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.

РАЗРАБОТКА И ИССЛЕДОВАНИЕ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ОПТИМАЛЬНОГО РАСШИРЕНИЯ СЕТИ С ОГРАНИЧЕНИЯМИ

(магистерская диссертация)

Обучающийся: Мареичев Петр Алексеевич
_____
(подпись обучающегося)
Научный руководитель: Кадощук Игорь Тарасович, кандидат физ.-мат. наук, доцент
_____
(подпись научного руководителя)

Москва 2018

Аннотация

Целью данной работы являлось изучение методик и разработка алгоритма для решения задачи оптимального расширения сетевой структуры при наличии бюджетных ограничений.

В ходе работы было реализованно и протестированно четыре алгоритма, находящие близкое к оптимальному решения задачи за разумное время. В работе приводится описание данных алгоритмов и результаты их тестирования

Оглавление

Аннотация 1
Оглавление 2
Введение 4
Постановка задачи 5
Формальное описание задачи 6
Существующие подходы и методы решения 9
Способы решения 10
Полный перебор 10
Приближенные методы решения 11
Наивный жадный алгоритм 12
Описание алгоритма 13
Оценка временной сложности алгоритма 14
Доказательство отсутствия гарантированной точности 15
Жадный поиск по кратчайшим путям 16
Описание алгоритма. 16
Оценка временной сложности алгоритма 17
Преимущества перед жадным 17
Доказательство отсутствия гарантированной точности 18
Пример графа, где алгоритм жадного поиска по кратчайшим путям проигрывает наивному 18
Grasp алгоритмы 19

Тестирование алгоритмов. 20
Наивный жадный алгоритм 21
Жадный поиск по кратчайшим путям 21
Grasp на основе наивного жадного алгоритма 23
Grasp на основе жадного поиска по кратчайшим путям 23
Заключение 25
Список литературы 26

Введение

Практически перед любой организацией, чья деятельность так или иначе связана с большими сетевыми структурами, регулярно встает вопрос о расширении существующей сети. Примерами таких структур могут служить кабельные, коммуникационные, железно- и автодорожные сети, энергетические сетевые структуры, сети офисов и представительств, сети точке продаж и прочее.

Для подобных структур обычно существует большое количество возможных направлений расширения, различные комбинации которых, как следствие, порождают колоссальный объем возможных решений. Ввиду этого, задача выбора оптимального с финансовой и технической точек зрения решения встает очень остро. К сожалению, на данный момент не разработан общий подход к решению подобного рода задач. Часто подобные задачи решаются вручную, бессистемным перебором решений и выбором лучшего из угаданных. Найденные таким образом решения часто бывают далеки от оптимальных.

В данной работе предложены алгоритмы, позволяющие эффективно искать близкие к оптимальным решения задачи планирования развития такой сетевой инфраструктуры, когда, при наличии ограниченных инвестиций, организация стремиться получить как можно большую прибыль. Эта задача является очень востребованной, поскольку в условиях конкуренции компания, наиболее эффективно использующая инвестиции получает существенное конкурентное преимущество.

Постановка задачи

В данной работе будет решаться задача в следующей формулировке: при заданном ограничении на общую стоимость требуется расширить сеть таким образом, чтобы максимизировать общий доход.

Далее мы будем моделировать сетевую структуру графом.

Будем считать, что доход компании зависит от включенных в сеть узлов, которым соответствуют вершины графа (торговые точки, подключенный пользователи, достигнутые города), в то время как основные расходы приходятся на соединение, то есть на ребра графа (прокладка коммуникаций, дорог и тд). Другими словами, все расходы вершин распределяем по ребрам, связанным с этими вершинами.

В случае, если подключение узла несет в сети дополнительные затраты, при построении модели можем добавить их к стоимости соединения. Поскольку при оптимальном решении задачи, соединения прокладываются только к еще не подключенным точкам, такое допущение не меняет оптимального решения.

