Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

ЗАДАЧА О ПАРОСОЧЕТАНИЯХ В ГРАФЕ: РЕАЛИЗАЦИЯ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Бирюкова Алексея Сергеевича

Научный руководитель:
зав. кафедрой ИиП, к.ф.-м.н. М.В. Огнева
(подпись, дата)

Зав. кафедрой:
к.ф.-м.н. М.В. Огнева
(подпись, дата)

Саратов 2017

ВВЕДЕНИЕ

Актуальность темы. В современном мире возникает огромное количество задач, решать которые помогают информационные технологии. Рассмотрим ситуацию, возникшую в некотором крупном брачном агентстве. У компании есть задача свести между собой наибольшее количество мужчин и женщин при поставленном условии: они должны быть уже знакомы между собой. Для автоматизации решения данной проблемы с помощью ЭВМ, используется теория графов, а если точнее, теория паросочетаний. И это далеко не единственный пример применения теории паросочетаний. Можно подбирать, например, компаниям или заводам сотрудников на работу или распределять пациентов больниц по операционным. Автоматизация данного процесса способна приносить реальную прибыль частным компаниям и успешно решать задачи муниципального и даже федерального уровней.

Цель бакалаврской работы – определение наилучшего с точки зрения производительности алгоритма поиска наибольшего паросочетания и его использование для решения задачи о назначениях.

Поставленная цель определила **следующие задачи**:

1. сформулировать задачу поиска наибольшего паросочетания;
2. изучить основные алгоритмы поиска наибольшего паросочетания;
3. изучить основные алгоритмы поиска максимального потока в сети;
4. применить абстракцию потока для нахождения наибольшего паросочетания;
5. выполнить программные реализации алгоритмов;
6. произвести сравнительную характеристику алгоритмов по времени их работы;
7. сформулировать задачу о назначениях;
8. изучить алгоритмы решения задачи о назначениях. Выполнить программные реализации;
9. выполнить сравнение выбранных алгоритмов.

Методологические основы разработки и анализа алгоритмов представлены в работах Асанова М. О.[1], Кормена Т. [4], Липского В. [11], Ху Т. Ч., Шинг М. Т. [12], Гэри М., Джонсона Д. [13], Клейнберга Дж., Тардос Е.[14].

Теоретическая и/или практическая значимость бакалаврской работы. Заключается в исследовании алгоритмов решения задачи поиска наибольшего паросочетания в графе, возможность понять, какой из алгоритмов выгоднее использовать на тех или иных графах, а также внедрение в венгерский алгоритм решения задачи о назначениях лучшего, по итогам анализа, алгоритма.

Структура и объём работы. Бакалаврская работа состоит из введения, трех разделов, заключения, списка использованных источников и пяти приложений. Общий объем работы — 88 страниц, из них 43 страницы — основное содержание, включая 14 рисунков и 5 таблиц, 45 страниц приложений, список использованных источников информации — 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Паросочетания в графе» посвящен разбору основных алгоритмов поиска наибольшего паросочетания в двудольном и алгоритмов поиска максимального потока в сети, решающих задачу, к которой можно свести исследуемую задачу. Кроме этого в разделе даны основные понятия, используемые на протяжении всей работы.

Пусть V — непустое конечное множество вершин, V^2 — множество всех двухэлементных подмножеств из V. Обыкновенным графом G называется пара множеств (V, E), где E — произвольное подмножество из V^2. Если $E \subseteq V$ x V, то элементы множества E называются дугами, а сам граф G — ориентированным. Если же $E \subseteq \{\{x, y\} : x, y \in V \wedge x \neq y\}$, то элементы множества E называют ребрами, а граф G неориентированным[1].

Путем в графе называется последовательность вершин, в которой каждая следующая вершина (после первой), является смежной с предыдущей вершиной на этом пути. Все вершины и ребра, составляющие простой путь, различны. Циклом называется простой путь, начальная и конечная вершины которого совпадают[2].

Граф $G = (V, E)$ называется двудольным, если множество его вершин V можно разбить на 2 непересекающихся подмножества V_1 и V_2 таких, что каждое ребро графа имеет одну вершину в V_1, а другую в V_2[3].

