Взвешенная задача вершинного покрытия и алгоритмы аппроксимации

Бруёк Алексей, Б05-022

Аннотация

В этой работе описан жадный алгоритм 2-приближения минимального вершинного покрытия с доказательством корректности и результатами тестирования. Основным источником стала книга [2].

1 Упрощенные постановки задачи

1.1 Невзвешенный случай

Определение 1. *Минимальным вершинным покрытием называется наименьшее по мощности подмножество вершин графа, являющееся вершинным покрытием графа.*

Задача 1. *Найти вершинное покрытие, мощности не более чем в 2 раза большей, чем у минимального за полиномиальное время. (Полиномиальный алгоритм 2-приближения)*

Очевидно, что найдя максимальное по включению паросочетание, мы сможем выбрать вершины, входящие в это паросочетание и получить вершинное покрытие. При этом минимальное вершинное покрытие должно содержать по крайне мере по одной вершине из каждого из этих ребер, поэтому мы не ухудшим ответ более чем в 2 раза.

1.2 Функция весов пропорциональна deg

Определение 2. *Назовем функцию весов $w : V \to \mathbb{Q}^+$ пропорционально-степенной, если существует такая константа $c \in \mathbb{Q}^+$, что*

$$\forall v \in V : w(v) = c \cdot deg(v)$$

где $deg(v)$ – степень вершины v.

Определение 3. *Минимальным вершинным покрытием во взвешенном случае называем вершинное покрытие минимального веса.*

Задача 2. *Дан граф G с пропорционально-степенной функцией весов w, нужно найти 2-приближение минимального вершинного покрытия за полиномиальное время.*

Лемма 1. *Если w пропорционально-степенная функция весов и OPT – минимальное вершинное покрытие, то*

$$w(V) \leq 2 \cdot w(OPT)$$

Доказательство.

$$w(V) = \sum_{v \in V} c \cdot deg(v) = c \sum_{v \in V} deg(v) = 2c|E|$$

$$w(OPT) = \sum_{v \in OPT} c \cdot deg(v) = c \sum_{v \in OPT} deg(v)$$

Так как для каждого ребра, хотя бы один из его концов принадлежит OPT, имеем

$$\sum_{v \in OPT} deg(v) \geq |E|$$

В итоге,

$$2w(OPT) \geq 2c|E| = w(V)$$

□

Следствие 1. *В случае пропорционально степенной функции весов, веса любых 2-х вершинных покрытий отличаются не более чем в 2 раза.*

2 Основная задача

Задача 3. *Дан граф G и произвольная функция весов w. Требуется найти 2-приближение минимального вершинного покрытия за полиномиальное время.*

Algorithm 1 GreedyAlgo

Require: $G = (V, E)$, $w : V \to \mathbb{Q}^+$ – функция весов
  $S \leftarrow \emptyset$ – итоговое вершинное покрытие
  while $V \neq \emptyset$ **do**
    $c \leftarrow \min_{v \in V} \frac{w(v)}{deg(v)}$
    $t(v) \leftarrow c \cdot deg(v)$ –
    $w \leftarrow w - t$
    $D \leftarrow \{v \in V : w(v) = 0\}$
    $S \leftarrow S \cup D$ – добавляем вершины в ответ
    $V \leftarrow V \backslash D$ – и удаляем их из графа
    $V \leftarrow V \backslash \{v \in V : deg(v) = 0\}$
  end while
  return S

Краткое описание алгоритма: На каждой итерации:

- находим максимальную пропорционально-степенную функцию весов t, которая не превосходит w на всех вершинах и вычитаем ее из w.
- Вершины вес которых стал нулевым добавляем в ответ и удаляем из графа.
- Вершины с нулевымы степенями просто удаляем

Очевидно, алгоритм удаляет вершины на каждом шаге, а значит он закончится и вернет ответ S.
Убедимся в корретности ответа

Лемма 2. *Множество вершин S полученное алгоритмом (1) является вершинным покрытием.*