Будем считать, что помимо существующей сети нам будет дано множество потенциальных узлов и стоимость соединения между узлами, где это соединение возможно.

Преобразуем данную задачу, сведя уже существующие узлы к одной вершине. Перенесем все неподключенные вершины и ребра между ними в новый граф. Построим в новом графе дополнительную вершину, которую далее будем называть корневой. Далее из каждой потенциальной вершины, для которой в исходной графе существует ребро, соединяющее ее с уже подключенной, построим ребро к корневой, вес которого примем равным минимальному из весов ребер, соединяющих данную потенциальную точку с уже подключенной точкой в изначальном графе.

После нахождения оптимального решения в новом графе, мы можем легко вернуться к начальному графу, присоединив в исходном графе все выбранные потенциальные, из которых есть ребро к корневой вершине, кратчайшим из ребер, ведущим к изначально подключенным вершинам. Поскольку присоединение потенциальной вершины к существующей сети по ребру с весом отличным от минимального повлечет увеличение стоимости, полученное решения будет оптимальным.

Формальное описание задачи

Перейдем к более формальному описанию данной задачи.

Пусть задан взвешенный неориентированный связный граф. Пусть каждой вершине графа сопоставлено в соответствие некоторое число обозначающее «доход» и каждому ребру поставлено в соответствие число, обозначающее «расход», «стоимость». Пусть задана корневая вершина графа. Пусть задано число, ограничение на сумму всех «расходов», обозначающее «бюджет».

Задача -- найти в данном графе подграф, включающий корневую вершину, сумма «доходов» от вершин которого будет максимальной, при том что суммарные «расходы» ребер в нем ограничены «бюджетом».

Сформулируем задачу в терминах задачи целочисленного программирования

Пусть задан граф G.

$V(G)$ - множество вершин графа

$E(G)$- множество его ребер.

пусть c_i- доход от вершины i

пусть $w_{i,j}$- стоимость ребра i,j

пусть v_i-индикатор вхождения вершины

пусть $e_{i,j}$- индикатор вхождения ребра

пусть W- бюджетное ограничение

Тогда нужно максимизировать функцию:

$$\sum_{i \in V(G)} \quad v_i c_{ii} \to max \qquad (1)$$

При следующих ограничениях:

$$\sum_{i,j \in V(G)} \quad \frac{1}{2} e_{i,j} w_{i,j} \leq W \qquad (2)$$

$$2e_{i,j} \leq v_i + v_j \ \ \forall i,j \in V(G) \qquad (3)$$

$$\sum_{i \in P} \quad v_i \geq \sum_{i,j \in P} \quad \frac{1}{2} e_{i,j} - 1 \quad \forall P \subseteq V(G) \qquad (4)$$

$$\sum_{i \in V(G)} \quad v_i = \sum_{i,j \in V(G)} \quad \frac{1}{2} e_{i,j} - 1 \qquad (5)$$

$$e_{i,i} = 0 \ \ \forall i \in V(G) \qquad (6)$$

$$e_{i,j} = e_{j,i} \ \ \forall i,j \in V(G) \qquad (7)$$

$$v_0 = 1 \qquad (8)$$

$$e_{i,j} \in \{0;1\} \ \forall i,j \in V(G) \qquad (9)$$

$$v_i \in \{0;1\} \ \ \forall i \in V(G) \qquad (10)$$

Уравнение (2) обозначает ограничение на суммарный вес вершин

Уравнение (4) гарантирует отсутствие циклов. Здесь P – любое подмножество множества вершин исходного графа

Уравнение (5) задает соотношение между числом ребер и вершин в графе, что вместе с уравнением (4) гарантирует связность графа.

Данная задача принадлежит классу NP-трудных задач.

Докажем это утверждение, сведя к ней к задаче о рюкзаке:

Положим бюджетное ограничение равным вместимости рюкзака.

Каждому предмету сопоставим вершину графа. Положим доход от вершины равной ценности предмета.

Построим от вершины единственное ребро к корневой вершине. Вес ребра положим равным весу предмета.

