Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт (государственный университет)»

Факультет управления и прикладной математики
Кафедра информатики и вычислительной математики

Направление подготовки / специальность: 03.04.01 Прикладные математика и физика
Направленность подготовки: Математические и информационные технологии
Форма обучения: очная

Допущен к ГИА
Руководитель учебного подразделения
_____/_____
(подпись) *расшифровка подписи*
«___» _____2018г.

МОДЕЛИРОВАНИЕ СЕТЕЙ И ДИФФЕРЕНЦИАЦИЯ МЕТОДОВ РЕШЕНИЯ ЗАДАЧ ОПТИМАЛЬНОГО ПОСТРОЕНИЯ СЕТЕВЫХ СТРУКТУР

(магистерская диссертация)

Обучающийся: Лунев Сергей Вадимович
_____
(подпись обучающегося)
Научный руководитель: Кадощук Игорь Тарасович, кандидат физ.-мат. наук, доцент
_____
(подпись научного руководителя)

Москва 2018

Аннотация

Целью данной работы являлось предложение инструментария и методик для проведения сравнения и отбора эффективных алгоритмов решения задачи расширения сетевых структур с бюджетными ограничениями.

В работе были рассмотрены существующие смежные задачи и методы их решения. Также была разработана программная среда, способная моделировать различные сетевые структуры и позволяющая проводить сравнительный анализ алгоритмов, применимых к задаче расширения сети. В работе приводится пример результатов сравнения алгоритмов, полученных при помощи разработанного программного комплекса.

Оглавление

1. Введение **4**

2. Формулировка задачи **6**

3. Обзор схожих задач **10**

Node-Weighted Steiner Tree Problem 10

Prize-Collecting Steiner Tree Problem 10

The quota Steiner Tree Problem with Prizes 11

Budget Steiner Tree Problem with Prizes 11

4. Используемые методы **13**

4.1. Генерация случайных графов **13**

Erdős-Rényi or a binomial graph 13

Barabási-Albert preferential attachment model 13

Holme and Kim algorithm 14

Random kernel graph 14

Random geometric graph 14

4.2. Точное решение **16**

5. Результаты **18**

6. Заключение **23**

7. Литература **24**

ПРИЛОЖЕНИЕ 1 26

ПРИЛОЖЕНИЕ 2 28

1. Введение

Любой бизнес, столкнувшийся с проблемой расширения существующей сети, стремится максимизировать будущие доходы от планируемых инвестиций. Под сетевыми структурами могут предполагаться большие кабельные, коммуникационные, железно- и автодорожные сети, энергетические и трубопроводные сетевые структуры, сети филиалов и офисов, сети точек продаж и др., содержащие сотни или тысячи узлов и точек доступа (под ними могут пониматься, в том числе, глобальные телекоммуникационные операторы, государственные автомобильные и железные дороги, и т.д.).

Обычно, в реальных задачах для подобных сетей существует множество различных направлений расширения, что, в свою очередь, порождает значительный объем возможных решений задачи. Поэтому проблема разработки оптимального плана инвестирования в развитие сетевых структур является достаточно актуальной, ведь её эффективное решение даст заметный экономический эффект и существенные конкурентные преимущества. Однако довольно часто подобные задачи решаются бессистемно, а это негативно сказывается на результате.

Существует достаточно много методов решения подобных проблем, но сложно оценить их эффективность, так как нет корректной процедуры их сравнения при одинаковых входных данных и условиях работы. Поэтому задача выбора и оценки эффективности метода решения проблемы оптимального расширения сетевой структуры является достаточно актуальной.

Кажется очевидным, что сравнение результатов некоторого метода с точным решением, полученным, например, полным перебором является самым простым, самым эффективным решением проблемы оценки и выбора метода. Однако, не все так просто. Такой подход возможен только для небольших сетевых структур.

Серьезной проблемой является то, что на действительно больших сетях точное решение получить практически невозможно, а качество работ приближенных алгоритмов плохо прогнозируется. Поэтому в работе был реализован так же метод оценки качества решений приближенных алгоритмов на больших сетевых структурах.

В данной работе также рассматриваются существующие подходы к решению задачи оптимального построения сетевых структур, проводится их сравнительный анализ для некоторых алгоритмов на смоделированных сетевых структурах.

Кроме того, разработано решение для генерации, хранения, визуализации и модификации различных сетевых структур. А также реализован интерфейс, позволяющий разработчикам алгоритмов решения задачи планирования расширения сети анализировать и сравнивать их результаты на разных типах сетей.

2. Формулировка задачи

В работе будут рассматриваться графы, как модели сетевых структур. Вершины характеризуются их ценностью, т.е. выручкой, которую мы получим при присоединении к нашей сети (например, подключенные пользователи, новые торговые точки), а рёбра – расходами, необходимыми для их постройки (прокладка кабеля, дороги).

Помимо уже существующей сети будем рассматривать множество потенциальных узлов с известной ценностью и стоимостью их присоединения к уже существующей структуре.

