Федеральное государственное автономное образовательное учреждение
высшего образования

«Национальный исследовательский университет

«Высшая школа экономики»

Факультет компьютерных наук
ООП «Прикладная математика и информатика»

Отчёт по семинарскому заданию №1.

Студент: Кобелев Максим Олегович

Группа: 165

Место выполнения: НИУ ВШЭ

Наставник:

Саакян Вильям Рустамович.

Москва, 2017

Оглавление

1. Реализация методов поиска максимального разреза

1.1 Эвристический алгоритм

Работа подразумевала собой реализацию эвристического алгоритма поиска максимального разреза. Эвристика алгоритма заключается в его жадности: на каждом шаге алгоритма мы пытаемся «улучшить» наш разрез, то есть увеличить его величину посредством замены текущего разреза на лучшего из его соседей. Соседний разрез здесь – разрез, отличающийся от исходного ровно на одну вершину. Понятно, что под лучшим соседом подразумевается разрез, с единственно-перекинутой вершиной, размер которого максимален. В этом и кроется улучшение простого жадного алгоритма: мы выбираем не первый попавшийся улучшающий разрез, а лучший из всевозможных улучшающий. Такой алгоритм завершит свою работу за конечное число шагов (в т.ч. очень большое) в тот момент, когда не останется улучшающих соседей, или же другими словами не останется вершин, увеличивающих дельту размеров наших разрезов.

1.2 Алгоритм отжига с понижением температуры согласно функции Коши

Плюс жадного алгоритма в том, что он находит достаточно близкое к истинному решение, однако делает это за длительное время. Алгоритм отжига был призван чтобы справляться с задачей долгого поиска максимального разреза эвристическими алгоритмами. Если в жадном алгоритме переход к «улучшающему» разрезу осуществлялся только при увеличении дельты, то в алгоритме отжига присутствует также ещё один параметр – температура. Обозначим их за:

$$\partial - \text{разность разрезов улучшающего и исходного;}$$

$$T_0 - \text{начальная температура отжига;}$$

$$T_i - \text{температура на } i - \text{ой итерации.}$$

Используя оба параметра переход к «улучшенному» разрезу осуществляется в нескольких случаях. Например, если существует хотя бы один «улучшающий сосед», то мы просто заменяем текущий разрез на него, причем на первый попавшийся, не боясь угодить в локальный минимум из-за следующего условия. Иначе, если вершина не улучшающая, мы переходим к другому разрезу используя некоторый вероятностный подход для перекидывания вершины, согласно формуле:

$$\exp\left(\frac{\partial}{T_i}\right) \quad (1)$$

Если же перекидывание снова не произошло, то просто переходим к следующей рассматриваемой вершине. Так как вершина в вероятностном перекидывании не является улучшающей, то $\partial < 0$, в следствии чего (1) всегда есть величина, не превосходящая 1, что доказывает корректность использования именно вероятностного подхода. На каждой итерации происходит уменьшение температуры – охлаждение разреза, что позволяется повышать вероятность выхода из локального минимума при помощи неулучшающей вершины. Сама же температура будет меняться по следующему правилу, именуемому также как метод Коши:

$$T_{i+1} = \frac{T_i}{i}, \text{где } i - \text{номер итерации алгоритма.}$$

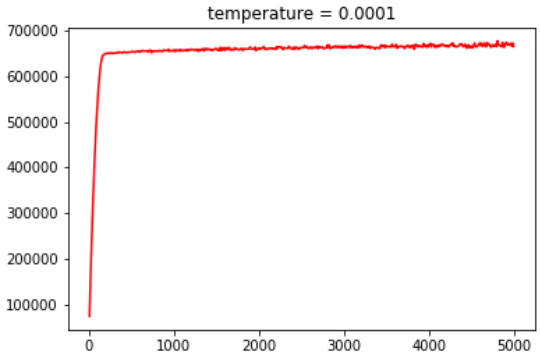
Важной особенностью этого алгоритма является то, что мы можем менять количество итераций, необходимых для его выполнения. Однако от этого будет также меняться ответ. Попробуем