Доказательство. Пусть $(u, v) \in E$ и вершины u, v не вошли в S.
Тогда u, v должны были быть удалены как вершины с нулевой степенью, что невозможно. □

Теперь докажем, что 1 является алгоритмом 2-приближения.
Введем обозначения:

- t_i – пропорционально-степенная функция весов, полученная на i-ой итерации
- $G_i = (V_i, E_i)$ – граф перед i-ой итерацией.
- S – ответ алгоритма
- OPT – настоящее минимальное вершинное покрытие
- k – количество итераций алгоритма
- w – исходная функция весов

Лемма 3.

$$\forall i \in \mathbb{N} \cap [1, k] \, \forall j \in \mathbb{N} \cap [j, k] : t_i(G_j \cap S) \leq 2t_i(G_j \cap OPT)$$

Доказательство. Зафикцируем $0 < i \leq j \leq k$.
Так как $V_j \subset V_i$, то t_i определена на V_j. Также из того, что OPT и S – вершинные покрытия G следует, что $OPT \cap G_j$ и $S \cap G_j$ – вершинные покрытия G_j. Остается применить лемму 1. и получить требуемый результат. □

Теорема 1. *Алгоритм 1 является алгоритмом 2-приближения.*

Доказательство. Пусть $v \in S$ и v была добавлена в S на i-ом шаге, тогда верно равенство

$$w(v) = \sum_{j \le i} t_j(v) = \sum_{j : v \in G_j} t_j(v)$$

Теперь $v \in OPT \backslash S$, и v была удалена на i-ом шаге, тогда

$$w(v) \ge \sum_{j \le i} t_j(v) = \sum_{j : v \in G_j} t_j(v)$$

Переходя от отдельных вершин к множествам:

$$w(S) = \sum_{v \in S} \sum_{j : v \in V_j} t_j(v) = \sum_{j \le k} \sum_{v \in V_j} t_j(v) = \sum_{j \le k} t_j(S \cap V_j)$$

$$w(OPT) \ge \sum_{v \in OPT} \sum_{j : v \in G_j} t_j(v) = \sum_{j \le k} t_j(G_j \cap OPT)$$

Пользуясь леммой 3 получаем

$$w(S) = \sum_{j \le k} t_j(S \cap V_j) \le 2 \sum_{j \le k} t_j(G_j \cap OPT) \le 2w(OPT)$$

чтд. □

3 Альтернативный алгоритм

Для сравнения приведем еще один алгоритм 2-приближения минимильного вершинного покрытия без доказательства. Алгоритм взят из лекции [1], а также реализован в библиотеке $networkx$.

Algorithm 2 Bar-Yehuda and Even's Greedy Algorithm

```
S ← ∅
while E ≠ ∅ do
    e = (u, v) ∈ E – берем одно любое ребро из E
    if w(u) > w(v) then
        (u, v) ← (v, u)
    end if
    w(v) ← w(v) − w(u)
    S ← S ∪ {u}
    V ← V\{u}
end while
return S
```

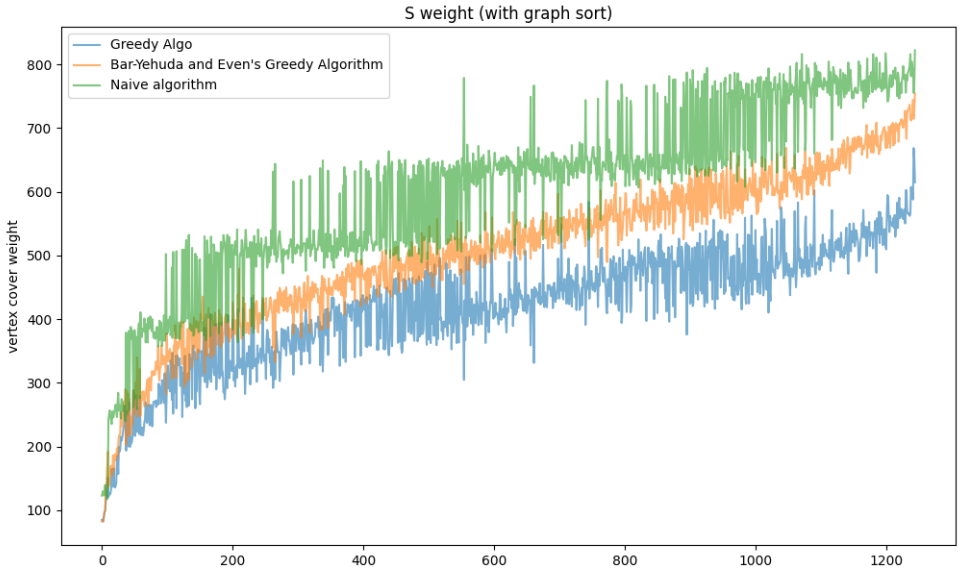
Также будем рассматривать наивный алгоритм, игнорирующий веса.