Для упрощения задачи сведём исходную сеть в одну вершину графа. Перенесем в новый граф все вершины, не входящие в начальную структуру. Затем добавим новую вершину, которую назовем корневой. Теперь из каждой вершины, которая в исходном графе была соединена хотя бы с одной из вершин исходной структуры, построим ребро в корневую вершину. Вес такого ребра прием равным минимальному из весов рёбер, соединяющих данную вершину с одной из вершин исходной структуры.

После получения оптимального решения на таком графе мы можем вернуться к изначальному графу. Для этого нужно только заменить корневую вершину исходной сетью. Для каждой потенциальной вершины из рёбер, соединяющих её с начальной структурой, оставляем минимальное по весу ребро (которому соответствует ребро в корневую вершину). Поскольку присоединение потенциальной вершины к существующей сети по ребру с весом отличным от минимального повлечет увеличение стоимости, полученное решения будет оптимальным.

Часто выручка и затраты могут измеряться разными характеристиками, не всегда сравнимыми между собой.

В другой популярной формулировке задачи предлагается максимизировать разность между прибылью и расходами. Такая задача применима лишь в тех случаях, когда ценность вершин и затраты на рёбра сравнимы (например, измеряются только деньгами).

Так же, распространена формулировка задачи, в которой требуется максимизировать выручку при ограничении на расходы. То есть, присоединить набор вершин с наибольшей суммарной ценностью при фиксированном бюджете на постройку рёбер.

Сформулируем нашу задачу более формально

Пусть задан неориентированный взвешенный связный граф, для каждой вершины которого поставлено в соответствие некоторое число (ценность вершины) и каждому ребру поставлено в соответствие некоторое число (ценность ребра). Пусть задана начальная вершина. Пусть задано число, обозначающее ограничение на суммарную ценность ребер.

Необходимо найти в заданном графе подграф, включающий начальную вершину, суммарная ценность вершин которого будет максимальной при суммарном ценности рёбер, не превосходящем некоторое «бюджетное» ограничение.

Сформулируем задачу в терминах задачи целочисленного программирования:

Введём следующие обозначения:

G – исходный граф

$V(G)$ – множество его вершин

$E(G)$ – множество его ребер

v_i – прибыль от вершины с индексом i

$e_{i,j}$– стоимость ребра, связывающего вершины с индексами i и j

x_i – индикатор вхождения вершины в решение ($x_i=1$, если решение включает вершину i; $x_i=0$, *если нет*)

$y_{i,j}$ – индикатор вхождения ребра в решение ($y_{i,j}=1$, если решение включает ребро (i,j); $y_{i,j}=0$, *если нет*)

B – ограничение на бюджет

P – подграф

Задача: найти

$$Maximize \sum_{i \in V(G)} v_i x_i$$

при ограничениях:

$$\sum_{(i,j) \in E(G)} e_{(i,j)} y_{i,j} \leq B$$

$$2y_{(i,j)} \leq x_i + x_j \quad \forall\, i,j$$

$$\sum_{i \in P} x_i \geq \sum_{(i,j) \in P} y_{(i,j)} - 1$$

$$\sum_{i \in V(G)} x_i = \sum_{(i,j) \in E(G)} y_{(i,j)} - 1$$

$$x_0 = 1$$

$$x_i \in \{0; 1\}$$

$$y_{(i,j)} \in \{0; 1\}$$

Полученная задача является NP-трудной. Доказать это утверждение можно сведением задачи о рюкзаке к нашей задаче.

Примем наше ограничение на бюджет равным вместимости рюкзака. Затем, каждому предмету поставим в соответствие по одной вершине графа. Далее,

ценность каждой вершины примем равной ценности соответствующего предмета. Теперь, от каждой вершины нашего графа построим по одному ребру в начальный узел, а ценности таких рёбер положим равными весам соответствующих вершинам предметов. Тогда получим, что решение нашей задачи для полученного графа будет являться решением задачи о рюкзаке.

Таким образом, задача о рюкзаке сводится к нашей задаче по Карпу. Следовательно, наша задача принадлежит классу NP-трудных задач.

3. Обзор схожих задач

Сформулированная задача схожа с бюджетной задачей о построении дерева Штейнера с наградами (Budget Steiner Tree Problem with Prizes). А она в свою очередь является обобщением классической задачи Штейнера о минимальном дереве:

Пусть задан взвешенный неориентированный граф и подмножество его вершин, называемых обязательными. Требуется построить в данном взвешенном графе подграф минимального веса, включающий в себя все обязательные вершины вне зависимости от вхождения в него остальных вершин.

Node-Weighted Steiner Tree Problem

Идея задачи в оптимизации линейной комбинации выручки и расходов.[1] В частности, предлагается минимизировать сумму весов входящих в решение рёбер и не входящих в него вершин.

$$Minimize\ (\sum_{(i,j)\in E} c_{i,j}x_{i,j} \ + \ \sum_{i\in V} r_i(1-y_i))$$

Впервые предложил в 1987 году Segev.[2] Он разработал три эвристических метода решения для случая, когда обязательной является только одна вершина.