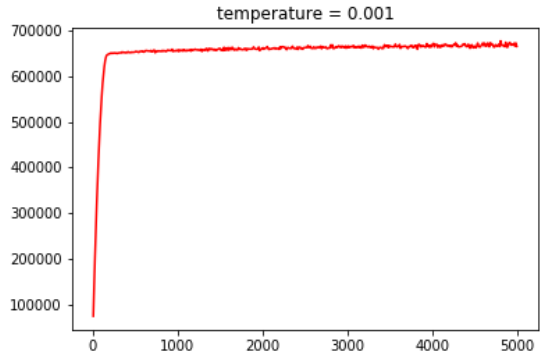
поисследовать этот момент и понять при каких оптимальных температуре и числе итераций отжиг будет возвращать наиболее приближенный к оптимальному ответ.

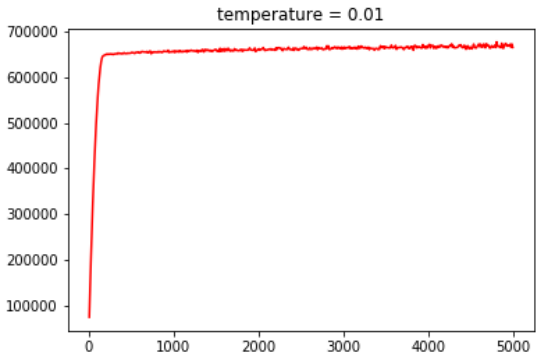
2. Графики зависимостей и поиск оптимальной температуры при помощи сравнения методов и варьирования параметров

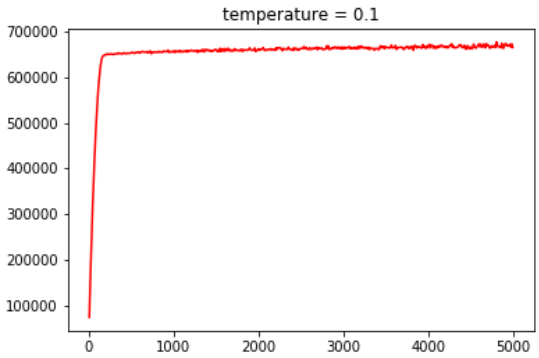
В этой главе предлагается ознакомиться с различными графиками зависимостей сумм размеров найденных разрезов в заданных графах от количества итераций (до 5000). Далее, основываясь на визуальных результатах алгоритма отжига – выбрать наиболее оптимальную температуру. Графы будут случайно сгенерированы при помощи специальных рандомизированных конструкторов, в которых существование ребра зависит от случайной величины, которую в свою очередь зависит от произвольно выбранного rand_seed. Для анализа, в качестве базы для запуска алгоритмов отжига и визуализации результатов было использовано 20 случайных графов, 20 двудольных графов и 10 графов звёзд. Все графы состояли из 350 вершин, вероятность ребра между любыми двумя < 0,7.

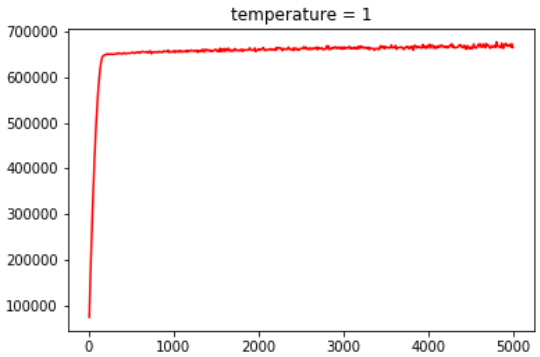
2.1 Графики зависимостей сумм разрезов на разных температурах