4 Тестирование

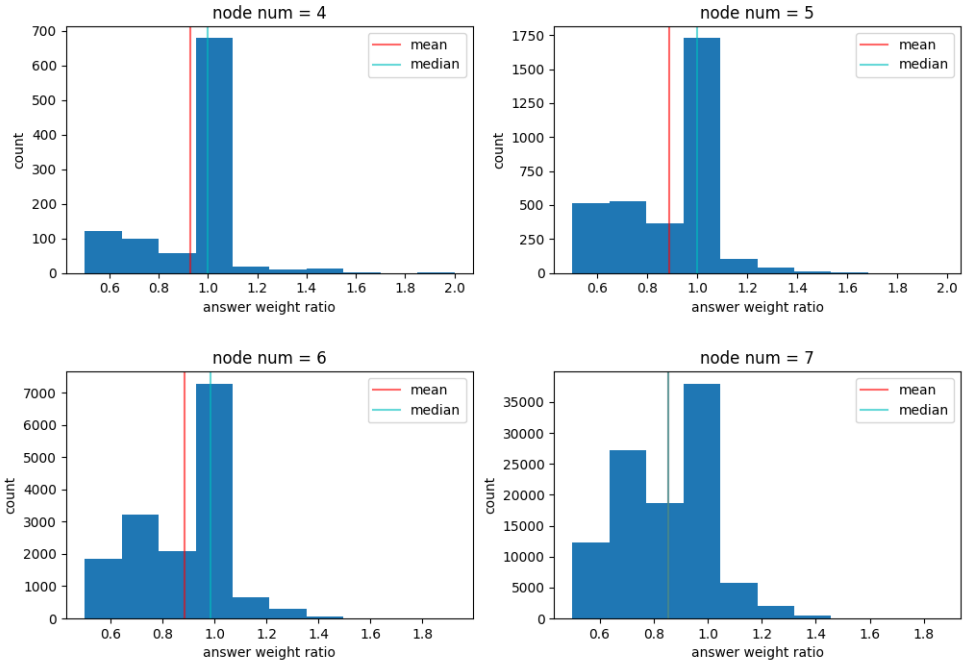
Веса вершин будем считать случайным целым числом от 1 до 256.

4.1 Простые графы с ≤ 7 вершин

Рассмотрим все неизоморфные графы на не более чем 7 вершинах. Для каждого сгенерируем 100 взвешенных графов и протестируем.
Если усреднить результаты на этих 100 графах и отсортировать графы можно получить представление о весах возвращаемых вершинных покрытий:



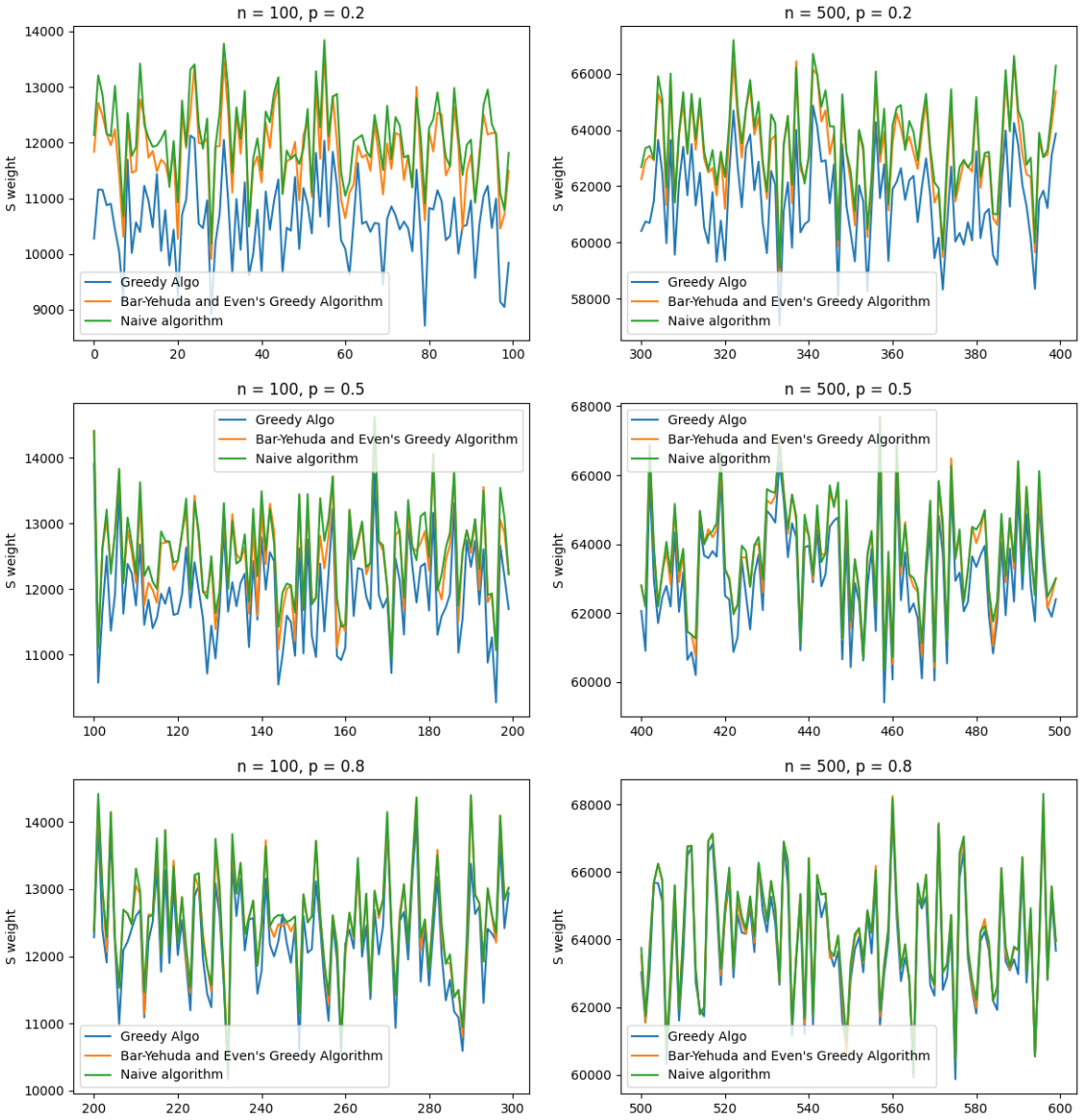
Из графика видно, что в большинстве случаев алгоритм 1 работает лучше других, а наивный – хуже.
Для детального сравнения алгоритмов 1 и 2 будем считать отношение весов ответов на одном и том же взвешенном графе и примерное распределение этой величины:



Нетрудно заметить, что это отношение всегда находится между 0.5 и 2 (оба алгоритма дают 2-приближение минимального вершинного покрытия), а также, что в большинсве случаев (медиана) алгоритм 1 дает лучший ответ, чем 2.