Prize-Collecting Steiner Tree Problem

В 1993 Bienstock и др. впервые предложили метод решения этой вариации задачи.[3] В основном они работали с задачей коммивояжёра с прибылью – вариация, в которой каждая вершина имела некоторую ценность, но не было обязательных для посещения вершин. Ими была разработана эвристика основанная на, предложенной Christofides в 1976 году эвристике для задачи коммивояжёра.[4]

Позже метод был адаптирован под Prize-Collecting Steiner Tree Problem. В 1995 году Goemans и Williamson улучшили производительность в худшем случае алгоритмом с аппроксимацией *(2−1/(|V|−1))*.[5] Они работали с корневой вариацией задачи (когда одна вершина считалась корнем и обязательно должна была входить в решение, как впервые предлагал Segev). Их методика является адаптацией более общего метода, который может применяться к широкому спектру оптимизационных задач на графах.

The quota Steiner Tree Problem with Prizes

В вариации задачи с квотой целью является минимизировать суммарную ценность рёбер при гарантированном минимуме суммарной ценности включенных в решение вершин.

$$Minimize \sum_{(i,j)\in E} c_{i,j} x_{i,j}$$

при $\sum_{i\in V} r_i y_i \geq Q,$

где Q – квота, то есть минимально допустимая суммарная ценность вершин, взятых в решение.

Данная вариация задачи Штейнера с призами очень слабо распространена и почти не встречается в литературе. Minkoff и Phillips в своей работе смогли получить алгоритм с 2.5-аппроксимацией.[6]

Budget Steiner Tree Problem with Prizes

Эта версия задачи является копией версии с квотой, но с обратными целями. Здесь нам нужно максимизировать ценность вершин при ограничении бюджета на постройку рёбер.

$$Maximize \sum_{i \in V} r_i y_i$$

при $\sum_{(i,j) \in E} c_{i,j} x_{i,j} \leq B,$

где B – ограничение на бюджет.

Данному варианту задачи исследований почти не посвящено. Однако Johnson, Minkoff и Phillips в своей работе доказали, что при многократном повторном применении алгоритма с 3-аппроксимацией к задаче с квотой можно получить алгоритм с *(5+ε)*-аппроксимацией, где *ε≥0,* для бюджетной версии задачи без корневой вершины.[6] Для более эффективной работы алгоритма ими было предложено использовать ту же стратегию, что и в задаче Штейнера с квотой, а именно, умножить стоимости на фактор *α,* (в этом случае для попытки найти решение, удовлетворяющее ограничениям) и применить бинарный поиск по значениям *α.*

4. Используемые методы

4.1. Генерация случайных графов

Важной частью моделирования сетевой структуры является метод генерации случайного графа. Поскольку в реальном мире встречается большое количество различных по структуре сетей, и поскольку результаты работы приближённых алгоритмов могут сильно зависеть от структуры графа, то для объективной оценки алгоритмов требуется применять различные способы генерации тестовых графов, для получения оценки наиболее подходящей для каждой конкретной задачи. Рассмотрим несколько известных подходов к генерации случайных графов.

Erdős-Rényi or a binomial graph

Зададим N точек и пронумеруем их от 1 до N. Получим $N(N-1)/2$ потенциальных рёбер, которыми можно соединить наши точки и получить граф. Для каждой пары вершин делаем независимый от других выбор, строить ребро или нет. Пусть вероятность построить ребро будет равна p. Всего имеем $2^{N(N-1)/2}$ возможных графов. Такой алгоритм занимает $O(n^2)$ времени. [7]

Barabási-Albert preferential attachment model

В данном алгоритме учитывается, что в реальной жизни сетевые структуры разрастаются постепенно, добавляя новые вершины со временем, а не готовой сетью из N вершин. Начиная с небольшого числа рёбер m_0, мы на каждом шаге добавляем новую вершину с $m(\leq m_0)$ рёбрами, соединяющими её с вершинами, уже находящимися в системе. Положим, что вероятность P соединения новой вершины с вершиной i зависит от связности этой

вершины. То есть $P(k_i) = k_i / \sum_j k_j$. Таким образом, спустя t шагов мы получим граф с (m_0+t) вершинами и mt рёбрами. [8]

Holme and Kim algorithm

С ростом сети, уменьшается её транзитность (отношение числа треугольников к числу всех возможных треугольников в графе). По сути данный алгоритм, является моделью роста Barabási-Albert, но с дополнительным шагом. После случайного добавления ребра с некоторым шансом добавляется связь с одним из соседей, т.е. создание треугольника. Этот алгоритм улучшает модель Barabási-Albert, в том смысле, что при желании позволяет достичь более высокой средней кластеризации. [9]

Random kernel graph

В этом алгоритме ядро определяется через его определенный интеграл, который должен быть представлен как один из аргументов. Если интеграл и корень интеграла ядра можно найти за время $O(1)$, то сам алгоритм выполняется за время $O(n+m)$, где m – ожидаемое число рёбер графа. [10]

Random geometric graph

Результатом этого алгоритма является случайный геометрический граф в единичном кубе определённой размерности. Для этого в единичный куб однородно случайно помещаются n вершин. Две вершины соединяются ребром в том случае, если расстояние между ними не превышает заранее определённого радиуса r. Если определить рёбра через K-мерное дерево, то сложность алгоритма значительно снизится – с $O(n^2)$ до $O(n)$ [11]

Помимо задачи генерации случайных графовых структур, которую в достаточной степени решают перечисленные выше методы, необходимо

вершинам и рёбрам присвоить значения их ценностей, то есть их выручки и стоимости, соответственно. Для этого в данной работе предлагается смоделировать различные вероятностные распределения, в зависимости от специфики конкретной моделируемой структуры. При тестировании алгоритмов в рамках данной работы использовалось распределение Гаусса, как наиболее часто встречающееся в реальных задачах.

