположениях о графе. Мы будем рассматривать два случая, когда в графе есть недотрога и когда все вершины доступные. Для оценки времени работы алгоритмов будет полезна следующая лемма.

Лемма 5 ([FKP+13])**.** Для любых неотрицательных целых a, b верно $\binom{a+b}{b} \leq 2^{2\sqrt{ab}}$.

Лемма 6. Пусть G – связный граф диаметра 2, имеющий искомое разбиение на кластеры с $U \neq \emptyset$. Тогда HIGHLY CONNECTED DELETION можно решить за время $\mathcal{O}^*(2^{\frac{3k}{2}})$.

Доказательство. Зафиксируем недотрогу s. Заметим, что по лемме 4 s лежит в кластере размера как минимум $n-k$. По определению s $N(s) \subset C(s)$ и $|N(s)| > \frac{|C(s)|}{2}$. Рассмотрим множество $V \setminus N[s]$, и его разбиение на две части $V \setminus N[s] = W \sqcup H$, где $W = \{v | 1 \leq |N(s) \cap N(v)| \leq 2\}$, а $H = \{v \mid 3 \leq |N(s) \cap N(v)|\}$. Заметим, что W не содержится в кластере по лемме 1. Если угадать $G = C(s) \cap H$ и $B = H \setminus G$, то задача решается за время $\mathcal{O}^*(2^{|B \cup W|})$ по теореме 1. Угадывание требует время $\mathcal{O}^*(\binom{|G|+|B|}{|G|})$, после чего запускается точный алгоритм. В итоге мы получим алгоритм с временем работы по лемме 5 не более

$$\mathcal{O}^*(\binom{|G| + |B|}{|G|} 2^{|B|+|W|}) = \mathcal{O}^*(2^{2\sqrt{|G||B|}+|B|+|W|}).$$

Остается это оценить. Мы знаем, что $|G| \leq \frac{|C(s)|}{2} \leq k$, $3|B| + |W| \leq k$, значит время работы можно оценить как

$$\mathcal{O}^*(2^{2\sqrt{k|B|}-2|B|+k}).$$

$|B| = \frac{k}{4}$. Подставив последнее получаем итоговую асимптотику $\mathcal{O}^*(2^{\frac{3k}{2}})$. □

Из доказательства леммы вытекает следующий алгоритм:

Алгоритм 2 Случай существования недотроги

```
function UNAFFECTED − CLUSTERING(G = (V, E), k)
    for s ∈ V do
        W = {v | v ∈ V \ N[s], |N(v) ∩ N(s)| ≤ 2}
        H = {v | v ∈ V \ N[s], |N(v) ∩ N(s)| ≥ 3}
        for g : (g + |N[s]|)/2 < |N(s)|  &  g ≤ k  &  3(|H| − g) + |W| ≤ k do
            for G ⊆ H  &  |G| = g do
                Q = N[s] ∪ G
                if G[S ∪ G] плотный then
                    if |E(Q, V \ Q)| + EXACT − CLUSTERING(G[V \ Q]) ≤ k then
                        return YES
                    end if
                end if
            end for
        end for
    end for
    return NO
end function
```

Остался второй случай, когда все вершины доступные. Этот случай будет немного сложнее. Сначала заметим, что если $n \leq 1.57k \leq k \log_2 3$, то можно применить алгоритм 1. Поэтому можно считать, что $1.57k \leq n \leq 2k$.

Докажем один факт о таких графах.

Лемма 7. Пусть (G, k) вход задачи HIGHLY CONNECTED DELETION, ответ на который положителен, G связный и $n \geq 1.57k$. Тогда в оптимальном разбиении на кластеры есть два различных кластера C_i и C_j таких, что $|C_i| + |C_j| \geq n - k$.