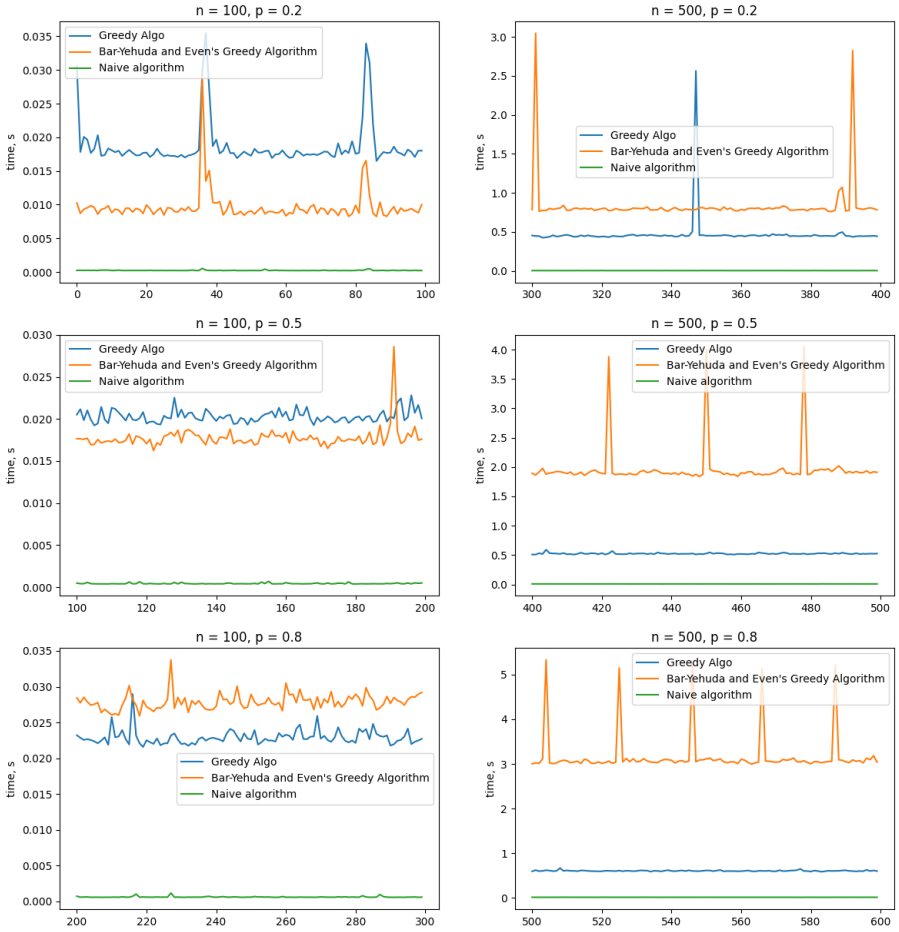
4.2 Случайные графы $G(n, p)$

Тут построим большие случайные графы и проведем те же наблюдения, а также посмотрим на время исполнения.



