САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ ЭНЕРГЕТИЧЕСКИХ СИСТЕМ

НИКОЛАЕВ Константин Игоревич

Выпускная квалификационная работа

Задача размещения элементов цепи поставок

Уровень образования: магистратура

Направление 01.04.01 «Прикладная математика и информатика»

Основная образовательная программа ВМ.5505.2019 «Математическое и информационное обеспечение экономической деятельности»

Научный руководитель:
кандидат физ.-мат. наук, доцент
Лежнина Елена Александровна

Рецензент:
Федеральное государственное бюджетное учреждение науки
Институт прикладных математических исследований
Карельского научного центра Российской академии наук
Перцовский Александр Константинович

Санкт-Петербург

2021

Оглавление

1. Введение

Задача размещения – задача, в которой необходимо найти наилучшее местоположение для различных элементов цепи поставок. Будь то производственные объекты, складские комплексы или распределительные центры, магазины розничной торговли, объекты обслуживания и тому подобное. Такие решения о местоположении объектов имеют сильное влияние на показатели компании в течение долгого времени, так как носят долгосрочный характер. Многие предприятия упускают выгоду или даже терпят убытки, так как качественно не берут во внимания вопрос размещения мощностей и оптимизации конфигурации цепи поставок.

Для того, чтобы решение о размещении того или иного объекта было эффективным оно также должно учитывать множество факторов. Некоторые из этих факторов: спрос потребителей, операционные издержки, конкуренты, налоги, курсы валют, расстояния до других предприятий, поставщики и т. д.

Значительное число таких проблем размещения может быть сформулировано и эффективно решено с помощью математических оптимизационных моделей.

Данная работа состоит из восьми разделов: введение, цель работы, обзор литературы, задачи размещения элементов цепи поставок, практический эксперимент, заключение, список литературы и приложение.

Во введении определяется объект исследования - задача размещения и её вариации.

В разделе 2 определяются цели работы и описываются этапы их достижения. Задачи исследования основаны на потребности работодателя - реальной компании, что подтверждает практическую значимость результатов исследования.

Обзор литературы представляет обобщения и критический анализ результатов, полученных отечественными и зарубежными исследователями. Обзор в том числе содержит недавние научные публикации, статьи и книги, что подтверждает актуальность и теоретическую значимости темы данной работы.

В разделе 4 приведены постановки задач (простейшая задача размещения, многостадийная задача размещения, задача Вебера, задача конкурентного размещения, задача о p - медиане) и методы их решения.

Раздел 5 содержит практическую часть: описание способов решения поставленных задач, адаптацию алгоритма муравьиной колонии под задачу размещения, его программную реализацию, анализ чувствительности параметров алгоритма и результаты вычислительного эксперимента.

В заключении кратко описаны результаты научной работы. В предпоследнем разделе размещен перечень научных статей и книг, на которые опирается научно-исследовательская работа.

В приложении содержится разработанный программный код, реализующий алгоритм, описанный в пятом разделе.

2. Цель работы

Целью работы является исследование задачи оптимального размещения элементов цепи поставок и дальнейшая практическая реализация эффективного алгоритма решения. Решение практической бизнес-задачи работодателя для получения оценки размещения объектов цепи поставок при выходе на новые рынки присутствия.

Достижение поставленной цели включает в себя следующие этапы (ряд задач):

1. Изучение и анализ существующих классов моделей задач, их применимости;
2. Исследование различных расширений и усложнений моделей размещения элементов цепи поставок и методов их решения;
3. Определение параметров (факторов) и сбор, обработка и анализ необходимых исходных данных для формализации подходящей математической модели, реализации оптимизационного алгоритма и проведения практического эксперимента;
4. Разработка программного кода оптимизационного алгоритма для решения задачи размещения;
5. Проведение анализа чувствительности параметров применяемого алгоритма;
6. Проведение ряда тестов на производительность полученного кода;
7. Проведение вычислительного эксперимента на реальных данных для решения прикладной бизнес-задачи.

