Минимальное доминирующее множество

Новиков Иван Б05-121

Алгоритм поиска минимального доминирующего множества в произвольном графе

Содержание

1 Введение

Доминирующим множеством в графе $G = (V, E)$ называется множество $M \subset V$, такое что любая вершина v либо лежит в M, либо соединена ребром с одной из вершин, лежащих в M

Задание
(а) Докажите, что задача поиска наименьшего доминирующего множества NP-трудная;
(б) Имплементируйте какой-нибудь алгоритм поиска наименьшего доминирующего множества, работающий за $O(c^n)$ для $c < 1.9$.

2 Определения

1. Минимальное доминирующее множество - доминирующее множество наименьшего размера
2. $MinSetCover = \{(S, U, M) \mid U-$набор подмножеств $S, S = \bigcup_{u \in U} u, V \subset U, M = \bigcup_{v \in V} v, W, K : W \subset U, K = \bigcup_{w \in W} w, |W| < |V|\}$
3. Рёберное покрытие графа — это множество рёбер C, такое, что каждая вершина графа инцидентна по меньшей мере одному ребру из C.
4. Максимальное рёберное покрытие - рёберное покрытыие наибольшего размера
5. Паросочетание в графе - набор попарно несмежных рёбер
6. Максимальное паросочетание - это такое паросочетание M в графе G, которое не содержится ни в каком другом паросочетании этого графа

3 NP-трудность

(Сведение взято из [1] из списка литературы)
Докажем, что Min Dominating Set (далее MDS) - NP-трудная, сведя к ней другую NP-трудную задачу: Min Set Cover (далее MSC).
MSC:
Дано:

$$C = \{S_1, ..., S_n\}$$
$$X = \bigcup_{S \in C} S = \{x_1, ..., x_m\}$$

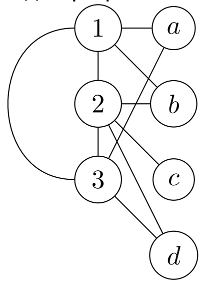
Задача - найти минимальное по мощности $Y \subset C$, такое, что $X = \bigcup_{S \in Y} S$
Она NP-трудная, сведём её к MDS
Построим неориентированный $G = (V, E)$ следующим образом:

$$V = \{1, ..., n, x_1, ..., x_m\}$$
$$E = \{(i, j) : 1 \leq i < j \leq n\} \cup \{(i, x) : 1 \leq i \leq b \land x \in S_i\}$$

Пример:

$$X = \{a, b, c, d\}$$
$$C = \{\{a, b\}, \{b, c, d\}, \{a, d\}\}$$

Тогда граф такой:



Заметим следующее:
Если есть доминирующее множество $V' \subset V$, то можно восстановить Set Cover:
Все i соединены между собой, и все x_j не соединены между собой.
Каждое x_j имеет хотя бы один i, с которым он связан ребром, так как существует хотя бы одно подмножество, в котором он содержится. Тогда мы можем заменить в нашем доминирующем множестве x_j на такой i и мощность этого доминирующего множества не увеличится.
Тогда заменим все $x_j \in V'$ на i, получится доминирующее множество только с числами - так как каждое число, по сути, отвечает за подмножество, то мы получили Set Cover размера $|V'|$.
Таким образом мы получили полиномиальное сведение, ведь вершин тут будет $n+m$, а рёбер не более $nm + n^2$
Следовательно MSC $\leq_p$ MDS $\Rightarrow$ MDS - это NP-трудная задача

4 Сведение к MSC

Алгоритм заключается в следующем:
1) свести MDS к MSC
2) решить MSC

Поймём, как свести MDS как MSC:
Дано:

$$G = (V, E)$$
$$V = \{1, ..., n\}$$

Построим C

$$C = \{S_1, ..., S_n\}$$
$$S_i = \{i\} \cup \{j \in [1, ..., n] : (i, j) \in E\}$$

Тогда заметим, что i может быть покрыт, взяв
1) S_i
2) $S_j, (i, j) \in E$
Добавление S_i эквивалентен взятию i в доминирующее множество, а добавление S_j взятию j. Тогда решив MSC мы решаем и MDS
Можно заметить, что такое сведение строит наборы размера не более $|G| + 1$, и каждый элемент в X встречается не более чем в $|G| + 1$ наборах. Таким образом, мы получили сведение с линейным замедлением

5 Алгоритм решения MSC

Заметим следующие вещи:

1. Задачу MSC можно задать, дав только S, а X восстанавить уже в алгоритме
2. Если |S| = 0, то ответ 0
3. Если $\exists X, Y \in S : X \subset Y \Rightarrow \exists$ SC без X
4. Если $\exists u \in U(S) : \exists! X \in S : u \in X \Rightarrow \forall$ SC содержит X
5. $\forall X \in S$ "подходит"только на 1 из пунктов (3), (4)