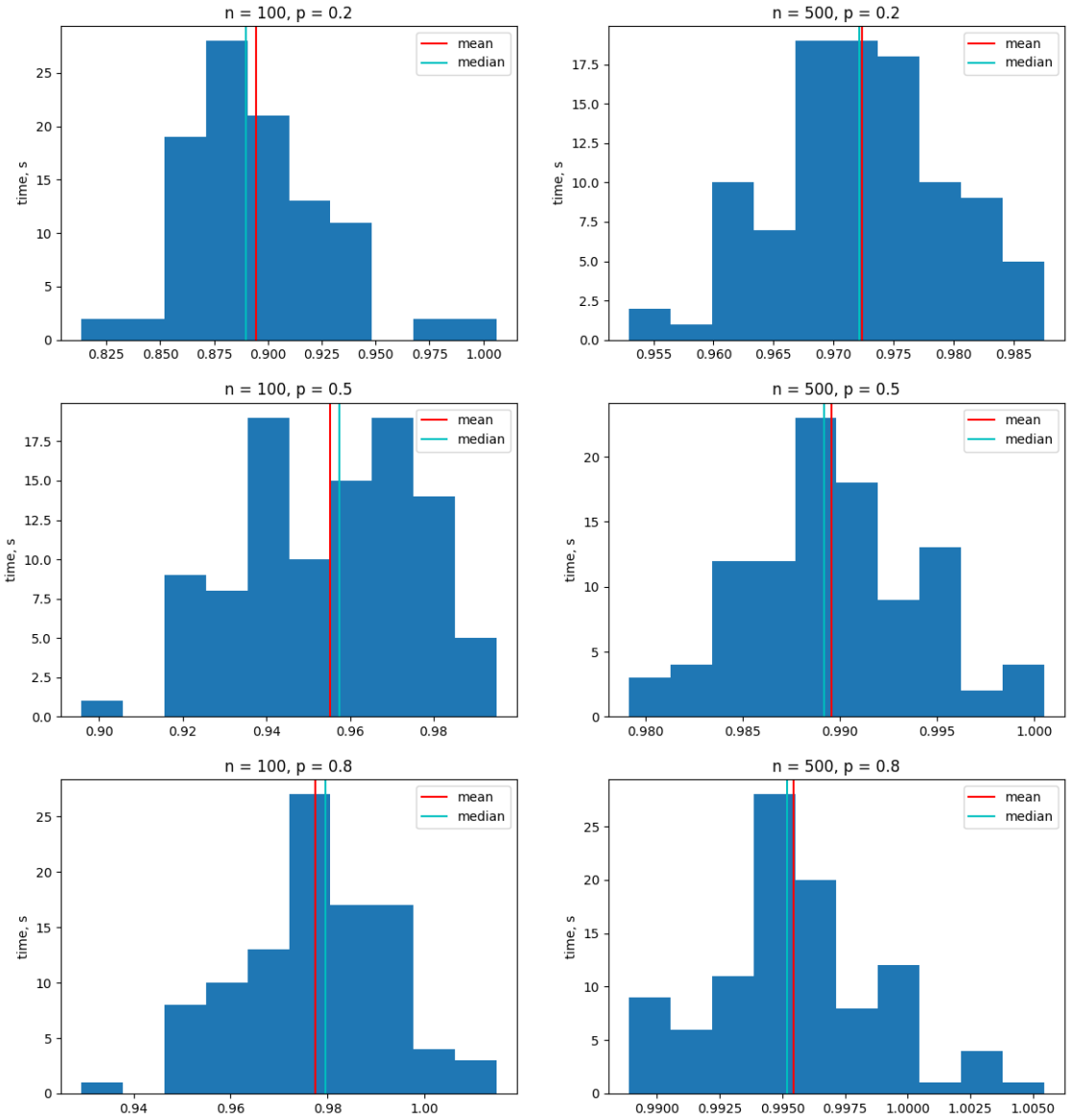
Тут можно отметить, что с увелиением плотности графа или количества вершин алгоритмы начинают вести себя почти одинаково. Случай $n = 500, p = 0.8$ анализировать вообще бессмысленно.

Давайте построим аналогичные графики по времени.



Видно, что наивный алгоритм всегда работает быстрее (неудивительно). Между двумя другими алгоритмами разница несущественна, и может быть списана на реализацию.

Выведем отношение весов ответов алгоритмов 1 и 2

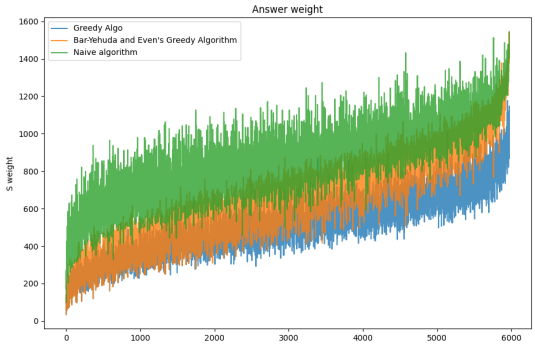


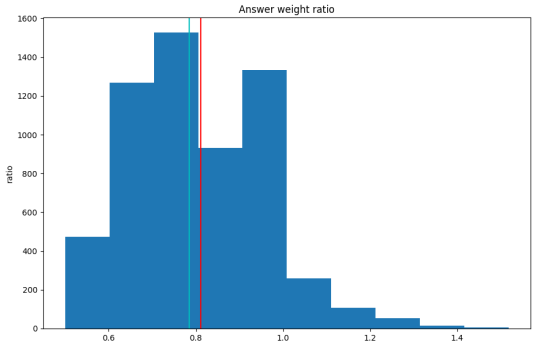
Тут статистика соответствует полученным ранее результатам. А именно, при большой плотности ребер и количестве вершин, результаты алгоритмов очень близки. Однако алгоритм 1 все же почти всегда немного лучше 2.