3. Обзор литературы

Задача оптимизации объектов цепи поставок и размещения производственных и логистических мощностей в научной литературе относится к теории о размещении мощностей (facility location theory). Она берет свое начало в начале двадцатого века, когда немецкий исследователь Альфред Вебер сформулировал задачу определения местоположения завода, реализующего товар определенному числу клиентов.

В исследованиях [1-7] рассматривается простейшая задача размещения для которой найдены решения как точными алгоритмами, так и приближенными с гарантированными оценками точности, Лагранжевы эвристики, вероятностные итерационные алгоритмы локального поиска. Также выделены полиномиально разрешимые классы задач.

В статьях [8] и [9] рассматриваются задачи размещения с ограничением на мощности предприятия и многостадийная задача размещения. Они являются усложнением простейшей задачи размещения. В многостадийной задаче размещения предполагается, что перед тем, как попасть к потребителю товар проходит ряд производственных объектов (технологических ассоциаций). В задаче размещения с ограничением на мощности каждый производственный объект может выпускать товар в ограниченных количествах. Такое предположение меняет математическую модель и сильно усложняет методы её решения. Наиболее продуктивными подходами к решению оказываются метод Лагранжевых релаксаций и метод генерации столбцов.

Задача местоположения Вебера (также называемая проблемой Ферма-Вебера. Ферма предложил геометрический вариант для трех точек) является

базовой моделью в теории местоположения, которая получила значительное внимание в научной литературе [33-35].

Дрезнер, Мехрез и Весоловский исследовали в [36] задачу Вебера для случая, когда функции расстояния постоянны после заданных пороговых значений. Эта постановка была названа задачей местоположения объекта на ограниченных расстояниях.

Наиболее популярным методом решения задачи Вебера в евклидовом пространстве является одноточечная итерационная процедура, впервые предложенная Вайсфельдом [37].

В [38] рассмотрена дискретная задача Вебера для которой предлагается последовательный детерминированный алгоритм, находящий точное решение задачи для k-дерева и конечного множества позиций размещения. Используется динамического программирования на основе дерева декомпозиции.

В исследовании А. Утешева [39] предложено аналитическое решение задачи Вебера о построении плоской взвешенной сети с минимальной стоимостью, соединяющей четыре терминала с двумя дополнительными объектами. Тем самым подтверждая геометрическое решение Г. Пика, ранее приведенное им без доказательства. Также в работе приведены условия существования сети в предполагаемой топологии и явные формулы для координат объектов. Полученные результаты использованы для исследования динамики сети при изменении параметров.

В статье [16] было положено начало исследований конкурентных задач размещения. В ней рассматривался процесс выбора оптимального местоположения объекта и выбор модели ценообразования двумя компаниями в условиях конкуренции.

В исследованиях [17], [18] развиваются представления конкурентной задачи размещения в концепции теории игр, на которых во многом основываются современные математические формализации данной задач. В задаче размещения в условиях конкуренции компании выступают в роли игроков, которые могут действовать как последовательно, так и параллельно. Игроки по очереди принимают свои решения, руководствуясь предыдущими действиями своего оппонента.

В статьях [19, 20] проводятся исследования при последовательной конкуренции в задаче нахождения оптимального местоположения. Главное особенностью постановки является наличие двух типов игроков – Лидера и Последователя. Равновесие Штакельберга [21] применяется в качестве концепции решения таких задач.

В [22], [23] задача конкурентного размещения дополняется тем, что помимо определения местоположения, игроки также определяют цены на выпускаемую продукцию. В [24–26] рассматриваются различные стратегии ценообразования при такой постановки задачи.

В исследованиях [27], [28] для соответствующей игры Штакельберга с учетом оптимального расположения производственных объектов и стратегий ценообразования сформулирована новая математическая постановка. В ней компания Лидер и компания Последователь последовательно выбирают местоположения размещения объектов, и, после их открытия, на рынках начинается ценовая конкуренция, описываемая моделью Бертрана. В результате одна из компаний устанавливает свою монополию и определяет цену на товар.

Исследования процессов размещения объектов производства и выбора стратегии ценообразования в рамках двухуровневых математических моделях

[29] приводит к большей вычислительной сложности. Однако так как такие модели оказываются более адекватными чем одноуровневые, интерес к ним постоянно растет.