Доказательство. Поскольку $n \geq 1.57k$, значит есть вершина $s \in V$, которой в некотором оптимальном E' смежно одно ребро. Пусть это ребро st. Рассмотрим два кластера $C(s)$ и $C(t)$. Докажем, что они искомые кластеры. Действительно, диаметр графа G равен 2, значит из вершины s можно добраться по пути длины не более 2 до любой вершины. Следовательно, любая вершина из $V \setminus (C(s) \cup C(t))$ соединена с $C(s) \cup C(t)$ ребром из E', а их не больше k, если ответ на задачу положителен. □

Пусть $|C(s)| > 2n - 3.14k$. Покажем, что в этом случае задача решается за время $\mathcal{O}^*(2^{n-\frac{|C(s)|}{2}}) = \mathcal{O}^*(3^k)$. Надо перебрать вершину степени 1, s, удаленное ребро, тогда определяется множество соседей s, размера как минимум $\frac{|C(s)|}{2}$. Точнее, это все вершины, которые остались соединены ребром с s. Далее поступим аналогично теореме 1. Определим три функции f, g, h на подмножествах $W = V \setminus U(s)$.

- $f(S) = |E(S, W \setminus S)|$ если $G[S]$ плотный, иначе ∞.
- $h(S) = \min\limits_i(f^{*i}(S))$.

- $g(S) = 2|E(W \setminus S, U)| + |E(S, W \setminus S)|$ если $G[U \cup S]$ плотный, иначе ∞.

Интерпретация этих функция следующая: $f(S)$ – это количество ребер, которое нужно удалить, чтобы отделить кластер S. $h(S)$ характеризует, насколько дорого разбить множество на кластеры. $g(S)$ показывает, насколько дорого сделать множество S искомой добавкой к $U(s)$ до кластера $C(s)$. Множитель 2 в определении g может показаться контринтуитивным, однако при рассмотрении определения $g*h$ на манер теоремы 1, не сложно увидеть, что в значениях функции h все ребра внутри U, не лежащие внутри одного кластера, будут учтены два раза, по причине участия в двух слагаемых с коэфициентом 1, а все ребра между W и U, не лежащие в кластере, будут учтены с коэфициентом 2. Таким образом мы доказали, что $(g * h)(W)$ равно удвоенному минимальному числу ребер требуемого для разбиения исходного графа на кластеры. Это заканчивает разбор случая $|C(s)| \geq 2n - 3.14k$.

Пусть $|C(s)| \leq 2n-3.14k$, следовательно $n-k \leq |C(s)|+|C(t)| \leq 2n-3.14k+|C(t)|$. Значит $|C(t)| + 2n - 3.14k \geq n - k$, что влечет $|C(t)| \geq 2.14k - n \geq 0.14k$. Следовательно, в $C(t)$ есть вершина, с которой смежно не более 7 ребер из E'. Переберем эту вершину t и удалим эти ребра из графа. Теперь мы знаем как минимум половину вершин $U(s)$ из $C(s)$ и как минимум половину $U(t)$ из $C(t)$. Положим, $U = U(s) \cup U(t)$. В текущий момент наша цель – решить задачу за время $\mathcal{O}^*(2^{n-\frac{|C(s)|+|C(t)|}{2}})$. Применим более сложный аналог алгоритма из теоремы 1. Приведем функции свертку которых нужно посчитать, а далее действуем так же, как и в доказательстве теоремы 1. Наши функции будут определены на подмножествах множества $W = V \setminus U$.

- $f(S) = |E(S, W \setminus S)|$ если $G[S]$ плотный, иначе ∞.
- $h(S) = \min_i(f^{*i}(S))$.
- $g_s(S) = 2|E(S, U(t))| + |E(S, W \setminus S)|$ если $G[S \cup U(s)]$ плотный, иначе ∞.
- $g_t(S) = 2|E(S, U(s))| + |E(S, W \setminus S)|$ если $G[S \cup U(t)]$ плотный, иначе ∞.