4.3 Другие виды графов

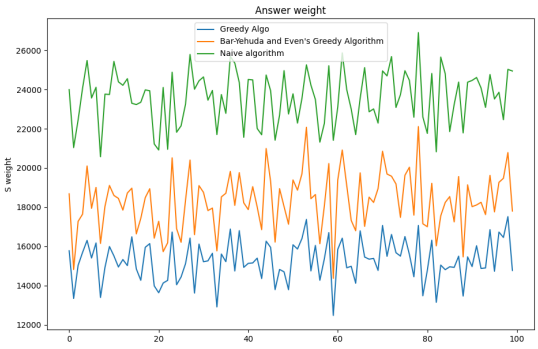
4.3.1 Планарные

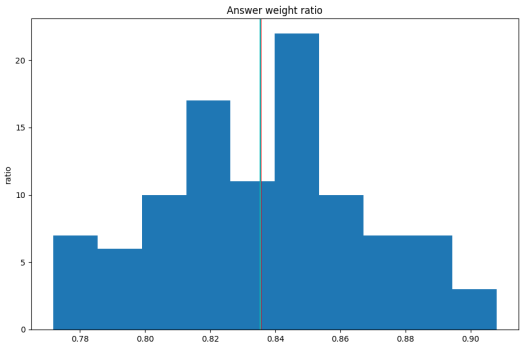
Данные были взяты с [3].





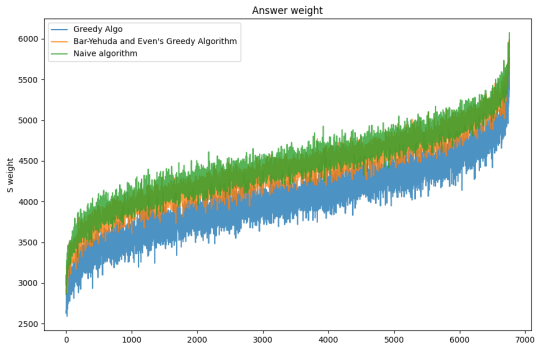
4.3.2 Деревья

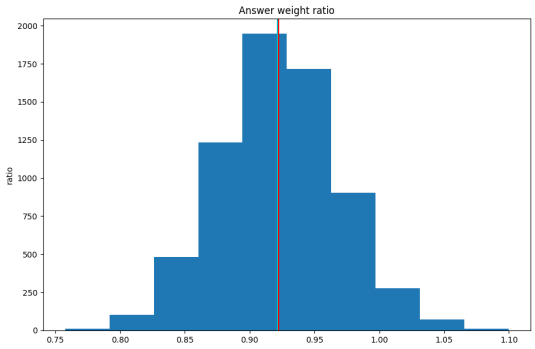




4.3.3 Сильно регулярные

Данные были взяты с [3].





Во всех трех случаях отмеченные закономерности также работают.

5 Выводы

Построенный алгоритм 2-приближения является достаточно эффективным в среднем даже в сравнении с другим алгоритмом 2-приближения.

Список литературы

[1] Samir Khuller, Advanced Algorithms, Lectures 4 and 5. (`https://www.cs.umd.edu/class/fall2018/cmsc858E/pdfs/651/vc.pdf`)

[2] А.В.Кононов, П.А.Кононова, Приближенные алгоритмы для решения NP-Трудных задач, (http://old.math.nsc.ru/LBRT/k5/Kononov/Kononovs_teaching_book.pdf)

[3] Примеры графов, (http://users.cecs.anu.edu.au/~bdm/data/graphs.html)