4.2. Точное решение

Для анализа и сравнения различных алгоритмов решения нашей задачи необходимо как-то оценить их результаты. Точное решение можно найти, например, полным перебором всех решений задачи. В данной работе применялся алгоритм с полным перебором всех возможных подмножеств вершин с последующим построением для них минимального остовного дерева.

Запишем более подробное описание алгоритма:

- Если суммарный вес вершин подмножества больше текущего максимума суммарной выручки:
    - Если подмножество вершин образует связный подграф:
        - В полученном подграфе находим минимальное остовное дерево
        - Если вес полученного дерева меньше бюджетного ограничения:
            - Обновляем текущий максимум суммарной выручки
            - Дерево запоминаем как текущее решение

Однако данный алгоритм имеет временную сложность $O(2^n \cdot n \cdot log(n))$, что делает его применимым только для сравнения алгоритмов на довольно небольших графах.

Для сетевых структур с большим числом узлов в работе предлагается использовать иной подход к оценке алгоритмов. Воспользуемся тем, что можем менять ограничения. Предлагается решить другую задачу, для которой существует аппроксимация с коэффициентом 2.

Запустим на графе алгоритм, предложенный Goemans и Williamson, дающий результат не более чем в 2 раза хуже оптимального для задачи. [12] Соответственно, приняв ограничение на бюджет равным весу полученного дерева, мы можем считать это дерево близким к оптимальному решению бюджетной задачи. Это, в свою очередь, позволит нам получить тестовую задачу, с большим количеством вершин, для которой мы будем знать хорошую оценку оптимального решения.

5. Результаты

В данной работе были протестированы следующие алгоритмы:

Наивный жадный алгоритм

На каждом шаге данный алгоритм подключает лучшую из достижимых вершин. После чего обновляет список достижимых вершин и пересчитывать стоимость их подключения.

Жадный поиск по кратчайшим путям

Данный алгоритм на каждом шаге выбирает лучший, согласно функции полезности, из кратчайших путей из каждой вакантной вершины графа в одну из множества уже выбранных вершин.

Grasp алгоритм

На каждом шаге строится множество допустимых решений, основываясь на одном из описанных жадных алгоритмов. Далее для найденных решений находится минимум и максимум функции полезности.

Затем из полученного множества отбираются решения, отличающиеся от лучшего не более чем на $\alpha \cdot (max\text{-}min)$, где α – некоторый заранее заданный параметр. Данная процедура прогоняется несколько раз, после чего выбирается лучшее из решений.

В данной работе было разработано решение для генерации, хранения визуализации и модификации различных сетевых структур, позволяющее также анализировать и сравнивать результаты различных алгоритмов на разных типах сетей. Для тестирования полученной программы был проведён сравнительный анализ четырёх алгоритмов, применённых к задаче оптимального расширения сетевой структуры: две вариации жадного алгоритма (поиск по кратчайшим путям и наивный) и две аналогичные вариации Grasp-алгоритма Результаты сравнения тестируемых алгоритмов на малых графах в ***приложении 1***. Результаты оценки

работы тестируемых алгоритмов на больших графах с 100 вершинами в ***приложении 2***.

Для оценки оптимальности метода, предложенного для нахождения приближённого решения на больших графах, сравним его с точным решением на малых графах, переопределив ограничение на бюджет. Для данного теста генерировалось 100 графов с 16 вершинами, Гауссовым распределением весов вершин и рёбер (с мат.ожиданием 20 и дисперсией 5), коэффициентом плотности графа 0.6. Запишем в таблицу, на каком количестве тестовых графов решение оказалось равно точному, хуже точного не более, чем на 2% и хуже точного не более, чем на 5%.

Таблица 1 – Оценка эффективности gw-алгоритма

	gw-algorythm
=100%	92
≥98%	100
≥95%	100

Отсюда видим, что в 92 случаях из 100 решение алгоритма совпало с точным, а в оставшихся 8 оказалось не более, чем на 2% хуже точного решения. Таким образом, используемый алгоритм даёт достаточно хорошую оценку решения задачи расширения сети

Программное решение предусматривает возможность визуализации полученных результатов для более наглядного сравнения. Ниже приведены примеры точного решения и результатов работы четырёх тестируемых алгоритмов. Красным и черным цветами обозначены соответственно вершины и рёбра, не вошедшие в результирующий подграф. Зелёные вершины синие рёбра составляют решение задачи.

