Сложности вычислений. Полиномиальный алгоритм раскраски 3-раскрашиваемого графа в $O(\sqrt{n})$ цветов

Упшинский Андрей, 495

Декабрь 2016

1 Вступление

Определение 1 *Раскраска графа называется *правильной*, если ни одно ребро не соединяет вершины одного цвета.*

Как известно, задача проверки графа на существование правильной раскраски в k цветов (а также ее поиска) при $k > 2$ является NP-полной. Поэтому интересно существование полиномиальных алгоритмов, частично решающих задачу.

Лемма 1 *Проверка графа на двудольность и нахождение раскраски в 2 цвета лежат в* $\mathbf{P}$.

Доказательство В силу того, что есть всего два цвета, цвет одной вершины определяет цвета всех остальных в ее компоненте связности. Значит, достаточно попытаться раскрасить. Если будет противоречие, то раскрасить нельзя. Если нет, то есть раскраска.

2 Жадный алгоритм

Определение 2 **Жадным* назовем следующий алгоритм:*

1. *Рассмотрим произвольную вершину.*
2. *Покрасим ее в минимальный цвет из незанятых соседями.*
3. *Продолжаем, пока остаются непокрашенные вершины.*

Доказательство Покажем его корректность. Действительно, на каждой итерации алгоритма что-то меняется только для закрашиваемой вершины и ее соседей. Тем, что выбираем незанятый цвет, мы гарантируем корректность раскраски на каждом шаге. Тогда и итоговая раскраска будет корректной.

Лемма 2 *Пусть максимальная степень вершины в графе d. Тогда жадный алгоритм раскрасит граф в не более чем $d+1$ цвет.*

Доказательство Среди первых $d+1$ цвета всегда найдется такой, что он не встречается у соседей, которых не больше d. Лемма доказана.

3 Алгоритм Вигдерсона

Теорема 1 *Раскраска 3-раскрашиваемого графа в $O(\sqrt{n})$ цветов возможна за полиномиальное время.*

Определение 3 *Рассмотрим следующий алгоритм:*

1. Посмотрим на вершину v с максимальной степенью в графе G.

2. Если $deg(v) >= \sqrt{n}$, рассмотрим отдельно граф G_v из вершины и ее соседей. Так как G 3-раскрашиваемый, то G_v 3-раскрашиваемый и при удалении v соседи могут быть покрашены в 2 цвета. Тогда красим соседей в 2 цвета, вершину v в третий и больше не используем эти цвета. Удаляем вершину v вместе с соседями и переходим обратно к итерации 1.

3. Если $deg(v) < \sqrt{n}$, запускаем жадный алгоритм на оставшихся вершинах.

Это и есть алгоритм Вигдерсона [1]. Покажем его корректность и полиномиальность времени работы.

Доказательство корректности: Несложно заметить, что алгоритм строит правильную раскраску. Покажем, что цветов $O(\sqrt{n})$.

В каждой итерации 2 добавляется максимум 3 цвета и убирается минимум $\sqrt{n}$ вершин. Значит таких итераций не может быть больше $\sqrt{n}$ и добавится не более чем $3 \cdot \sqrt{n}$ цветов. Жадный алгоритм раскрасит оставшиеся вершины в максимум $\sqrt{n}$ цветов, то есть всего будет не более $4 \cdot \sqrt{n}$ цветов. Корректность доказана.

Доказательство полиномиальности времени работы: Предложим полиномиальную реализацию. Подробнее - см. [2]

1. Поиск максимальной степени вершины работает за количество ребер

2. Раскраска графа G_v работает не более чем за количество ребер

То есть одна итерация цикла работает за количество ребер, которое не превосходит квадрата количества вершин. Итераций по доказанномуне больше $\sqrt{n}$, а значит цикл работает за полином

3. Здесь запускается жадный алгоритм. Он работает за количество ребер

```
def get_coloring(graph):
    k = max(int(len(graph) ** 0.5), 2)
    colors = [0] * len(graph)
    min_color = 1

    while True:
        # 1.
        max_d, v = _get_maximal_degree(graph, colors)
        if max_d < k:
            break

        vertices = set()
        colors[v] = min_color
        min_color += 1

        for u in graph[v]:
            if colors[u] == 0:
                vertices.add(u)

        # 2.
        _coloring(graph, vertices, colors, min_color, _bin_dfs)
        min_color = max(colors) + 1

    # 3.
    vertices = set()
    for v in range(len(graph)):
        if colors[v] == 0:
            vertices.add(v)

    _coloring(graph, vertices, colors, min_color, _greed_dfs)

    return colors
```

Таким образом в сумме алгоритм работает за $O(n^2\sqrt{n})$. Полиномиальность доказана.

4 Заключение

Существуют и более опримальные приближения, однако они требуют более громоздкой теории. Этот алгоритм хорош простотой реализации и довольно неплохими результатами работы (тестирование алгоритма - см. [2]).

Список литературы

[1] Avi Wigderson. *Improving the Performance Guarantee for Approximate Graph Coloring.* J. of the ACM, 1983. URL `https://sachdevasushant.github.io/spring15/presentations/charles-coloring.pdf`.

[2] Andrey Upshinskiy. Implementation and testing of widgerson's algorithm. URL `https://github.com/egiby/WidgersonAlgorithm`.