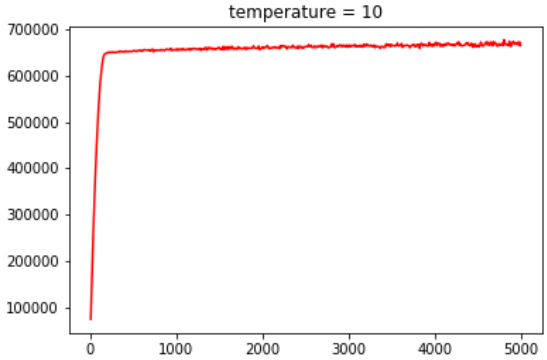


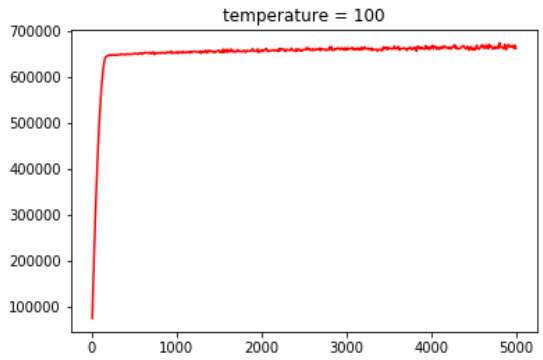


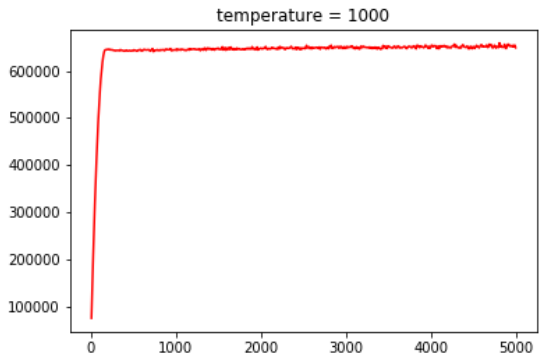


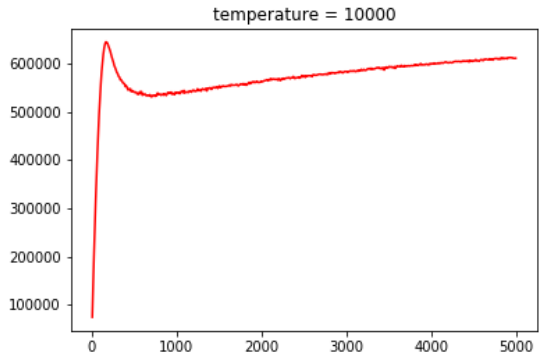












Интересно то, что графики для начальной температуры ниже чем 10 – практически совпадают. Скорее всего дело в том, что при таких низких значениях стартовой температуры, итерационная температура в среднем дает уже достаточно хорошее значение вероятности для выхода из «локального минимума». В свою очередь начальная температура 1000 демонстрирует небольшую просадку в районе 400 итераций алгоритма, а начальная температура 10000 дает достаточно глубокую просадку в районе 1500 итераций. При этом на таких высоких значениях температуры алгоритм показывает более худший результат максимального разреза, по сравнению с отжигом на стартовых температурах ниже 100. А сумма результатов отжига на низких стартовых температурах вообще постоянна. В качестве доказательства продемонстрируем следующие тесты.

2.2 Выбор оптимальной начальной температуры для алгоритма отжига

Некоторые примеры тестирования с указанием всех параметров запуска, включая случайный параметр rand_seed, а также результата запуска жадного алгоритма на том же графе, состоящем из 250 вершин, для наиболее обоснованного выбора оптимальной температуры.