Соответственно, решение задачи для построенного графа будет означать нахождение решения для задачи о рюкзаке, следовательно, задача о рюкзаке сводится к нашей по Карпу, а значит, наша задача является NP-трудной.

Существующие подходы и методы решения

Полученная задача известна в литературе как версия проблемы построения дерева Штейнера с призами с бюджетным ограничением (Budget Steiner Tree Problem with Prizes). Данная задача впервые описана в работе *Segev, A.* в 1987 году[1] и является обобщением известной задачи Штейнера о минимальном дереве:

Пусть задан взвешенный неориентированный граф. Пусть задано подмножество вершин графа, называемое терминалами. Построить в данном взвешенном графе подграф минимального веса, включающий в себя все терминалы и, возможно некоторые другие вершины графа[2]

Данная вариация задачи довольно редко встречается в литературе. На сегодняшний день известна только $(5 + \varepsilon)$-аппроксимация предложенная Johnson, Minkoff и Phillips в 2000 году[3]. Суть алгоритма состоит в многократном применении алгоритма для поиска оптимального решения другой вариации задачи на построение оптимального дерева Штейнера, когда при ограничении на минимальный суммарный доход требуется найти минимальное по стоимости дерево. Для данной задачи, известной как версия проблемы построения дерева Штейнера с призами с ограничением в виде квоты(Quota Steiner Tree Problem with Prizes), существует $(2 + \varepsilon)$аппроксимация <ссылка>. Авторы показали, что подбирая квоту методом бинарного поиска, и прогоняя упомянутый выше алгоритм, пока вес полученного дерева не станет соответствовать бюджетным ограничениям исходной задачи, можно получит аппроксимацию с гарантированной точностью $(5 + \varepsilon)$.

В результате авторами был предложен алгоритм дающий решение гарантированно не более чем в пять раз хуже оптимального, что практически не применимо для решения реальных задач.

Способы решения

Поскольку, как было показано выше, данная задача принадлежит к классу NP-трудных задач, не существует (в предположении, что$P \neq NP$) эффективного метода ее решения.

Полный перебор

Для нахождения точного решения данной задачи необходимо применять полный перебор всех возможных решений. В рамках данной работы был разработан и реализован алгоритм для нахождения точного решения данной задачи, использующий перебор всех возможных подмножеств вершин с последующим построения минимального остовного дерева на подграфе, порожденным данным множеством.

Ниже приведено подробное описание алгоритма

Для каждого подмножества вершин:

- Если суммарный вес вершин в множестве больше текущего максимума:
    - Если данное множество вершин образует связный подграф:
        - На данном подграфе находим минимальное остовное дерево используя алгоритм Краскала[4] Если вес построенного дерева меньше ограничения:
            - обновляем текущий максимум
            - запоминаем полученное дерево.

Данный алгоритм имеет временную сложность $O(2^n n \log(n))$, что делает его неприменимым на практике для сетей с большим количеством вершин. Однако, данный алгоритм может быть применен для нахождения точного решения на

небольших графах, которые потом могут использоваться для тестирования и оценки точности приближенных алгоритмов.

Приближенные методы решения

В данной работе будут рассматриваться различные приближенные алгоритмы, принадлежащие к классу «жадных», то есть на каждом шаге выбирающие локально-оптимальное решение[5]. Как будет показано ниже, данные алгоритмы не гарантируют близость полученного решения задачи к оптимальному с никакой наперед заданной точностью, однако на практике позволяющие найти хорошее приближенное решение за разумное время. Также будет использован рандомизированный подход к поиску оптимальных решений.

Наивный жадный алгоритм

Наиболее очевидным алгоритмом для решения данной задачи является простой жадный алгоритм. На каждом шаге будем подключать лучшую из достижимых вершин. После чего обновлять список достижимых вершин и пересчитывать стоимость их подключения. Данный алгоритм можно попытаться улучшить, используя различные функции полезности для достижимых вершин и, например, варьируя их в зависимости от шага. Также, можно улучшить результат, перестраивая на множестве подключенных вершин минимальное дерево, тем самым, уменьшая суммарную стоимость уже подключенных вершин.