На данный момент выработан широкий спектр видов задач с различными целевыми функциями и ограничительными условиями оптимизации, процедур расчетов, методов и пр.

В рамках сетевых моделей оптимизации цепь поставок представляется в виде ориентированного графа, в котором ребра обозначают маршруты транспортировки, а вершины – расположения точек производства, хранения или сбыта товаров. Потенциальные местоположения объектов цепи поставок ограничены заданным подмножеством вершин графа и точками на его ребрах. При таком представлении целевая функция минимальной суммы предполагает нахождение такого местоположения объектов цепи поставок, при котором сумма расстояний от вершин графа до ближайших к ним локаций будет минимальной. Целевая функция также дополняется набором условий: на количество объектов, постоянные затраты, переменные затраты и прочие). Математическое описание моделей данного класса часто очень близка к другому классу моделей – моделям смешанной дискретной оптимизации размещения мощностей.

Для решения задач о поиске местоположения элементов цепи поставок, сформулированных в терминах теории графов, чаще всего применяются такие методы как: метод кратчайшего пути, покрывающего дерева, решения задачи о максимальном потоке и прочие.

Для поиска оптимального решения задач по размещению одного объекта предлагается повторяющаяся процедура, по которой последовательно рассчитываются новые координаты расположения объекта. Процедура

заканчивается, когда большее сокращение затрат перестает иметь практический смысл. Вместе с тем фирмы сталкиваются с задачами, в которых необходимо разместить в цепи поставок более одного объекта. Сложность решения таких задач заключается в том, что в отсутствии ограничения на количество объектов в цепи поставок определение его оптимального значения представляет большую сложность. Также с другой стороны, при росте числа объектов в цепи поставок количество допустимых сбыта продукции далее по цепочке возрастает нелинейно. Ввиду большой вычислительной сложности и нелинейности целевой функции, классическая итерационная процедура оказывается непривлекательной с точки зрения требуемого количества расчетов. Поэтому к задаче нелинейного смешанного целочисленного программирования применяется один из алгоритмов линеаризации (linear programming relaxation) и в дальнейшем решения ищутся точными методами, либо используются приближенные алгоритмы (greedy bump-shift algorithm, карты Кохонена, генетические алгоритмы и прочие).

Модели смешанного целочисленного программирования, также называемые моделями смешанной дискретной оптимизации, позволяют найти решение задачи на заданном дискретном множестве потенциальных местоположений производственных и логистических объектов. Иными словами, задачи смешанного целочисленного программирования - основной инструмент оптимизации цепей поставок (и решения задач по формированию оптимальной структуры цепи), когда известны потенциальные местоположения мощностей, а также определены связанные затраты.

4. Задачи размещения элементов цепи поставок

Простейшая задача размещения

Рассмотрим некоторые постановки задачи размещения. Начнем с простейшей задачи размещения. Пусть множество $I = \{1, \dots, I\}$ задает потенциальные местоположения производственных объектов, которые выпускают одинаковые продукты. Производственное предприятие может быть открыто в любом из заданных объектов $i \in I$. Величина $c_i \geq 0$ задает соответствующие затраты на открытие (постоянные затраты). Каждый открытый объект может выпускать неограниченное количество товаров для потребителей.

Будем считать, что множеством $J = \{1, \dots, J\}$ задается перечень клиентов, которым реализуется произведенная продукция. Для каждой пары производственный объект - клиент *ij* заданы производственные, транспортные и иные сопутствующие затраты через $g_i \geq 0$ (переменные затраты). Задача заключается в нахождении подмножества множества потенциальных местоположений производственных объектов для оптимального размещения предприятий $S \subseteq I, S \neq \emptyset$, которые позволяют с минимальными затратами удовлетворить спрос всех клиентов.

$$\min_{S \subseteq I} \left(\sum_{i \in S} C_i + \sum_{j \in 5} \min_{i \in S} g_{ij} \right)$$

Поставленная задача является обобщением задачи покрытия множеств, также является NP-трудной задачей в сильном смысле. Решения данной задачи можно найти как с помощью точных методов, так и приближенных с

гарантированными оценками точности, Лагранжевыми эвристиками, вероятностными итерационными алгоритмами локального поиска.