Отличие от предыдущего случая в том, что вместо одной функции g появилось две функции g_s, g_t. Считая свертку $h * g_s * g_t$, мы получаем искомую величину, а именно в $(h * g_s * g_t)(W)$ будет находится удвоенное минимальное число ребер, необходимое для выделения кластеров $C(s)$, $C(t)$ и разбиения остального множества на кластеры. Итого, имеем время работы $\mathcal{O}^*(2^{n-\frac{n-k}{2}}) = \mathcal{O}^*(2^{\frac{n+k}{2}}) = \mathcal{O}^*(2^{\frac{3k}{2}})$.

Приведем псевдокод алгоритма 3, который будет проверять наличие множества E' размера не более k в предположении, что в любом разбиении на кластеры нет недотрог.

Таким образом, мы доказали, что в случаях когда мы не можем совершить шаг расщепления по кратчайшему пути длины 3 между парой вершин, мы можем решить задачу за время $\mathcal{O}^*(3^k)$. Если мы обозначим через $T(k)$ верхнюю оценку времени работы алгоритма расщепления, с параметром k, то оно удовлетворяет рекуррентному соотношению $T(k) \leq 3T(k-1)$. Поскольку любая ситуация, в которой расщепление невозможно так же обрабатывается за время $\mathcal{O}^*(3^k)$, значит и весь алгоритм работает за время $\mathcal{O}^*(3^k)$. Что подводит итог в доказательстве нашей теоремы.

□

Алгоритм 3 Случай отсутсвия недотрог

function $AFFECTED - CLUSTERING((V, E), k)$
  if $|V| \leq 1.57k$ **then**
    return $EXACT - CLUSTERING((V, E)) \leq k$
  end if
  if $|V| > 2k$ **then**
    return NO
  end if
  for $sx \in E$ **do**
    $U(s) = N[s] \setminus \{x\}$
    if $|U(s)| > n - 1.57k$ **then**
      Вычисление $f, h, g, g * h$ для подмножеств $V \setminus U(s)$
      if $(g * h)(V \setminus U(s)) \leq 2k$ **then**
        return YES
      end if
    else
      for $0 \leq l \leq 7, (ty_1, \ldots, ty_l) \in E^l$ **do**
        $U(t) = N[t] \setminus \{y_1, \ldots, y_l\}$
        $U = U(s) \cup U(t)$
        if $U(s) \cap U(t) = \emptyset \wedge |U| \geq \frac{k}{2}$ **then**
          Вычисление $f, h, g_s, g_t, h * g_s * g_t$ на подмножествах $V \setminus U$
          if $(h*g_s * g_t)(V \setminus U) \leq 2(k - |E(U(s), U(t))|$ **then**
            return YES
          end if
        end if
      end for
    end if
  end for
  return NO
end function

3 Субэкспоненциальные алгоритмы

Сначала решим параметризованную версию задачи HIGHLY CONNECTED SUBGRAPH.

HIGHLY CONNECTED SUBGRAPH
Вход: Неориентированный граф $G = (V, E)$, неотрицательное целое число k.
Вопрос: Существует ли множество вершин S размера s такое, что $G[S]$ плотный граф и $|E(S, V \setminus S)| \leq k$.

Данную версию задачи активно изучали Хафнер и соавторы[HKS15]. В частности, для более общей постановки задачи

f-HIGHLY CONNECTED SUBGRAPH
Вход: Неориентированный граф $G = (V, E)$, неотрицательное целое число k, функция $f : V \to \mathbb{Z}_{\geq 0}$.
Вопрос: Существует ли множество вершин S размера s такое, что $G[S]$ плотный граф и $|E(S, V \setminus S)| + \sum\limits_{s \in S} f(s) \leq k$.

был найден набор правил упрощения входа задачи. Доказательство корректности следующих правил упрощения и лемм можно найти в работе [HKS15].

Правило 4. Если G содержит компоненту связности C размера менее s, то перейдем к входу $(G \setminus C, f, k)$.