Параметры тестового графа: 16 вершин, распределение весов вершин и рёбер по Гауссу с мат.ожиданием 20 и дисперсией 5, коэффициент плотности графа 0.6, ограничением на бюджет 120

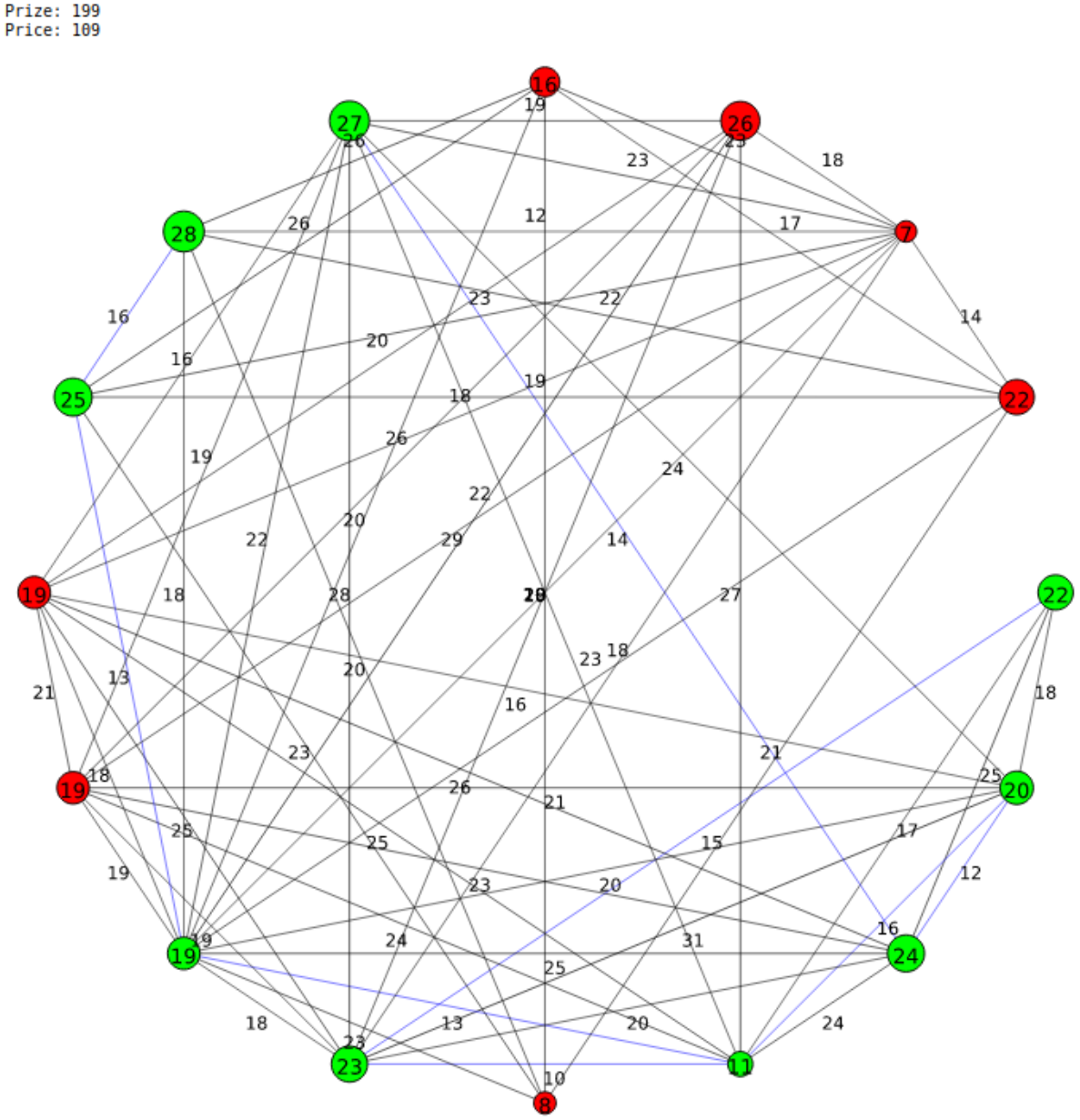

Рисунок 1 – Точное решение (полный перебор)