Многостадийная задача размещения

В многостадийной (многоэтапной) задаче о размещении объектов цепей поставок предполагается, что производственный цикл товара происходит последовательно на различных объектах. Помимо производственных объектов и клиентов задается также множество допустимых производственных путей, которые определяются иерархией системы производства. Задается стоимость открытия предприятий, производственные и транспортные расходы для каждой цепочки производства. Задача заключается в том, чтобы определить набор производственных объектов чтобы с минимальными затратами удовлетворить спрос клиентов.

На рисунке 1 представлен пример иерархической структуры цепочки производства и выделено возможное решение задачи.

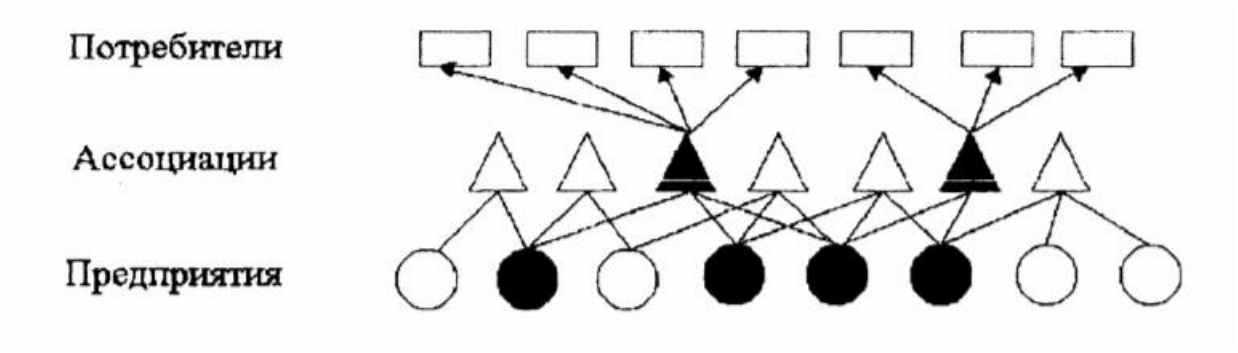

Рисунок 1. Схема размещения предприятий и производственных цепочек

Математическая постановка может быть записана в терминах целочисленного программирования. Пусть множество $I = \{1, ..., I\}$ задает местоположения производственных объектов. $J = \{1, ..., J\}$ — множество клиентов которым реализуется произведенная продукция. Также $L = \{1, ..., L\}$ будем считать множеством производственных путей. При этом $L_i \subset L$ — множество путей, в которые входит i-е предприятие.

$c_i \geq 0$ — затраты на открытие предприятия i;

$g_{ij} \geq 0$ — затраты на удовлетворения спроса j-го клиента i-й производственным путем (транспортные/производственные затраты).

$$y_i = \begin{cases} 1, & \text{если открывается предприятие } i \\ 0 & \text{в ином случае} \end{cases}$$

$$x_{lj} = \begin{cases} 1, & \text{если спрос } j\text{–го клиента удовлетворяется } i\text{–ым производственным путем} \\ 0 & \text{в ином случае} \end{cases}$$

Многостадийная задача размещения может быть записана следующим образом [13, 14]:

$$min_{x,y}\{\sum_{i \in I} c_i y_i + \sum_{j \in J} \sum_{l \in L} g_{lj} x_{lj}\}$$

$$\sum_{i \in I} x_{ij} = 1,\ j \in J,$$

$$\sum_{l \in L} x_{lj} \leq y_i,\ i \in I,\ j \in J,$$

$$x_{lj}, y_i \in \{0, 1\},\ i \in I,\ j \in J,\ l \in L.$$

Целевая функция выражает величину общих постоянных (стоимость открытия) затрат и переменных (транспортные, производственные) затрат. Последующее ограничение гарантирует удовлетворение спроса всех клиентов.

Последнее неравенство определяет затраты на открытие предприятий в случае, если они участвуют в удовлетворении спроса.