Описание алгоритма

- Заносим в множество достижимых вершин все вершины соседние корневой, таких что вес ребра между ними и корневой не превышает бюджета.
- Для каждой вершины находим стоимость ее присоединения (для недостижимых на первом шаге вершин полагаем стоимость равной бесконечности)
- Пока множество достижимых вершин не пусто:
    - Выбираем из множества достижимых вершин лучшую, согласно функции полезности.
    - Присоединяем ее к множеству подключенных вершин и удаляем из множества достижимых.
    - Уменьшаем остаток бюджета на величину равную стоимости подключения выбранной вершины.
    - Для каждой вершины, смежной выбранной:
        - Если ее нет в списке доступных -- добавляем.
        - Записываем стоимость ее присоединения, как минимум из цены ребра и старой стоимости.
    - Удаляем из списка достижимых все вершины, стоимость присоединения которых превышает остаток бюджета
- По завершении цикла возвращаем минимальное дерево, построенное на выбранных вершинах.

Оценка временной сложности алгоритма

Оценим временную сложность описанного выше алгоритма:

на каждом шаге мы выбираем лучшую вершину из N. сложность O(n)

далее обновляем список доступных, тоже за O(n)

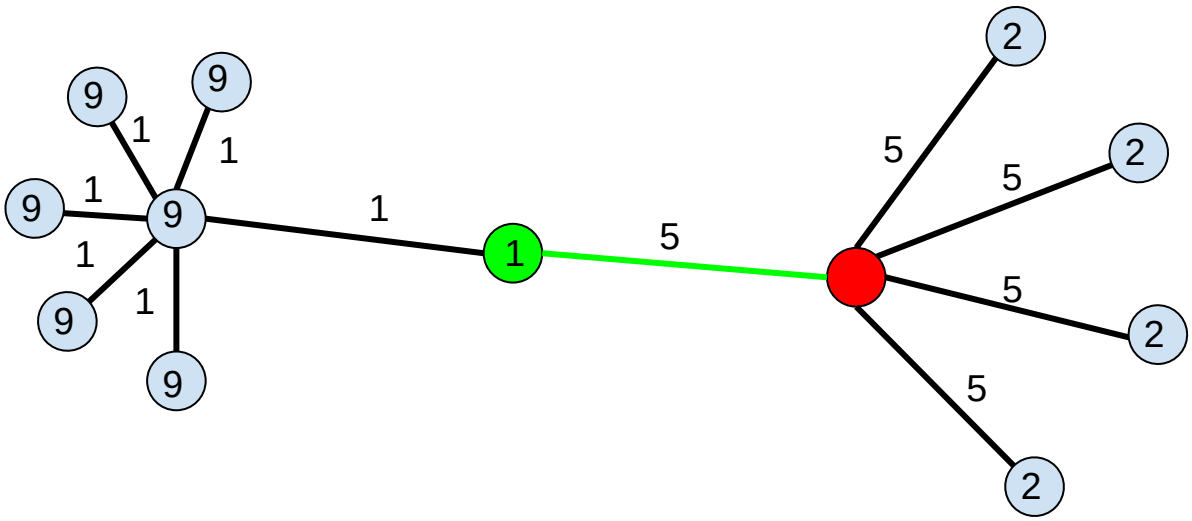
действия повторяются для каждой вершины, соответственно временная сложность оценивается как $O(n^2)$

Проводится построение минимального остовного дерева на выбранных вершинах методом Краскала, сложность которого не превышает $O(n^2)$. Соответственно, получаем оценку $O(n^2)$ для всего алгоритма.

Доказательство отсутствия гарантированной точности

Покажем, что данный алгоритм не гарантирует никакой точности.