Паросочетанием в графе $G = (V, E)$ называется подмножество его ребер такое, что каждой вершине инцидентно не более одного ребра. Паросочетание, содержащее наибольшее количество ребер, называется наибольшим, а паросочетание, не содержащееся ни в каком другом паросочетании — максимальным[2][4]. Паросочетание, порядок которого равен порядку графа, т. е. включает в себя все вершины двудольного графа, называется совершенным. Совершенное паросочетание является и наибольшим, и максимальным[5].

Задача поиска наибольшего паросочетания формулируется следующим образом: для произвольного двудольного графа G найти паросочетание наибольшей мощности. Самый очевидный из способов решения задачи — последовательная генерация всех возможных паросочетаний и выбор наилучшего из них не рассматривается в работе в силу своей заведомой неперспективности. Время его выполнения есть экспоненциальная функция от числа вершин[6].

Чередующейся цепью называется простой путь, в котором при рассмотрении любых 2-х смежных ребер одно принадлежит, а другое не принадлежит паросочетанию[7]. Чередующаяся цепь называется увеличивающей (аугментальной), если вершины, являющиеся ее концами, являются свободными, т. е. не принадлежат паросочетанию.

Алгоритм Куна поиска наибольшего паросочетания заключается в прямом применении теоремы Бержа, по которой паросочетание является наибольшим тогда и только тогда, когда не существует аугментальной цепи относительно паросочетания[8]. Пусть $M = \emptyset$ — паросочетание, не содержащее ни одного ребра. Пока в графе G существует чередующаяся цепь p, необходимо выполнять чередование паросочетания вдоль нее: $M = M \cup p$. Как только чередующаяся цепь не будет найдена, работа алгоритма должна быть остановлена[9].

Алгоритм Хопкрофта-Карпа, подобно алгоритму Куна, выполняет увеличение паросочетания вдоль чередующейся цепи, но вместо того, чтобы находить один увеличивающий путь, алгоритм, используя идеи Диница, находит все такие пути. Алгоритм работает следующим образом:

0. вводится фиктивная вершина v_{-1};

1. с помощью поиска в ширину, множество вершин V_1 первой доли графа разбивается на слои. Изначально все вершины $v \in V_1$ являются свободными, т. е. ни одно инцидентное им ребро не принадлежит паросочетанию, и образуют единый монолитный слой. В свою очередь, введенная фиктивная вершина принадлежит слою уровнем выше по отношению к первой найденной свободной вершине. На последующих итерациях, начиная со свободных вершин, поиск в ширину пересчитывает номера слоев для вершин, смежных с ними и далее смежных всем рассматриваемым вершинам, в соответствии со следующей формулой: $d[v] = d[u] + 1$, где $d[v_i]$ – номер слоя, в который помещена вершина v_i, $u \in V_1$— рассматриваемая, изначально свободная вершина, $v \in V_1$ — вершина, смежная с вершиной $w \in V_2$ в случае, если ребро (u, w) принадлежит паросочетанию. Если же ребро (u, w) не принадлежит паросочетанию, то вершина u соответствует введенной фиктивной вершине v_{-1}. Именно на этом этапе разбиение на слои будет завершено;

2. посредством поиска в глубину алгоритм выполняет увеличение паросочетания по ребрам со свободными вершинами в предыдущем слое[10].

Сформулируем задачу поиска максимального потока в сети. Пусть задана сеть $G = (V, E)$ с истоком s и стоком t. Необходимо найти такой поток (функцию $f: V \text{ x } V \to \mathbb{R}$, удовлетворяющую ограничению по пропускной способности, свойству кососимметричности и условию сохранения потока), что никакой другой поток из s в t не обладает большей мощностью. Такой поток называется максимальным, а задача его построения — задачей поиска максимального потока в сети[2]. Решение задачи поиска максимального потока в сети сводится к задаче поиска наибольшего паросочетания в двудольном графе следующим алгоритмом:

1. каждое ребро $(u, v)_{u \in V_1, v \in V_2}$ преобразовать в единичную дугу, исходящую из вершины u;
2. добавить узел s и единичную дугу (s, u) для каждой $u \in V_1$;
3. добавить узел t и дугу единичного веса (v, t) для каждой $v \in V_2$[15].