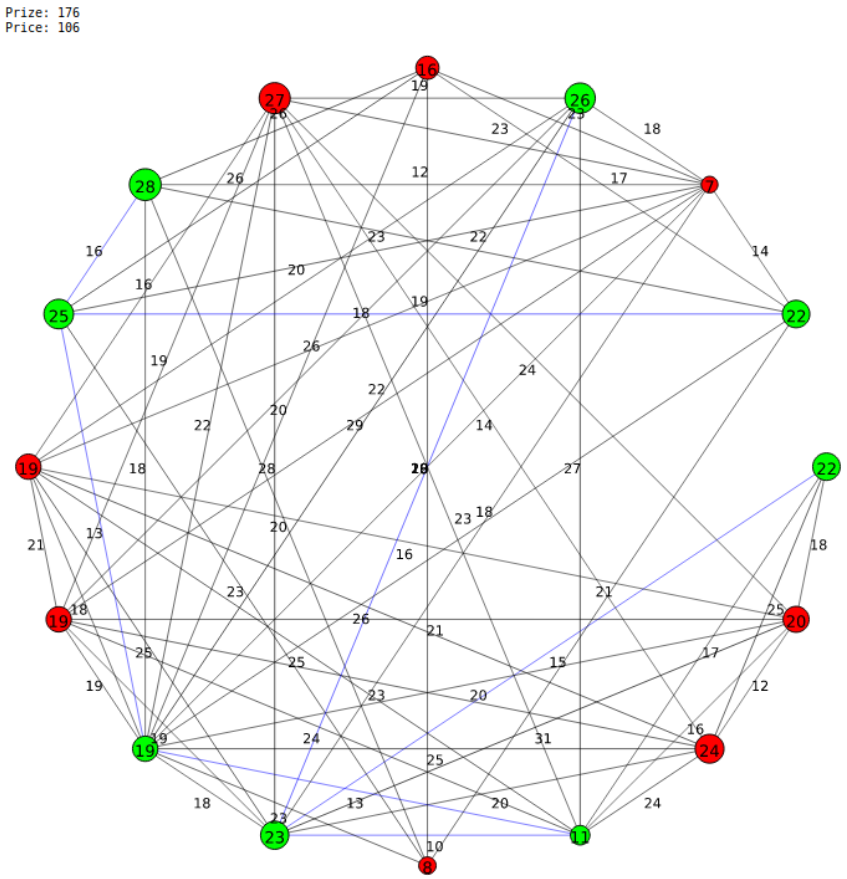

Рисунок 2 – Жадный поиск по кратчайшим путям

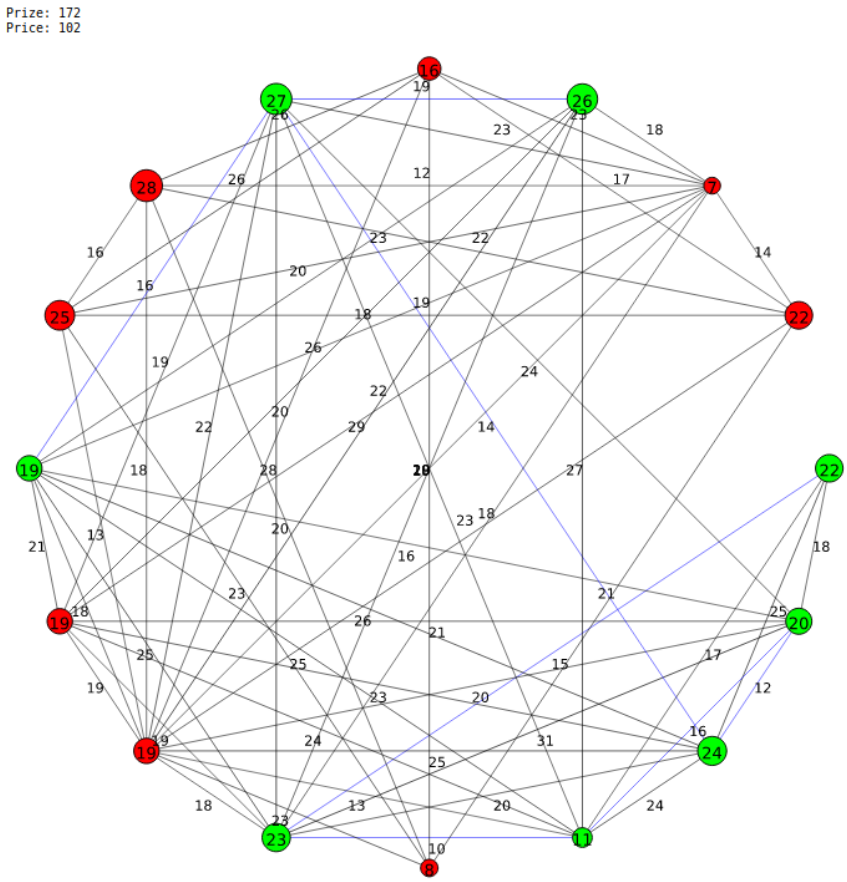

Рисунок 3 – Наивный жадный алгоритм

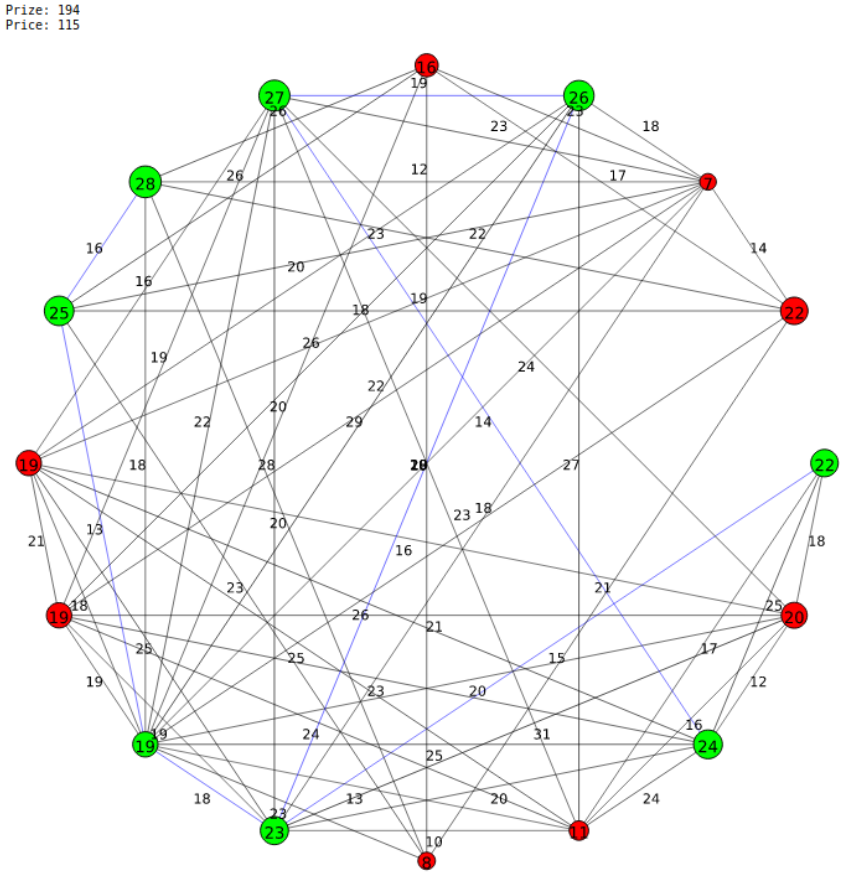

Рисунок 4 – Grasp-алгоритм с кратчайшими путями

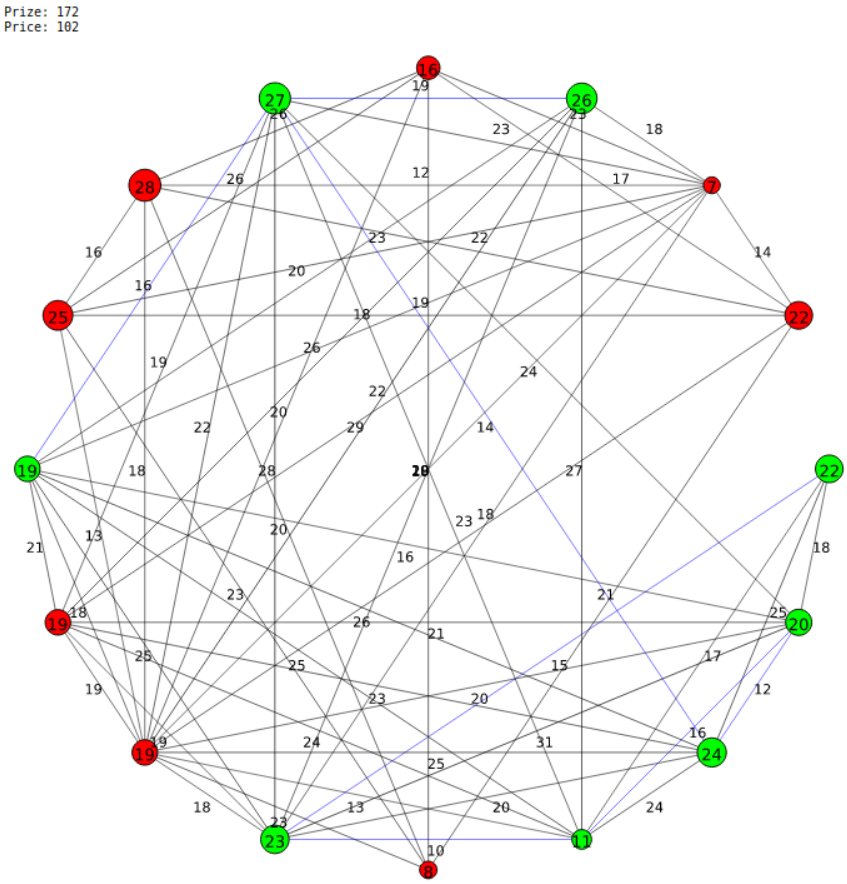

Рисунок 5 – Наивный Grasp-алгоритм

6. Заключение

В работе рассмотрена задача об оценки качества различных методов оптимального расширении сети с ограничением на бюджет. Проведен также анализ и обзор смежных задач и подходов к их решению.

Кроме того была разработана программная среда позволяющая моделировать различные типы графовых структур и, соответственно, получать и проводить сравнительный анализ результатов работы разных алгоритмов на различных входных данных, моделирующих различные реальные ситуации в рамках задачи об оптимальном расширении сетевой структуры с бюджетным ограничением.

В дальнейшем планируется довести работу до конечного программного продукта.