Описанная задача относится к числу NP-трудных в сильном смысле задач дискретной оптимизации. Частным её случаем является простейшая задача размещения. Также двухуровневая задача расположения [10], задача с ограничениями на мощности предприятия могут быть представлены в виде многостадийной задачи размещения.

В работе [14] приходят к выводам, что вероятностные методы решения задачи, несмотря на то, что являются более трудоемкими, показывают более лучшие результаты, чем их детерминированные аналоги.

Задача Вебера

Согласно [39] классическая задача Вебера или обобщенная задача Ферма-Торричелли задается как задача нахождения точки (объекта) $W = (x_*, y_*)$, которая минимизирует сумму взвешенных расстояний от себя до $n \geq 3$ неподвижных точек (терминалов) $\{P_j = (x_j, y_j)\}_{j=1}^{n}$ на евклидовой плоскости:

$$\min_{W \subset R^2} \sum_{j=1}^{n} m_j |WP_j|$$

Здесь и далее веса $\{m_j\}_{j=1}^{n}$ предполагаются положительными действительными числами. Модулем обозначается Евклидово расстояние.

Классическая задача Вебера может быть решена с помощью модифицированного алгоритма Вайцфельда.

Рассмотрение проблемы в случае терминалов с $n = 3$ было впервые предпринято в 1872 году Лонхардтом , интерес которого возник из очевидной связи с проблемой экономической географии, известной в настоящее время как оптимальное расположение объекта. В 1909 году Альфред Вебер предложил другую экономическую интерпретацию проблемы с тремя терминалами и сформулировал расширение на случай с четырьмя терминалами. Математически поставленная задача может быть сформулирована как задача нахождения точек $W_1 = (x_*, y_*)$ и $W_2 = (x_{**}, y_{**})$ исходя из:

$$\min_{[w_1, w_l] \subset R^2} F(W_1, W_2)$$

$$F(W_1, W_2) = m_1|W_1P_1| + m_2|W_1P_2| + m_3|W_2P_3| + m_4|W_2P_3| + m|W_1P_2|$$

Общая мультифакторная задача Вебера задаётся как задача расположения заданного числа $l \geq 2$ объектов $\{W_i\}_{i=1}^{l}$ в R^d, связанных с терминалами $\{P_j\}_{j=1}^{n} \subset R^d$

$$\min_{[w_1,\ldots,w_l] \subset R^d} \left\{ \sum_{j=1}^{n} \sum_{i=1}^{l} m_{ij} |W_i P_j| + \sum_{k=1}^{l} \sum_{i=k+1}^{l-1} \tilde{m}_{ik} |W_i W_k| \right\}$$

В данной постановке веса некоторые веса m_{ij}, $\tilde{m}_{ik}$ могут иметь нулевые значения. Эта задача может рассматриваться как естественное обобщение задачи минимального дерева Штейнера, направленной на построение сети минимальной длины, связывающей терминалы.

Задача конкурентного размещения

Рассмотрим математическую модель [30] конкурентной борьбы на рынке двух производителей сходной продукции – Лидера и Последователя – при их последовательном вхождении на рынок. Принятие решений участниками рынка происходит поэтапно.

- Лидер выбирает точки размещения для своих предприятий каких местах разместить свои предприятия;
- Последователь с учетом размещения Лидера, размещает свои предприятия;
- каждый клиент выбирает для себя наиболее предпочтительное предприятие, которое и приносит доход его обладателю (Лидеру или Последователю).

Цель Лидера максимизация прибыли с помощью выбора мест размещения предприятий в условиях, когда Последователь преследует такую же цель. Прибыль складывается из дохода, получаемого от обслуживания (реализации товаров) потребителей за вычетом стоимости открытых ими предприятий. Предпочтения потребителей полагаются известными.

Пусть множество $I = \{1, \dots, m\}$ задает потенциальные местоположения производственных объектов, которые выпускают одинаковые продукты. Производственное предприятие может быть открыто в любом из заданных объектов $i \in I$.