Теперь, когда известно, как решать исследуемую проблему с помощью алгоритмов поиска максимального потока, рассмотрим их.

Метод Форда-Фалкерсона является итеративным. Вначале величине потока каждого ребра присваивается значение 0. На каждой итерации величина потока увеличивается посредством поиска увеличивающего пути (т. е. некоторого пути от истока s к стоку t, вдоль которого можно послать больший поток) и последующего увеличения потока на остаточную пропускную способность $c_f(p) = c(u, v) - f(u, v), u, v \in V$[2]. Как только отыскать такой путь становится невозможно, алгоритм завершает свою работу. Поиск увеличивающего пути может осуществляться посредством разных алгоритмов. В работе выполнены реализации с помощью поиска в глубину и поиска в ширину.

Высотой вершины называется функция $h: V \to \mathbb{N}$ такая, что для любого остаточного ребра верно: $h_s = |V|$, $h_t = 0$ и $h_u \leq h_v + 1$.

Идея *алгоритма проталкивания предпотока* состоит в следующем. Поток может проталкиваться исключительно вниз. Иными словами высота

вершины, из которой происходит проталкивание, должна быть больше высоты вершины, получающей поток. Высоты истока и стока являются фиксимрованными величинами и равны $|V|$ и 0, соответственно. Высота всех остальных вершин изначально равна нулю и динамически изменяется со временем. Первым делом алгоритм посылает максимально возможный поток вниз от истока к смежным вершинам, т. е. количество потока, полностью исчерпывающее пропускную способность выходящих из истока ребер. При посещении потоком некоторой транзитной вершины, он накапливается внутри нее и со временем проталкивается вниз. Если все не заполненные потоком ребра, выходящие из некоторой вершины u, ведут к вершинам, которые лежат на одном уровне с v или находятся выше нее, то для «сливания» избыточного потока из вершины u необходимо «повысить» ее настолько, чтобы она стала на единицу выше, чем самая низкая из смежных с ней вершин. Таким образом, рано или поздно, весь возможный поток «вливается» в сток[4].

Алгоритм «поднять в начало» — одна из возможных реализаций метода проталкивания предпотока. Внутри себя алгоритм поддерживает список вершин сети. Он, начиная с начала, анализирует список, как результат выбирает некоторую переполненную вершину u, а затем «разгружает» ее, «сливая» из нее весь имеющийся избыток. Поднятие вершины в буквальном смысле предопределяет ее перенос в начало списка, после которого, начиная со следующей по отношению в поднятой вершине, возобновляется сканирование списка[4].

Также в разделе приведены пошаговые примеры работы указанных алгоритмов, опубликованные автором в соавторстве с М. А. Ищук и М. В. Огневой в [14].

Второй раздел «Сравнительная характеристика алгоритмов поиска наибольшего паросочетания» посвящен анализу времени работы реализованных алгоритмов поиска наибольшего паросочетания.

Все алгоритмы реализованы и прошли тестирование с замерами времени работы. Для получения практического времени работы алгоритмов использовался ноутбук со следующими характеристиками: процессор Intel® Core™ i7 4700MQ, оперативная память (Мб) — 8192.

Тестирование производительности алгоритмов производилось на полном двудольном графе с числом вершин $N = 2000, 4000, 10000, 20000$; на графе, с числом ребер практически в 2 раза меньше, чем в полном двудольном; на графах с различными соотношениями вершин в разных долях. Кроме того, упомянуты результаты, полученные на «небольших» по количеству вершин ($N = 100, 200, 300$) и разреженных графах.

Самым стабильным алгоритмом на большинстве тестов признан алгоритм *Хопкрофта-Карпа*.

Третий раздел раздел «Задача о назначениях» посвящен разбору основных алгоритмов решения задачи о назначениях, проведению их сравнительной характеристики.

В задаче о назначениях необходимо назначить сотрудников некоторой абстрактной компании на определенные работы, причем каждый сотрудник может выполнять ту или иную работу разное количество человеко-часов. Целью задачи является поиск оптимального распределения работников по заявленным работам[16]. Пусть матрица $(a_{i,j})$ $, i, j = \overline{1, N}$ представляет собой собой матрицу времени выполнения (или матрицу стоимости выполнения) некоторых абстрактных заданий сотрудниками некоторой фирмы.