Так же заметим, что если все множества мощности 2, то MSC сводима к поиску наименьшего рёберного покрытия, а тот в свою очередь сводится к поиску максимального паросочетания:

$$S = \{S_1, ..., S_n\}$$

$\forall x \in U(S)$ заведём вершину u в графе, а $\forall S_i = \{u, v\}$ заведём ребро uv
Сначала находится максимальное паросочетание (конкретно у меня через алгоритм сжатия цветков),

а потом для каждой вершины, не имеющей пары возьмём любое инцидентное ей ребро, так получим MSC

В таком случае алгоритм следующий

```
set del(S, X) {
    return {Y|Y = Z\X ≠ ∅, Z ∈ S};
}

int msc(S) {
    if(|S| = 0)
        return 0;
    if(∃X, Y ∈ S : X ⊂ Y)
        return msc(S\{X});
    if(∃u ∈ U(S)∃!X ∈ S : u ∈ X)
        return 1+msc(del(S, X));

    берём X ∈ S максимальной мощности;
    if(|X| = 2)
        return msc2(X);
    return min{msc(S\{X}), 1+msc(del(S, X))};
}
```

Заявляется, что такой алгоритм находит MSC за $O^*(2^{0.305(|S|+|U(S)|)})$ (2 в списке литературы, пункты 3.1, 3.2)

Так как мы решаем MDS, то в силу описанного сведения |S| + |U(S)| = n + n = 2n, то MDS работает за $O^*(2^{0.61n}) = O^*(1.5263^n)$

Примечание:

В моей реализации алгоритма возвращается не величина ответа, а набор подмножеств которые образуют минимальное покрытие

6 Тесты

Не уделяя особого внимания Unit тестам, которые проверяют минимальную работоспособность программы, перейдём к стресс тестам

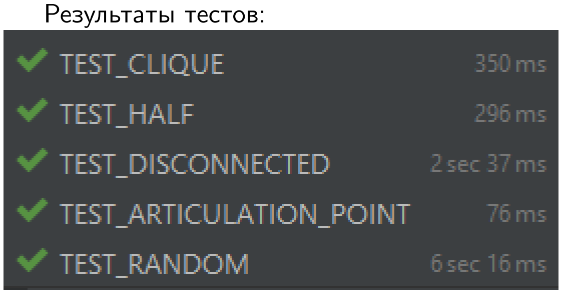
Во всех стресс тестах кроме теста RANDOM в цикле создаются графы на i вершинах, где i от 5 до 120, в RANDOM от 5 до 45.

1. Для проверки части алгоритма с выбрасыванием множества, если он является подмножеством другого есть тест, где граф выглядит как вершина, которая соединена со всеми остальными, а других рёбер нет.

2. Для проверки части, где в ответ берётся множество, которое содержит уникальную вершину есть тесты на несвязные графы и клики

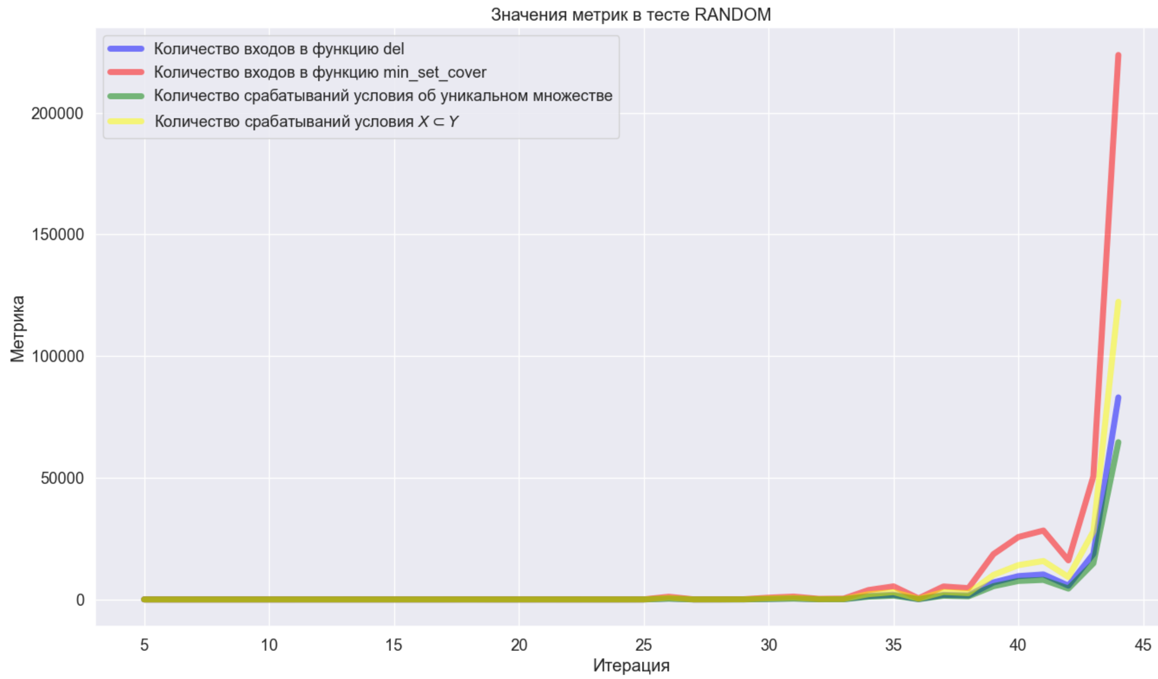
3. Так же есть просто тест где рёбра выбираются случайно(шанс появления ребра 20%)