Рассмотрим граф изображенный ниже. Очевидно, что оптимальным решением будет включить в сеть точки из левой части графа, однако жадный алгоритм не рассматривает вершины лежащие слева, пока не будет построено ребро, выделенное на рисунке зеленым цветом. Веса в графе подобраны таким образом, что данное ребро считается наихудшим, имея очень большой вес при малом доходе от вершины и не будет взято пока не будут присоединены все прочие доступные вершины. Увеличивая доход от вершин слева, мы можем получить граф, на котором жадный алгоритм будет давать решение сколь угодно далекое от оптимального.



Жадный поиск по кратчайшим путям

В рамках работы был разработан жадный алгоритм, выбирающий оптимальный из множества кратчайших путей из каждой вакантной вершины графа в уже построенный граф.

Описание алгоритма.

- Ищем кратчайшие пути между каждой пары вершин, используя алгоритм Флойда-Уоршелла[6] и сохраняем.
    - Далее в цикле:
        - Для каждой еще не подключенной вершины:
            - Перебираем пути к уже подключенным вершинам и находим минимальный.
            - Если его стоимость не превышает остаток бюджета, заносим его в список кандидатов.
        - Из списка кандидатов выбираем лучший из подходящих под ограничения, согласно функции полезности.
        - Все вершины выбранного пути заносим в список подключенных.
        - Уменьшаем остаток бюджета на величину равную стоимости подключения выбранной вершины.
        - Продолжаем, пока можем присоединить хоть одну вершину.

Оценка временной сложности алгоритма

Оценим временную сложность описанного выше алгоритма

Первым шагом алгоритм применяет алгоритм Флойда-Уоршелла, имеющий сложность $O(n^3)$.

Затем на каждом шаге

Для каждой вершины:

Перебираем путь во все подключенные: $O(n^3)$

Остальные шаги алгоритма не добавляют сложность.

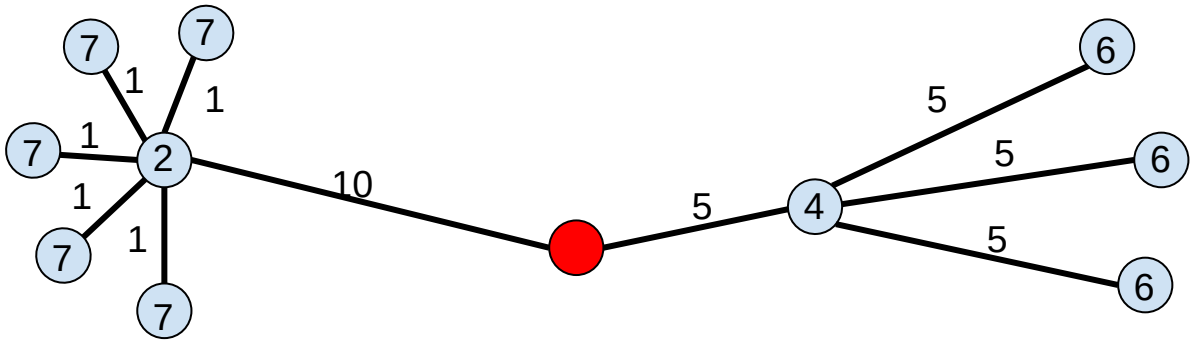
В результате получаем итоговую сложность алгоритма равную $O(n^3)$

Преимущества перед жадным

Данный алгоритм имеет большую временную сложность относительно простого жадного, однако, в большинстве случаев он находит гораздо лучшее решение, за счет того, что на любом шаге он рассматривает все вершины в качестве доступных для подключения, а не только смежные с уже подключенными.