Метод Мака решения задачи о назначениях можно описать следующим образом. Разделим множество столбцов на 2 множества C и C', где множество C — множество выбранных столбцов, а множество C' — множество невыбранных. Очевидно, что изначально $C = \emptyset$, $C' = \{1, \dots, n\}$. Введем помимо этого еще множества помеченных элементов F и множество временно помеченных элементов F'.

Первым делом необходимо выбрать в каждой строке по одному минимальному элементу и добавить их в F. Если в каждом столбце помечено ровно по одному элементу, то помеченные элементы являются базисом и определяют оптимальный выбор.

Распишем алгоритм по шагам:

1. Выбрать из множества C' столбец, содержащий наибольшее число помеченных элементов, и перевести его в множество C;
2. Пусть минимальный элемент множества C из строки i равен min_i, а из множества C' — min'_i. Тогда введем обозначение $\min = \min\limits_{i=\overline{1,N}}(min'_i - min_i)$. Обозначим элемент, из которого производилось вычитание через b_{min};
3. Увеличить все элементы множества C на посчитанный на предыдущем шаге min;
4. $F' = F' \cup \{b_{min}\}$;
5. Пусть b_{min} содержится в столбце col. Если в нем помечено любым из способов более двух столбцов, то $C = C \cup \{col\}$, $C' = C' \setminus \{col\}$, $F' = F' \setminus \{b_{min}\}$, и вернуться ко второму пункту алгоритма;
6. $F' = F' \setminus \{b_{min}\}$, $F = F \cup \{b_{min}\}$;
7. Вернуться к шагу 1 для начала новой итерации. Алгоритм прекращает свою работу, как только в каждом столбце появляется ровно по одному помеченному элементу[17][18].

В основе *метода ветвей и границ* лежит идея последовательного разбиения множества допустимых решений на подмножества (стратегия «разделяй и властвуй»)[19]. На каждом шаге метода элементы разбиения проверяются на предмет наличия оптимального решения. На каждом таком подмножестве вычисляется минимум и если он оказывается не меньше лучшего результата, то данное подмножество должно быть отброшено.

Решение задачи будет получено, как только будут отброшены все подмножества, а пока этого не произошло, из неотброшенных подмножеств выбирается наиболее перспективное подножество и т. д.[19].

Распишем алгоритм решения задачи по шагам:

0. ввести переменную $S = 0$ для хранения суммы стоимостей уже выполненных назначений;
1. вычеркнуть первую строку и j-ый столбец в зависимости от того, на какую работу назначается первый сотрудник и вычислить значения $\varphi_{1j} = a_{1j} + \min(\sum row + \sum col)$, где $\sum row$ — сумма минимальных значений невычеркнутых строк, а $\sum col$ — сумма минимальных значений невычеркнутых столбцов;
2. пусть $min = \min\limits_{j} \varphi_{1j}$. Назначить сотрудника 1 на работу с номером min. Исключить из дальнейшего рассмотрения строку 1 и столбец min. Добавить min к переменной S;
3. для каждой следующей строки повторять алгоритм, вычисляя на каждом шаге значения $\varphi_{ij} = S + a_{ij} + \min(\sum row + \sum col)$, в последствии выбирая минимум из этих значений и так же, как и на шаге 2, прибавляя это значение к S.

Основная идея *венгерского алгоритма* состоит в сведении рассматриваемой задачи к задаче выбора N нулевых элементов — по одному в каждой строке и каждом столбцов матрице стоимостей, каждый элемент которой является неотрицательным[20].

Потенциалом называются 2 произвольных массива чисел величины N, для которых выполняется условие $u_i + v_j \leq a_{ij}, i, j = \overline{1, N}$. Здесь значение u_i, перед стартом работы алгоритма, соответствует минимальному значению в каждой из строк матрицы стоимостей, а v_j — минимуму в каждом столбце по следующей характеристике: $a_{ij} - u_i$.