Множеством $J = \{1, \dots, n\}$ задается перечень клиентов, которым реализуется произведенная продукция.

Будем говорить, что для всякого $i \in I$ заданы величины f_i и g_i - фиксированные (постоянные) затраты на открытие i-го предприятия Лидером и Последователем соответственно.

Для i ∈ I и j ∈ J через p_{ij} обозначим величину дохода, получаемого предприятием i при обслуживании потребителя j.

Считаем, что для всякого j ∈ J на множестве I задано отношение порядка $\succ_j$, показывающее предпочтения потребителя j при выборе им предприятия. Отношение $i \succ_j k$ для i, k ∈ I означает, что из двух открытых предприятий i и k потребитель j выберет предприятие i.

Для формальной записи задачи используем следующие переменные:

$$x_i = \begin{cases} 1, & \text{если Лидер открывает предприятие } i \in I \\ 0 & \text{в противном случае} \end{cases}$$

$$x_{lj} = \begin{cases} 1, & \text{если предприятие } i \in I \text{ является наилучшим для потребителя} \\ & j \in J \text{ среди всех предприятий,открытых Лидером} \\ 0 & \text{в противном случае} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{если Последователь открывает предприятие } i \in I \\ 0 & \text{в противном случае} \end{cases}$$

$$z_{lj} = \begin{cases} 1, & \text{если предприятие } i \in I \text{ является наилучшим для потребителя} \\ & j \in J \text{ среди всех предприятий,открытых Лидером и Последователем} \\ 0 & \text{в противном случае} \end{cases}$$

С использованием указанных переменных и введенных обозначений задача конкурентного размещения предприятий записывается в следующем виде:

$$max_{(x_{ij}),(x_{ij})} \left\{ -\sum_{i \in I} f_i x_i + \sum_{j \in J} \left(\sum_{l \in I} p_{ij} x_{ij} \right) \left(1 - \sum_{l \in I} \tilde{z}_{ij} \right); \right.$$

$$x_i + \sum_{k | i \succ_j k} \tilde{x}_{kj} \leq 1, , i \in I, j \in J;$$

$$x_i \geq x_{ij}, i \in I, j \in J;$$

$$x_i , x_{ij} \in (0,1), i \in I, j \in J;$$

$$((\tilde{z}_i),(\tilde{z}_{ij})) - \text{оптимальное решение задачи}$$

$$max_{(\tilde{z}_i),(\tilde{z}_{ij})}\{-\sum_{\mathrm{i}\in\mathrm{I}} g_{\mathrm{i}}\,\mathrm{z}_{\mathrm{i}} + \sum_{\mathrm{i}\in\mathrm{I}}\sum_{\mathrm{j}\in\mathrm{J}} p_{\mathrm{ij}}\mathrm{z}_{\mathrm{ij}});$$

$$\mathrm{x}_{\mathrm{i}} + \mathrm{z}_{\mathrm{i}} + \sum_{\mathrm{k}|\mathrm{i}\succ_j\mathrm{k}} \tilde{z}_{kj} \leq 1,, i \in \mathrm{I}, j \in \mathrm{J};$$

$$\mathrm{z}_{\mathrm{i}} \geq \mathrm{z}_{\mathrm{ij}}, i \in \mathrm{I}, j \in \mathrm{J};$$

$$\mathrm{z}_{\mathrm{i}}\,, \mathrm{z}_{\mathrm{ij}} \in (0,1), i \in \mathrm{I}, j \in \mathrm{J}.$$

Целевая функция сформулированной задачи выражает величину прибыли, получаемой Лидером с учетом потери части потребителей, «захваченных» Последователем.

Последующее неравенство реализует правило выбора потребителем наиболее предпочтительного предприятия среди всех предприятий, открытых Лидером. Это же неравенство гарантирует, что каждый потребитель может выбрать для своего обслуживания не более одного открытого предприятия.

Ограничение, следующее за неравенством, означает, что потребитель для своего обслуживания может выбрать только открытое предприятие.