Using rand seed: 1507491918 Number of vertexes is: 200 Iterations: 5000
Greedy algorithm Result: 289568
temperature: 0.0001 Result: 222501 distance between solvers: 67067
temperature: 0.001 Result: 222501 distance between solvers: 67067
temperature: 0.01 Result: 222501 distance between solvers: 67067
temperature: 0.1 Result: 222501 distance between solvers: 67067
temperature: 1 Result: 222501 distance between solvers: 67067
temperature: 10 Result: 222501 distance between solvers: 67067
temperature: 100 Result: 221029 distance between solvers: 68539
temperature: 1000 Result: 207888 distance between solvers: 81680
temperature: 10000 Result: 204279 distance between solvers: 85289

Using rand seed: 1507491959 Number of vertexes is: 250 Iterations: 5000
Greedy algorithm Result: 450802
temperature: 0.0001 Result: 349546 distance between solvers: 101256
temperature: 0.001 Result: 349546 distance between solvers: 101256
temperature: 0.01 Result: 349546 distance between solvers: 101256
temperature: 0.1 Result: 349546 distance between solvers: 101256
temperature: 1 Result: 349546 distance between solvers: 101256
temperature: 10 Result: 349546 distance between solvers: 101256
temperature: 100 Result: 348137 distance between solvers: 102665

temperature: 1000 Result: 331239 distance between solvers: 119563
temperature: 10000 Result: 317657 distance between solvers: 133145

Using rand seed: 1507491824 Number of vertexes is: 300 Iterations: 5000
Greedy algorithm Result: 647555
temperature: 0.0001 Result: 492415 distance between solvers: 155140
temperature: 0.001 Result: 492415 distance between solvers: 155140
temperature: 0.01 Result: 492415 distance between solvers: 155140
temperature: 0.1 Result: 492415 distance between solvers: 155140
temperature: 1 Result: 492415 distance between solvers: 155140
temperature: 10 Result: 492415 distance between solvers: 155140
temperature: 100 Result: 491476 distance between solvers: 156079
temperature: 1000 Result: 475355 distance between solvers: 172200
temperature: 10000 Result: 450459 distance between solvers: 197096

Using rand seed: 1507491585 Number of vertexes is: 350 Iterations: 5000
Greedy algorithm Result: 880013
temperature: 0.0001 Result: 670366 distance between solvers: 209647
temperature: 0.001 Result: 670366 distance between solvers: 209647
temperature: 0.01 Result: 670366 distance between solvers: 209647
temperature: 0.1 Result: 670366 distance between solvers: 209647
temperature: 1 Result: 670366 distance between solvers: 209647
temperature: 10 Result: 670366 distance between solvers: 209647
temperature: 100 Result: 669338 distance between solvers: 210675
temperature: 1000 Result: 653581 distance between solvers: 226432
temperature: 10000 Result: 613275 distance between solvers: 266738

Видно, что результат работы алгоритмов с различными температурами не зависит от количества вершин графа. Поведение алгоритма не зависит от размера графа

2.3 Примеры сравнения

Некоторые примеры для сравнения двух алгоритмов можно наблюдать в примерах тестирований, приведенных выше, на графах, с различным числом вершин. Была выдвинута теория о том, что алгоритм отжига показывает себя достаточно плохо, даже в сравнении с эвристическим жадным алгоритмом, находящим лучший ответ. Для её опровержения был предпринят метод увеличения количества итераций для алгоритма отжига. Как видно из приведенных ниже тестов отжиг действительно начал работать лучше и при большем количестве итераций решил выдавать результат, более приближенный к результату работы жадного алгоритма. Однако точность (близость к жадному) растет нелинейно относительно увеличения количества итераций.