Сам алгоритм можно описать следующим образом:

1. Построить по матрице A двудольный граф G. Пусть $M = \emptyset$ — паросочетание в этом графе;
2. На каждом шаге алгоритма производятся попытки не изменяя потенциалов увеличить мощность паросочетания M в графе, состоящем только из вершин, для которых выполняется условие $u_i + v_j = a_{ij}$;
3. Если попытка увеличения мощности паросочетания не увенчалась успехом, необходимо произвести пересчет потенциала. Для этого считается величина $\alpha = \min_{i \in \text{Vis1}, j \in \text{NotVis2}} (a_{ij} - u_i - v_j)$, где множество $\text{Vis1} \subset V_1$ является множеством посещенных в процессе работы алгоритма Куна вершин, а множество $NotVis2 \subset V_2$ — множество вершин, так и не посещенных алгоритмом поиска наибольшего паросочетания. Сам пересчет потенциала принимает вид: $u_i = u_i - \alpha, v_j = v_j + \alpha$. Получившийся потенциал, очевидно, останется корректным.
4. В конечном итоге будет найден потенциал, соответствующий совершенному паросочетанию M.

Решением задачи является сумма стоимостей ребер полученного совершенного паросочетания. Классическим вариантом реализации данного алгоритма является поиск совершенного паросочетания с помощью алгоритма Куна. В работе, выполнены 2 реализации: классическая — с помощью алгоритма Куна и модификация — использование алгоритма Хопкрофта-Карпа.

Кроме венгерского алгоритма рассматривался еще один алгоритм, в который закладывается абстракция паросочетаний. *Алгоритм поиска наибольшего паросочетания минимальной стоимости*:

1. построить из матрицы стоимостей двудольный граф G;
2. преобразовать граф G по алгоритму, описанному в пункте 1.3.3 данной работы. Каждому фиктивному ребру (s, v_i) и (u_j, t) поставить в соответствие стоимость, равную нулю, каждому

ребру (v_i, u_j) присвоить стоимость из заданной матрицы стоимостей a_{ij}. Наконец, каждому ребру (u_j, v_i) задать стоимость $-a_{ij}$;

3. с помощью алгоритма поиска кратчайших путей выполнять поиск путей с минимальной стоимостью в остаточном графе G_M[15] и использовать их для увеличения мощности искомого паросочетания. Стоит отметить, что алгоритм поиска кратчайших путей следует подбирать таким образом, чтобы он мог корректно работать при появлении в графе отрицательных циклов. В настоящей работе использовался алгоритм Беллмана-Форда.

В конце главы выполнен анализ времени работы алгоритмов для случайной матрицы стоимостей.

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены и реализованы алгоритмы поиска наибольшего паросочетания, а именно алгоритм Хопкрофта-Карпа и алгоритм Куна. Помимо них, были изучены, реализованы и сведены к задаче поиска наибольшего паросочетания алгоритм Форда-Фалкерсона, алгоритм Эдмондса-Карпа и алгоритм «поднять в начало» поиска максимального потока в сети. На практике, практически на все тестах наилучшую производительность показал алгоритм Хопкрофта-Карпа. Алгоритм Куна выгоднее всего использовать, когда количество вершин в левой доле двудольного графа сильно меньше, чем в правой. Во всех остальных случаях, несмотря на свою специализацию, алгоритм Куна показал себя слабее модифицированного алгоритма Форда-Фалкерсона с использованием DFS для поиска аугментальных путей, который, в свою очередь, показал достаточно стабильное время работы на всех тестах.

Задача о назначениях решалась с помощью четырех алгоритмов: венгерским алгоритмом, алгоритмом Мака, методом «ветвей и границ» и алгоритмом поиска наибольшего паросочетания минимальной стоимости. Поиск паросочетания в венгерском алгоритме осуществлялся двумя способами: классическим — с использованием алгоритма Куна и