Результаты тестов:

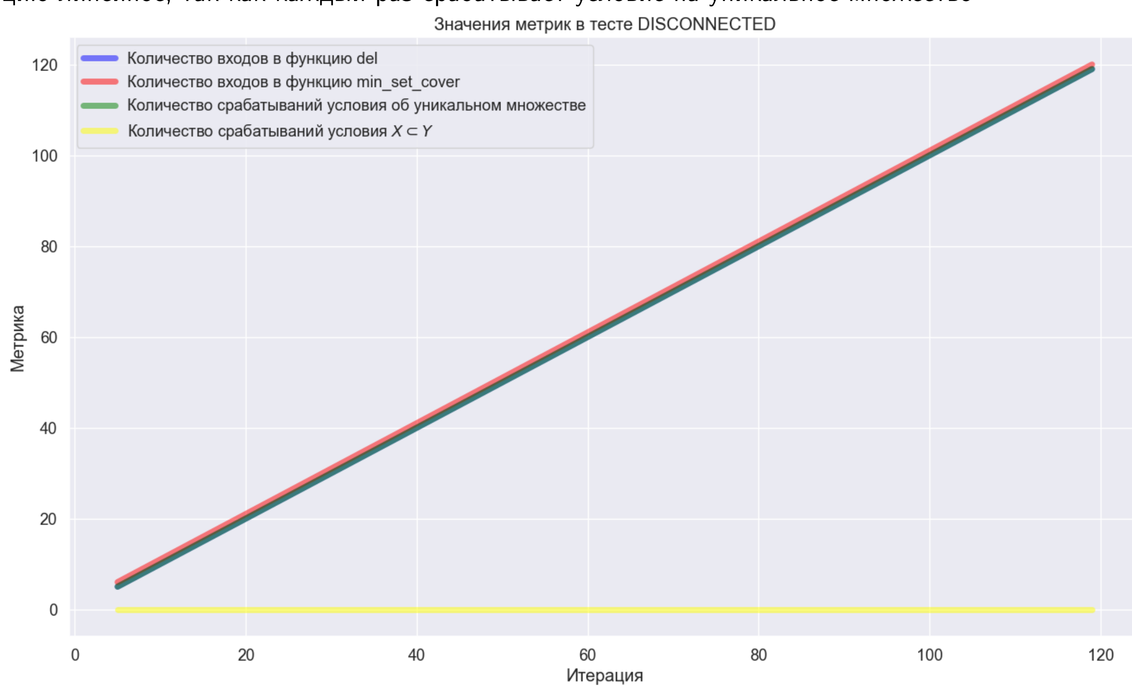


Посмотрим подробнее на тесты:

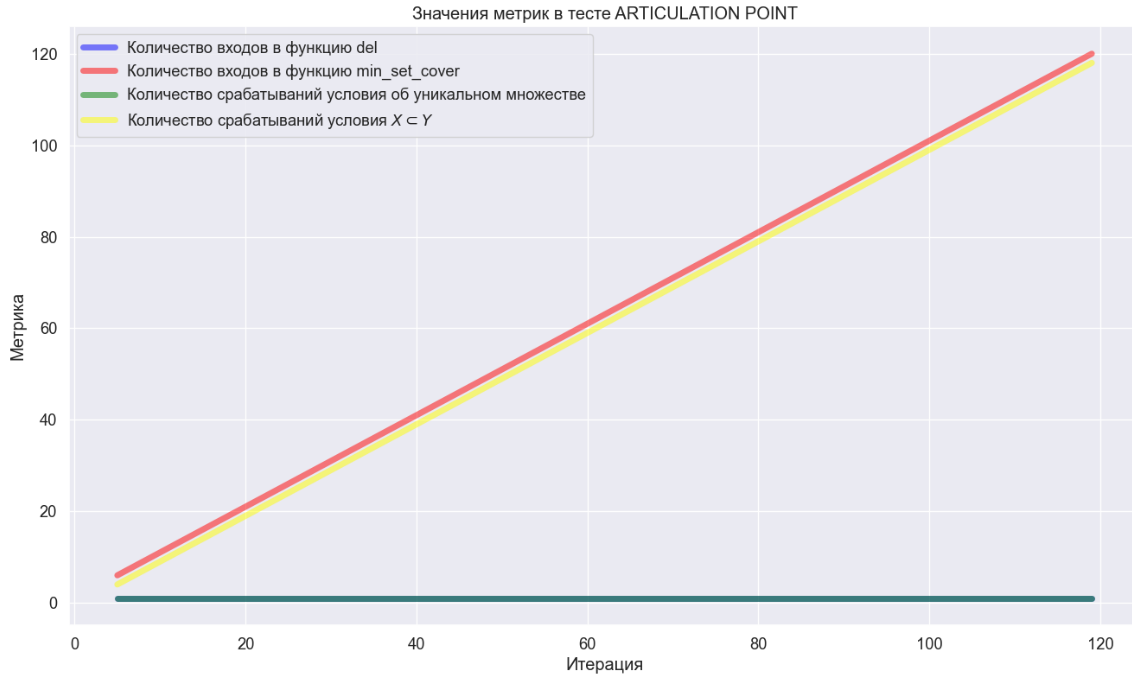
1. Случайный тест демонстрирует экспоненту в асимптотике



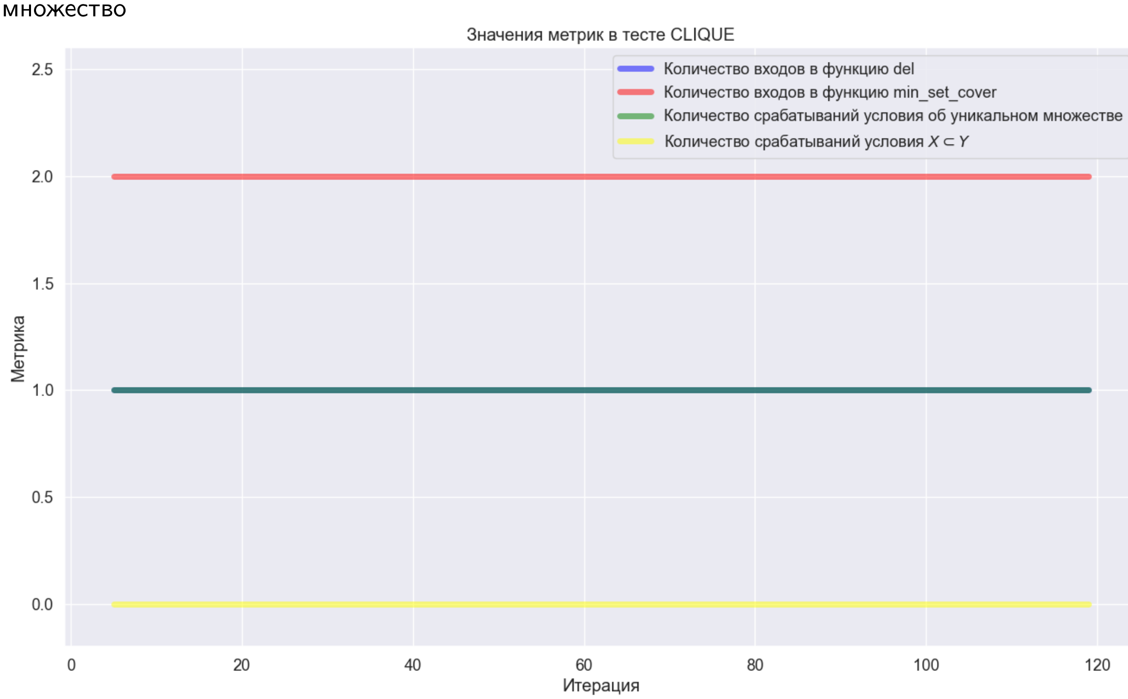
2. Тест на граф без рёбер работает намного быстрее, причём количество входов в основную функцию линейное, так как каждый раз срабатывает условие на уникальное множество



3. Тест на точку сочленения работает быстрее всех, каждый раз убираются всех множества размера 2 и остаётся только множество с точкой сочления и всеми соседями



4. Тест на клику работает с константным количеством входов в функцию, так как перед первым заходом в функцию min_set_cover одинаковые множества убираются, потому там всегда одно множество



Список литературы

[1] On the Approximability of NP-complete Optimization Problems, Viggo Kann

[2] A Measure & Conquer Approach for the Analysis of Exact Algorithms, FEDOR V. FOMIN, FABRIZIO GRANDONI, DIETER KRATSCH