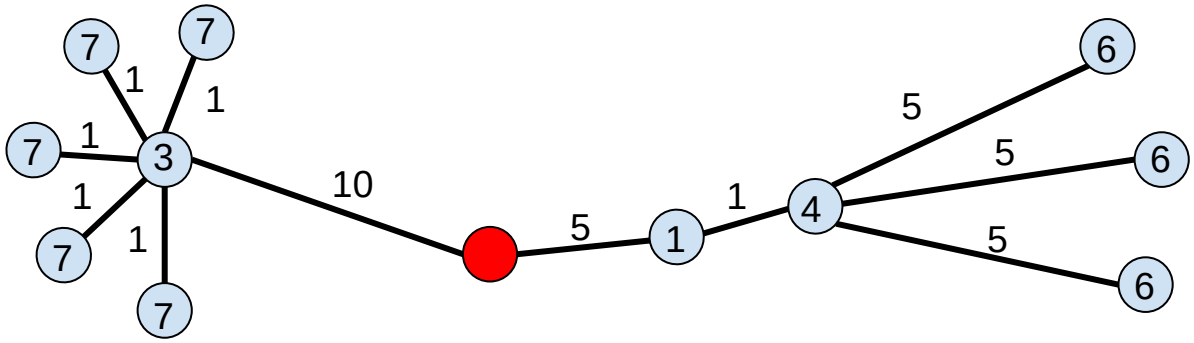
Доказательство отсутствия гарантированной точности

Рассмотрим граф изображенный ниже. Описанный выше алгоритм на первом шаге будет считать оптимальным путь в одну из вершин правой части графа, как имеющий большую доходность вершин при меньшей суммарной стоимости. После чего он будет подключать вершины из правой части, поскольку путь в них также будет считаться оптимальнее пути в любую из вершин левой части. Однако, оптимальным решением будет переход в левую часть, и присоединение большого числа вершин, соединенных дешевыми ребрами.



Пример графа, где алгоритм жадного поиска по кратчайшим путям проигрывает наивному

Используя описанные выше принципы построения «плохого» графа, для данного алгоритма можем построить граф, где он будет работать хуже, чем наивный жадный:



.

Grasp алгоритмы

Представленные выше алгоритмы были адаптированы для использования процедуры жадного случайного адаптивного поиска (Greedy randomized adaptive search procedure, GRASP), впервые описанной T.A. Feo и M.G.C. Resende в 1995 году[7]

Идея процедуры состоит в случайном выборе из множества локально наиболее оптимальных решений на каждом шаге. Оптимальность решения определяется некоторой функцией полезности.

На каждом шаге строится множество допустимых решений, для которых находится минимум и максимум функции полезности.

Далее из полученного множества отбираются решения, отличающиеся от лучшего не более чем на $\alpha * (max - min)$, где α параметр, который подбирается экспериментально. При $\alpha = 0$ процедура вырождается в обычный жадный алгоритм. Если же принять $\alpha = 1$, получим случайный поиск из множества допустимых решений.

Данная процедура прогоняется несколько раз, после чего выбирается лучшее из решений.

Описанный способ позволяет значительно улучшить и стабилизировать результаты жадного алгоритма, давая возможность алгоритму выбраться из локального максимума.

Тестирование алгоритмов.

Для тестирования алгоритмы были запущены на случайно сгенерированных графах. Для графов малых размеров сравнение проводилось с результатом точного алгоритма. Для тестирования на графах больших размеров, использовался алгоритм Гоеманса и Вильямсона[8] для решения вариации задачи поиска на данном графе оптимального дерева Штейнера с целевой функцией в виде разности между доходом и стоимостью вершин, после чего вес полученного дерева принимался за бюджетное ограничение, а доход от входящих в него вершин использовался, как оценка оптимального решения бюджетной версии задачи.

В таблице приведен процент графов, на которых данный алгоритм нашел решение, доход от которого составляет более обозначенного процента от максимального. Тестирование проведено на различных значениях плотности графа и различных бюджетных ограничениях.