1000 iterations
Using rand seed: 1507496563 Number of vertexes is: 250
Greedy algorithm Result: 450184
temperature: 0.0001 Result: 338998 distance between solvers: 111186
temperature: 0.001 Result: 338998 distance between solvers: 111186
temperature: 0.01 Result: 338998 distance between solvers: 111186
temperature: 0.1 Result: 338998 distance between solvers: 111186
temperature: 1 Result: 338998 distance between solvers: 111186
temperature: 10 Result: 338935 distance between solvers: 111249
temperature: 100 Result: 337769 distance between solvers: 112415
temperature: 1000 Result: 327249 distance between solvers: 122935
temperature: 10000 Result: 225153 distance between solvers: 225031

5000 iterations
Using rand seed: 1507491959 Number of vertexes is: 250
Greedy algorithm Result: 450802
temperature: 0.0001 Result: 349546 distance between solvers: 101256
temperature: 0.001 Result: 349546 distance between solvers: 101256
temperature: 0.01 Result: 349546 distance between solvers: 101256
temperature: 0.1 Result: 349546 distance between solvers: 101256
temperature: 1 Result: 349546 distance between solvers: 101256
temperature: 10 Result: 349546 distance between solvers: 101256
temperature: 100 Result: 348137 distance between solvers: 102665
temperature: 1000 Result: 331239 distance between solvers: 119563
temperature: 10000 Result: 317657 distance between solvers: 133145

20000 iterations
Using rand seed: 1507494524 Number of vertexes is: 250
Greedy algorithm Result: 450486
temperature: 0.0001 Result: 353112 distance between solvers: 97374

temperature: 0.001 Result: 353112 distance between solvers: 97374
temperature: 0.01 Result: 353112 distance between solvers: 97374
temperature: 0.1 Result: 353112 distance between solvers: 97374
temperature: 1 Result: 353112 distance between solvers: 97374
temperature: 10 Result: 353112 distance between solvers: 97374
temperature: 100 Result: 352588 distance between solvers: 97898
temperature: 1000 Result: 329939 distance between solvers: 120547
temperature: 10000 Result: 324369 distance between solvers: 126117

100000 iterations
Using rand seed: 1507494640 Number of vertexes is: 250
Greedy algorithm Result: 450406
temperature: 0.0001 Result: 377794 distance between solvers: 72612
temperature: 0.001 Result: 377794 distance between solvers: 72612
temperature: 0.01 Result: 377794 distance between solvers: 72612
temperature: 0.1 Result: 377794 distance between solvers: 72612
temperature: 1 Result: 377794 distance between solvers: 72612
temperature: 10 Result: 377794 distance between solvers: 72612
temperature: 100 Result: 372703 distance between solvers: 77703
temperature: 1000 Result: 330138 distance between solvers: 120268
temperature: 10000 Result: 326430 distance between solvers: 123976

Однако с возрастанием количества итераций алгоритм отжига действительно приближается к ответу полученному жадным алгоритмом. Но в любом случае жадный алгоритм по качеству результата далеко впереди от отжига с небольшим числом итераций. Если количество итераций устремлять в бесконечность, точность отжига будет лишь повышаться, ведь по формуле (1) ясно, что в случае её применения улучшающий сосед не был найден, значит $\partial < 0$, а значит и сама величина $\exp\left(\frac{\partial}{T_i}\right) \to 0$, то есть вероятность перехода к плохому соседу будет очень близка к нулю. Тесты выше подтверждают данное наблюдение, связанное с увеличение количества шагов и повышения точности отжига.

3. Вывод

Отжиг, работая в разы быстрее жадного алгоритма выдает приближенный к нему ответ. Однако он далеко не точный и отличается примерно на 20%. Повышая количество итераций для отжига, иными словами увеличивая время его работы мы получаем более точное решение (например, при 100 тысячах итераций – разница между результатами алгоритмов составила <15%) взамен жертвуя драгоценным временем. Это напоминает TIME-SPACE закон всех программ: жертвуя памятью мы экономим время, и наоборот, экономя память мы жертвуем временем. Так и с эвристическими алгоритмами – приходится выбирать между качеством решения и временем его получения. Однако это позволяет нам вручную регулировать его качество, основываясь на том времени, которое мы хотим потратить для его поиска. Такая настройка является более гибкой, что безусловно, помогает алгоритму варьироваться от задачи к задаче.