Правило 5. Если G содержит компоненту связности $C = (V', E')$, имеющую минимальный реберный разрез больше k, тогда, если C плотный граф, $|V'| = s$ и $|E(V', V \setminus V')| + \sum\limits_{s \in V'} f(s) \leq k$ выдать ответ «да», иначе перейти к входу $(G \setminus C, f, k)$.

Правило 6. Если в G есть связная компонента C с минимальным разрезом (A, B) размера не более чем $\frac{s}{2}$, тогда для каждой вершины $v \in A$ переопределим $f(v) = f(v) + |N(v) \cap B|$ и для каждой вершины $v \in B$ переопределим $f(v) = f(v) + |N(v) \cap A|$ и перейдем к входу задачи $(G \setminus \ E(A, B), f, k)$.

Лемма 8. Если правила 4, 5, 6 нельзя применить, тогда $k > \frac{s}{2}$.

Лемма 9. Применение правил 4, 5, 6 можно выполнить за время $\mathcal{O}((sn + k)m)$.

Далее нам потребуется следующий результат

Предложение 1. [FV12]

Пусть G граф. Тогда для любой вершины $v \in V(G)$ и $b, f \leq 0$, число связных множеств $B \subseteq V(G)$ таких, что

i) $v \in B$

ii) $|B| = b + 1$

iii) $|N(B)| = f$

не превосходит $\binom{b+f}{b}$. Более того, все эти множества могут быть перечислены за время $\mathcal{O}(\binom{b+f}{b}(n + m)b(b + f))$.

Теперь мы готовы изложить алгоритм.

Теорема 3. Задача f-HIGHLY CONNECTED SUBGRAPH может быть решена за время $2^{\mathcal{O}(k^{\frac{2}{3}} \log k)}$.

Доказательство. Применив правила 4, 5, 6, по лемме 8 получим вход задачи, в котором $2k > s$. Мы рассмотрим два случая: когда $k^{\frac{2}{3}} < s$ и $k^{\frac{2}{3}} \geq s$. Рассмотрим случай $s \leq k^{\frac{2}{3}}$. Переберем все связные множества вершин размера s, у которых размер окрестности не превышает k. Очевидно, что если искомое S существует, то оно находится среди таких множеств. Таких множеств не более, чем $n\mathcal{O}^*(\binom{s+k}{s})$ по предложению 1, что можно грубо оценить как $\mathcal{O}^*((s+k)^s)$. Поскольку $s < 2k$ и $s < k^{\frac{2}{3}}$, имеем оценку на время перебора таких множеств $\mathcal{O}^*(2^{k^{\frac{2}{3}} \log k})$. Понятно, что если мы переберем все такие множества S и для каждого проверим свойство, граф $G[S]$ плотный и $|E(S, V \backslash S)| + \sum\limits_{v \in S} f(v) \leq k$.

Теперь, случай когда $k^{\frac{2}{3}} < s$. Пусть искомое множество S существует, оно задает какое-то множество $E' = E(S, V \setminus S)$. Рассмотрим функцию $d : S \to \mathbb{Z}_{\geq 0}$, где $d(v) = \ |N(v) \cap (V \backslash S)|$. Если $\sum\limits_{v \in S} d(v) = |E(S, V \backslash S| \leq k$, значит есть $v \in S$ с $d(v) \leq \frac{k}{s} < k^{\frac{1}{3}}$. Заметим, что если такая v существует, то $|N(v)| \leq s + k^{\frac{1}{3}}$. Действительно, все ребра которые не учтены в $d(v)$ ведут в S, а таких не более s. Угадаем такую вершину. Это займет время не более чем

$$n \sum_{i \leq k^{\frac{1}{3}}} \binom{s + k^{\frac{1}{3}}}{i} \leq n k^{\frac{1}{3}} 2^{2^{\sqrt{\left(s + k^{\frac{1}{3}} - i\right) i}}} \leq n k^{\frac{1}{3}} 2^{2^{\sqrt{3k^{4/3}}}} = n 2^{O(k^{2/3})} .$$