Наивный жадный алгоритм

Результаты тестирования на графе из 16 вершин, с весами и ребрами распределенными по закону гаусса с матожиданием 20 и дисперсией 5.

p	0,4				0,6				0,8		
budg	90	120	150	180	90	120	150	180	120	150	180
100%	17	21	11	16	14	10	10	22	6	12	33
98%	37	32	33	42	27	28	38	49	25	35	55
95%	59	52	55	66	53	50	62	79	54	66	84

Результаты тестирования на графе из 100 вершин, с весами и ребрами распределенными по закону гаусса с матожиданием 20 и дисперсией 5.

p	0,4	0,6	0,8
100%	0	0	0
98%	46	33	54
95%	100	100	100

Жадный поиск по кратчайшим путям

Результаты тестирования на графе из 16 вершин, с весами и ребрами распределенными по закону гаусса с матожиданием 20 и дисперсией 5.

p	0,4				0,6				0,8		
budg	90	120	150	180	90	120	150	180	120	150	180
100%	14	9	11	14	6	10	16	32	8	5	47
98%	28	23	25	36	15	17	34	64	16	35	77
95%	41	44	57	62	37	38	68	86	43	73	93

Результаты тестирования на графе из 100 вершин, с весами и ребрами распределенными по закону гаусса с матожиданием 20 и дисперсией 5.

p	0,4	0,6	0,8
100%	15	24	35
98%	100	100	99

Grasp на основе наивного жадного алгоритма

Результаты тестирования на графе из 16 вершин, с весами и ребрами распределенными по закону гаусса с матожиданием 20 и дисперсией 5.

p	0,4				0,6				0,8		
budg	90	120	150	180	90	120	150	180	120	150	180
100%	17	16	8	15	11	14	11	21	5	10	31
98%	34	25	29	41	25	30	34	39	26	31	56
95%	50	53	53	70	49	56	66	78	51	62	83

Результаты тестирования на графе из 100 вершин, с весами и ребрами распределенными по закону гаусса с матожиданием 20 и дисперсией 5.

p	0,4	0,6	0,8
100%	0	0	0
98%	40	38	47
95%	100	100	100

Grasp на основе жадного поиска по кратчайшим путям

Результаты тестирования на графе из 16 вершин, с весами и ребрами распределенными по закону гаусса с матожиданием 20 и дисперсией 5.

p	0,4				0,6				0,8		
budg	90	120	150	180	90	120	150	180	120	150	180
100%	24	24	27	30	15	18	28	51	15	31	58
98%	40	40	57	64	36	49	61	80	48	66	91
95%	63	79	82	89	59	74	82	95	75	91	99

Результаты тестирования на графе из 100 вершин, с весами и ребрами распределенными по закону гаусса с матожиданием 20 и дисперсией 5.

p	0,4	0,6	0,8
100%	23	31	40
98%	100	100	100

Заключение

В данной работе были рассмотрены и протестированы четыре алгоритма, позволяющие эффективно решать задачу оптимального расширения сетевых структур при наличии ограничения на бюджет.

В результате тестов данные алгоритмы показали хорошую точность и могут быть успешно реализованы на практике

Планируется продолжать работу, доведя алгоритмы до конечного программного продукта, востребованного при решении реальных задач. Так же имеет смысл продолжить работать над улучшением описанных методов.

Список литературы

1. Segev, A. The node-weighted Steiner tree problem, Networks 17: 1–17, 1987
2. Ivanov, A. O. and Tuzhilin, A. A. Minimal Networks: The Steiner Problem and Its Generalizations. Boca Raton, FL: CRC Press, 1994.
3. 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
4. Joseph. B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. // Proc. AMS. 1956. Vol 7, No. 1. C. 48-50
5. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to Algorithms. — 3rd. — MIT Press, 2009.
6. Floyd, Robert W. “Algorithm 97: Shortest Path.” Communications of the ACM 5, no. 6 (June 1, 1962): 345.
7. T.A. Feo and M.G.C. Resende (1995) Greedy randomized adaptive search procedures. Journal of Global Optimization, 6:109–133, 1995.
8. Goemans, M. X. and Williamson, D. P. A general approximation technique for constrained forest problems, SIAM Journal on Computing 24: 296–317, 1995