7. Литература

1. *Alysson M. Costa, Jean-François Cordeau and Gilbert Laporte,* Steiner tree problems with profits *INFOR*, 44, 99–115, 2006
2. *Segev, A.* The node-weighted Steiner tree problem, Networks 17: 1–17, 1987
3. *Bienstock, D., Goemans, M. X., Simchi-Levi, D. and Williamson, D.,* A note on the prize collecting traveling salesman problem, Mathematical Programming 59: 413–420, 1993
4. *Christofides, N.* Worst-case analysis of a new heuristic for the travelling salesman problem, Technical Report 388 - Graduate School of Industrial Administration, Carnegie Mellon University, 1976
5. *Goemans, M. X. and Williamson, D. P.* A general approximation technique for constrained forest problems, SIAM Journal on Computing 24: 296–317, 1995
6. *Johnson, D. S., Minkoff, M. and Phillips, S.,* The prize collecting Steiner tree problem: theory and practice, Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms, San Francisco : 760–769, 2000
7. *E.N. Gilbert*, Random Graphs, Ann. Math. Stat., 30, 1141 (1959)
8. *A. L. Barabási and R. Albert,* “Emergence of scaling in random networks”, Science 286, pp 509-512, 1999.
9. *P. Holme and B. J. Kim*, “Growing scale-free networks with tunable clustering”, Phys. Rev. E, 65, 026107, 2002.
10. *Hagberg A, Lemons* N, “Fast Generation of Sparse Random Kernel Graphs”. PLoS ONE 10(9): e0135177, 2015. doi:10.1371/journal.pone.0135177, 2015
11. *Penrose, Mathew, Random Geometric Graphs*, Oxford Studies in Probability, 5, 2003

12. *Piotr Indyk, Ludwig Schmidt*, A Nearly-Linear Time Framework for Graph-Structured Sparsity Chinmay Hegde, ICML 2015

ПРИЛОЖЕНИЕ 1

Результаты сравнения тестируемых алгоритмов на 100 графах с N вершинами, коэффициентом плотности графа k, ограничением на бюджет B и распределением весов вершин и рёбер по Гауссу с мат.ожиданием 20 и дисперсией 5

Таблица 2 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,4, B=90

	greedy spath	greedy simple	grasp spath	grasp simple
100%	14	17	24	17
≥98%	28	37	40	34
≥95%	41	59	63	50

Таблица 3 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,4, B=120

	greedy spath	greedy simple	grasp spath	grasp simple
100%	9	21	24	16
≥98%	23	32	40	25
≥95%	44	52	79	53

Таблица 4 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,4, B=150

	greedy spath	greedy simple	grasp spath	grasp simple
100%	11	11	27	8
≥98%	25	33	57	29
≥95%	57	55	82	53

Таблица 5 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,4, B=180

	greedy spath	greedy simple	grasp spath	grasp simple
100%	14	16	30	15
≥98%	36	42	64	41
≥95%	62	66	89	70

Таблица 6 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,6, B=90

	greedy spath	greedy simple	grasp spath	grasp simple
100%	6	14	15	11
≥98%	15	27	36	25
≥95%	37	53	59	49

Таблица 7 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,6, B=120

	greedy spath	greedy simple	grasp spath	grasp simple
100%	10	10	18	14
≥98%	17	28	49	30
≥95%	38	50	74	56

Таблица 8 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,6, B=150

	greedy spath	greedy simple	grasp spath	grasp simple
100%	16	10	28	11
≥98%	34	38	61	34
≥95%	68	62	82	66

Таблица 9 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,6, B=180

	greedy spath	greedy simple	grasp spath	grasp simple
100%	32	22	51	21
≥98%	64	49	80	39
≥95%	86	79	95	78

Таблица 10 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,8, B=120

	greedy spath	greedy simple	grasp spath	grasp simple
100%	8	6	15	5
≥98%	16	25	48	26
≥95%	43	54	75	51

Таблица 11 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,8, B=150

	greedy spath	greedy simple	grasp spath	grasp simple
100%	5	12	31	10
≥98%	35	35	66	31
≥95%	73	66	91	62

Таблица 12 – сравнение тестовых алгоритмов на малом графе с параметрами N=16, k=0,8, B=180

	greedy spath	greedy simple	grasp spath	grasp simple
100%	47	33	58	31
≥98%	77	55	91	56
≥95%	93	84	99	83

ПРИЛОЖЕНИЕ 2

Результаты сравнения тестируемых алгоритмов на 100 графах с 100 вершинами, коэффициентом плотности графа k и распределением весов вершин и рёбер по Гауссу с мат.ожиданием 20 и дисперсией 5

Таблица 13 – сравнение тестовых алгоритмов на большом графе с параметром k=0,4

	greedy spath	greedy simple	grasp spath	grasp simple
100%	15	0	23	0
≥98%	100	46	100	40
≥95%	100	100	100	100

Таблица 14 – сравнение тестовых алгоритмов на большом графе с параметром k=0,6

	greedy spath	greedy simple	grasp spath	grasp simple
100%	24	0	31	0
≥98%	100	33	100	38
≥95%	100	100	100	100

Таблица 15 – сравнение тестовых алгоритмов на большом графе с параметром k=0,8

	greedy spath	greedy simple	grasp spath	grasp simple
100%	35	0	40	0
≥98%	99	54	100	47
≥95%	100	100	100	100