Таким образом, мы уже угадали как минимум $\frac{s}{2} + 1$ вершин из S, обозначим это множество через W. Мы хотим расширить наше множество W до S. Для этого опишем алгоритм основанный на расщеплении, принимающий на вход (G, f, k, s, W, B). Множество W будет соответствовать угаданной части множества S, а множество B части, которая точно не лежит в множестве S. Сначала запустим алгоритм с параметрами $(G, f, k, s, W, \emptyset)$. Для большей наглядности мы приводим псевдокод алгоритма 4. Сначала докажем корректность данного алгоритма. В некотором смысле мы ищем разбиение множества V на $S = W$ и $B = V \setminus W$. В случае расщепления это происходит явно, а в случае, когда мы находим множество W размера s, мы отправляем в B все множество $V \setminus (W \cup B)$. Заметим, что алгоритм выдает «да» только если находит пару множеств $(W, V \setminus W)$ таких, что $\sum\limits_{x \in W} f(x) + |E(W, V \setminus W)| \leq k$. $|W| = s$ мы вычтем из начального параметра k величину равную $|E(W, B)|$, так как любое ребро из такого множества вычитается ровно один раз в момент, когда мы определяем, в каких множествах лежат оба конца ребра. Отметим, что ответ «нет» выдается в случае, когда Q становится пустым. Если мы докажем, что в случае $W \subseteq S$ для какого-то S, удовлетворяющего условию задачи, Q непустое, то тем самым будет доказана корректность алгоритма. Пусть $W \subseteq S$, тогда существует $x \in S \setminus W$ и $|S \cap N(x)| > \frac{s}{2}$. Но тогда $|N(x) \cap W| + |N(x) \cap (S \setminus W)| > \frac{s}{2}$, значит $|N(x) \cap W|$ можно оценить снизу как $|N(x) \cap W| > \frac{s}{2} - |N(x) \cap (S \setminus W)| > \frac{s}{2} - (s - |W|) = |W| - \frac{s}{2}$. Следовательно, Q не пусто, что завершает доказательство корректности.

Алгоритм 4 Алгоритм выделения плотной компоненты

```
function BRANCHING(G, f, k, s, W, B)
    if k < 0 then
        return NO
    end if
    if |W| = s then
        return G[W] плотный и Σ_{v∈W} f(v) + |E(W, V \ (W ∪ B))| ≤ k
    end if
    Q = {v | v ∈ V \ (W ∪ B), |N(v) ∩ W| > |W| − s/2}
    if Q пусто then
        return NO
    end if
    выберем x ∈ Q
    if BRANCHING(G, f, k − |N(x) ∩ B|, s, W ∪ {x}, B) = YES then
        return YES
    end if
    if BRANCHING(G, f, k − |N(x) ∩ W|, s, W, B ∪ {x}) = YES then
        return YES
    end if
    return NO
end function
```