модифицированным — с помощью алгоритма Хопкрофта-Карпа. Непревзойденную производительность показал как раз венгерский алгоритм. По времени работы он превзошел своих оппонентов в сотни раз. А вот алгоритм поиска наибольшего паросочетания минимальной стоимости себя не оправдал, показав худшее время работы из рассмотренных алгоритмов. Таким образом, можно сделать вывод, что грамотное применение алгоритмов паросочетаний позволяет существенно повысить производительность алгоритмов при решении прикладных задач.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Асанов М. О., Баранский В. А., Расин В. В.* Дискретная математика: графы, матроиды, алгоритмы. — Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001. — 288 с.
2. *Седжвик Р.* Фундаментальные алгоритмы на C++. Алгоритмы на графах: Пер. с англ. — СПб: ООО «ДиаСофтЮП», 2002. — 496 с.
3. *Скиена С.* Алгоритмы. Руководство по разработке. — 2-е изд.: Пер. с англ. — СПб.: БХВ-Петербург, 2011. — 720 с.
4. *Кормен Т. и др.* Алгоритмы: построение и анализ, 2-е издание: Пер. с англ. — М.: Издательский дом «Вильямс», 2005. — 1296 с.
5. *Кирсанов М. Н.* Графы в Maple. Задачи, алгоритмы, программы. — М.: Издательства ФИЗМАТЛИТ, 2007. — 168 с.
6. *Ахо А. и др.* Структуры данных и алгоритмы. : Пер с англ.: М. : Издательский дом «Вильямс», 2003 с. — 384 с.
7. *Майника Э.* Алгоритмы оптимизации на сетях и графах : Пер. с англ. — М.: Мир, 1981. — 323 с.
8. *Кристофидес Н.* Теория графов. Алгоритмический подход. : Пер. с англ. : М.: Мир, 1978. — 432 с.
9. Алгоритм Куна для поиска максимального паросочетания — Викиконспекты. Университет ИТМО [Электронный ресурс]. URL: http://neerc.ifmo.ru/wiki/index.php?title=%C0%EB%E3%EE%F0%E8%F2%EC_

%CA%F3%ED%E0_%E4%EB%FF_%EF%EE%E8%F1%EA%E0_%EC%E0%EA%F1%E8%EC%E0%EB%FC%ED%EE%E3%EE_%EF%E0%F0%EE%F1%EE%F7%E5%F2%E0%ED%E8%FF (Дата обращения: 25.05.2017)

10. *Актанорович С. В. и др.* Алгоритмы и структуры данных. Потоковые алгоритмы : метод. пособие по курсу «Теория графов. Потоковые алгоритмы» для студ. спец. I-31 03 04 «Информатика» всех форм обуч. — Минск: БГУИР, 2011. — 47 с.

11. *Липский В.* Комбинаторика для программистов: Перевод с польского — М.: «Мир», 1988. — 200 с.

12. *Ху Т. Ч., Шинг М. Т.* Комбинаторные алгоритмы / Пер. с англ. Нижний Новгород : Изд-во Нижегородского госуниверситета им. Н. И. Лобачевского, 2004. — 330 с.

13. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи: Пер. с англ. — М. : Мир, 1982. — 416 с.

14. *Бирюков А.С., Ищук М.А., Огнева М.В.* Использование алгоритмов теории графов в задачах по программированию повышенной сложности//Информационные технологии в образовании. 2016. с. 25-30.

15. *Клейнберг Дж., Тардос Е.* Алгоритмы: разработка и применение. Классика Computers Science / Пер. с англ. Е. Матвеева. — Спб.: Питер, 2016. — 800 с.

16. *Таха Хэмди А.* Введение в исследование операций. 6-е издание.: Пер. с англ.: — М.: Издательский дом «Вильямс», 2001. — 912 с.

17. *Банди Б.* Основы линейного программирования /Пер. с англ. - М.: Радио и связь, 1989. - 176с.

18. *Муравьев, В.И.* Модели и методы оптимизации в экономике и менеджменте: учеб. пос. для практ. занятий / В.И. Муравьев [и др.]; Балт. гос. техн. ун-т.– СПб., 2008.: 197 с.

19. Метод ветвей и границ. Задача о назначениях. Задача о рюкзаке. Задача коммивояжера[Электронный ресурс]. URL: http://www.studfiles.ru/preview/3654868/ (Дата обращения: 29.05.2017)

20. *Зандер Е. В.* Исследование операций в экономике : учеб. пособие. — Сибирский федеральный ун-т. Красноярск: 2007. — 202 с.