Теперь докажем оценку на трудоемкость алгоритма расщепления. Для этого достаточно оценить количество вершин в дереве расщепления. Пусть $BRANCHING$ вызывается с параметром $k \geq 0$. Рассмотрим последовательность рекурсивных вызовов, которые привели к текущему вызову. В одной из веток перебора увеличивается B, а в другой увеличивается W. Пускай в i раз, когда мы уходили в ветку перебора увеличения B, мы сделали $a_i - 1$ переходов в ветку увеличения W, а через b обозначим количество уходов в ветку перебора увеличения W после последнего увеличения B. Заметим, что каждой вершине дерева перебора соответствует не ровно один такой набор (он определяется однозначно), и любым двум различным вершинам перебора, в которых $k \geq 0$, соответствуют разные наборы $(a_1, \ldots, a_l, b)$. Оценим количество таких наборов. Заметим, что $a_i \leq a_{i+1}$ и $\sum\limits_{i=1}^{l} a_i \leq k$. Последнее верно, так как каждый переход в ветку увеличения W увеличивает величину $|W| - \frac{s}{2}$ как минимум на 1, которая является нижней оценкой на величину $|N(x) \cap W|$, причем $|W| - \frac{s}{2} \geq 1$. Следовательно, в i раз, когда мы уходим в ветку перебора увеличения B, мы уменьшаем k на $|N(x) \cap W| \geq a_i$, следовательно $\sum\limits_{i=1}^{l} a_i \leq k$. Значит, мы можем оценить число вершин в дереве расщепления числом наборов $(a_1, \ldots, a_l, b)$ таких, что $a_i \geq 1, a_i \leq a_{i+1}, \sum\limits_{i=1}^{l} a_i \leq k, 0 \leq b \leq s$. Число наборов $(a_1, \ldots, a_l)$ ограничено $2^{\mathcal{O}(\sqrt{k})}$. Доказательство данного факта можно найти в работе [dAP09]. В свою очередь $b \leq n$. Значит, суммарное число наборов $(a_1, \ldots, a_l, b)$ ограничено $s2^{\mathcal{O}(\sqrt{k})}$.

Остается заметить, что число вызовов с параметром $k < 0$ не превосходит удвоенного числа вызовов с параметром $k \geq 0$.

Итоговый алгоритм будет угадывать хотя бы $\frac{s}{2} + 1$ элемент из S, а потом с помощью алгоритма расщепления пытаться дополнить его до S за время $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})})$. Поскольку суммарное число вариантов начальных множеств $\mathcal{O}^*(2^{\mathcal{O}(k^{\frac{2}{3}})})$, итого получаем время работы $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})}2^{\mathcal{O}(k^{\frac{2}{3}})}) = \mathcal{O}^*(2^{\mathcal{O}(k^{\frac{2}{3}})})$, что завершает доказательство теоремы. □

Далее мы улучшим результат для задачи SEEDED HIGHLY CONNECTED EDGE DELETION.

SEEDED HIGHLY CONNECTED EDGE DELETION
Вход: Граф $G = (V, E)$, множество вершин $S \subseteq V$ и целые неотрицательные a и k.
Вопрос: Существует ли множество ребер $E' \subseteq E$ размера не более k такое, что $G - E'$ состоит из изолированных вершин и плотной компоненты C такой, что $S \subseteq V(C)$ и $|V(C)| = |S| + a$.

В работе [HKS15] был представлен алгоритм, для задачи SEEDED HIGHLY CONNECTED EDGE DELETION с временем работы $\mathcal{O}(16^{k^{0.75}} + k^2nm)$. Мы улучшили этот результат, доказав следующую теорему.

Теорема 4. Задача SEEDED HIGHLY CONNECTED EDGE DELETION может быть решена за время $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k}\log k)})$.

Нам потребуется несколько лемм и правил упрощения входа задачи из работы [HKS15].

Правило 7. Если существует такая компонента связности $C = (V', E')$ в G, что минимальный разрез в ней хотя бы $k + 1$, тогда выдать ответ «да», если C плотная, $S \subseteq V'$, $|V' \setminus S| = a$ и оставшийся граф содержит не более k ребер, иначе ответить «нет».

Правило 8. Если существует компонента связности в G такая, что величина минимального разреза не более чем $\frac{a+|S|}{2}$, тогда удалить этот разрез и уменьшить параметр k на величину разреза.

Лемма 10. Если правила 7 и 8 неприменимы, тогда $k > \frac{a+|S|}{2}$

Лемма 11. Суммарное время применения правил 7 и 8 ограничено $\mathcal{O}(k^2nm)$.

Теорема 5. Любой вход SEEDED HIGHLY CONNECTED EDGE DELETION может быть преобразован в эквивалентный с числом вершин не более $2k + \frac{4k}{a}$ и ребер $\binom{2k}{2} + k$ за время $\mathcal{O}(k^2nm)$.

Теперь используя предыдущие леммы, правила и теорему, мы готовы доказать теорему 4.

Доказательство теоремы 4. С помощью правил 7 и 8 мы преобразуем вход задачи к эквивалентному с числом вершин не более $2k + \frac{4k}{a}$ и ребер $\binom{2k}{2} + k$ по теореме 5. Рассмотрим два случая: когда $a \leq 2\sqrt{k}$ либо $a > 2\sqrt{k}$. В первом случае мы можем перебрать все множества V' и проверить удовлетворяет ли V' требованиям задачи, что можно сделать за полиномиальное время, и выдать «да» если хотя бы одно множество V' подходит. Очевидно, что такой алгоритм будет корректен. Осталось оценить время его работы. Всего подмножеств размера a множества V не более чем $\binom{2k+\frac{4k}{a}}{a} \leq \binom{6k}{a} \leq (6k)^a \leq 2^{\mathcal{O}(\sqrt{k}\log k)}$. Следовательно, в первом случае можно узнать ответ за время $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k}\log k)})$.

Рассмотрим второй случай, когда $a > 2\sqrt{k}$. Заметим, что в таком случае степень любой вершины из V' должна быть больше $\sqrt{k}$, так как вершины степени меньше $\sqrt{k}$ не могут попасть в V'. Но в V' могут не попасть не более чем $\sqrt{k}$ вершин степени больше $\sqrt{k}$, так как если бы их было больше, то размер E' был бы больше k. Мы можем перебрать эти вершины за время $\mathcal{O}^*(\sum\limits_{i\leq\sqrt{k}} \binom{n}{i}) = \mathcal{O}^*(\binom{6k}{\sqrt{k}}) = \mathcal{O}^*(2^{\mathcal{O}(\sqrt{k}\log k)})$, взять в V' все остальные вершины степени больше $\sqrt{k}$ и проверить подходит ли V' в качестве ответа. Если хотя бы одно подходит то выдать ответ «да». $\square$

4 Заключение

В нашей работе мы улучшили оценки для четырех задач, связанных с выделением в графе кластеров. Пожалуй, самый интересный вопрос – это параметризованная сложность задачи редактирования графа до набора плотных связных компонент в случае, когда мы разрешаем добавлять и удалять ребра из графа. Вопрос о разрешимости данной задачи за время $f(k)poly(n)$, где k это верхняя оценка на число операций изменений, является открытым.

Список литературы

[Alb05] Reka Albert. Scale-free networks in cell biology. *Journal of cell science*, 118(21):4947–4957, 2005.

[BHKK07] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2007.

[Cha66] Gary Chartrand. A graph-theoretic approach to a communications problem. *SIAM Journal on Applied Mathematics*, 14(4):778–781, 1966.

[dAP09] Wladimir de Azevedo Pribitkin. Simple upper bounds for partition functions. *The Ramanujan Journal*, 18(1):113–119, 2009.

[FKP+13] Fedor V Fomin, Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 20. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[FV12] Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012.

[HKLN14] Falk Hüffner, Christian Komusiewicz, Adrian Liebtrau, and Rolf Niedermeier. Partitioning biological networks into highly connected clusters with maximum edge coverage. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 11(3):455–467, 2014.

[HKS15] Falk Hüffner, Christian Komusiewicz, and Manuel Sorge. Finding highly connected subgraphs. *SOFSEM 2015: Theory and Practice of Computer Science*, pages 254–265, 2015.

[HS00] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information processing letters*, 76(4):175–181, 2000.

[New10] Mark Newman. *Networks: an introduction.* OUP Oxford, 2010.

[SST04] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1):173–182, 2004.

[SUS07] Roded Sharan, Igor Ulitsky, and Ron Shamir. Network-based prediction of protein function. *Molecular systems biology*, 3(1):88, 2007.