Аналогичный смысл имеют целевая функция и ограничения задачи записанной во второй части. Целевая функция определяет прибыль, Последователя, а с помощью неравенство выполняется правило выбора клиентом самого привлекательного для него предприятия среди всех предприятий, открытых как Лидером, так и Последователем. Помимо этого, ограничение показывает, что если предприятие открыто Лидером, то оно не может быть открыто Последователем.

Задача о р – медиане

Задача p-медианы отличается от простейшей задачи размещения двумя аспектами: отсутствие затрат на открытие объектов и существование верхней границы количества объектов, которые должны быть открыты. Она моделирует задачу поиска кластеризации с минимальной стоимостью и в строгом смысле относится к классу NP-сложных задач. Также задачу p – медианы можно свети к задаче о вершинном покрытии

Пусть I = {1, ..., n} - набор потенциальных местоположений для p предприятий;

J = {1, ..., m} - множество клиентов;

p – количество размещаемых предприятий;

z_i – матрица распределения, где

$$z_i = \begin{cases} 1, & \text{если точка } i \in I \text{ размещения предприятия используется} \\ 0, & \text{если точка } i \in I \text{ размещения предприятия не используется} \end{cases}$$

Известна матрица связи клиентов с предприятиями (g_{ij}) – n x m которая является матрицей транспортных затрат для удовлетворения i – го клиента j – ым предприятием.

Ниже приводится формальная запись задачи в виде целочисленного линейного программирования:

$$F(z) = \sum_{i \in I} \sum_{j \in J} g_{ij} z_i \rightarrow min$$

При условиях:

$$\sum_{i \in \mathrm{I}} z_{\mathrm{i}} = p$$

Задача состоит в том, чтобы разместить не более p объектов в точках I, чтобы минимизировать общие транспортные расходы для удовлетворения спроса клиентов. Потребности каждого клиента удовлетворяются ближайшим открытым объектом.

5. Практический эксперимент

По приведенным ранее задачам дискретной оптимизации существуют как точные, так и приближенные методы их решения. Среди точных методов наиболее распространены метод ветвей и границ и метод Лагранжевых релаксаций. Однако точные методы в некоторых случаях немногим лучше приближенных методов и более того реализуются за существенно большее время. Среди приближенных методов наибольшую эффективность демонстрируют вероятностные мета эвристики: Поиск с запретами, Имитация отжига, Генетический алгоритм и Алгоритм муравьиной колонии. Рассмотрим последний из приведенных алгоритмов для решения задачи о p – медиане.

Алгоритм муравьиной колонии

Алгоритм муравьиной колонии (Ant Colony) появился благодаря исследованию поведения живых муравьев при поиске кратчайшего пути между источником пищи и их домом. Было обнаружено, что при движении муравей производит вещество, называемое феромоном, которое остается в виде следа по пути движения муравья. След феромона используется другими

членами колонии в поисках источника пищи, строительных материалов и поиска обратного пути до дома. Причем вероятность выбора муравьем пути увеличивается при большей концентрации феромона. Большая концентрация феромона на пути достигается большим количеством прохождения этого пути другими муравьями: чем выше уровень феромона, тем вероятнее направление. Поскольку через определенное время кратчайший путь будет иметь наибольшее количество феромонов, он становится наиболее вероятным. Кроме того, так как феромон со временем испаряется, кратчайший путь (который занимает меньшее время на прохождение) дольше будет содержать высокую концентрацию феромонов. Такое поведение муравьев можно интерпретировать как процесс оптимизации. Феромон содержит информацию о качестве маршрута и используется как метод передачи данных.

Модель муравьиной колонии заключается в следующих наблюдениях за муравьями в живой природе:

- первый муравей покидает муравейник для поиска источников пропитания или строительных материалов;
- по пути своего движения туда и обратно он оставляет след из феромонов;
- муравьев, находящихся поблизости, привлекает запах феромонов, и они присоединяются к маршруту первого муравья, увеличивая приоритет этого пути, добавляя своих феромонов;
- из нескольких маршрутов, по более короткому успеют пройти больше муравьев за одинаковый промежуток времени;
- в связи с этим у короткого маршрута повышается привлекательность для других муравьев;

- феромоны находящиеся на более длинных путях с течением времени испаряться.

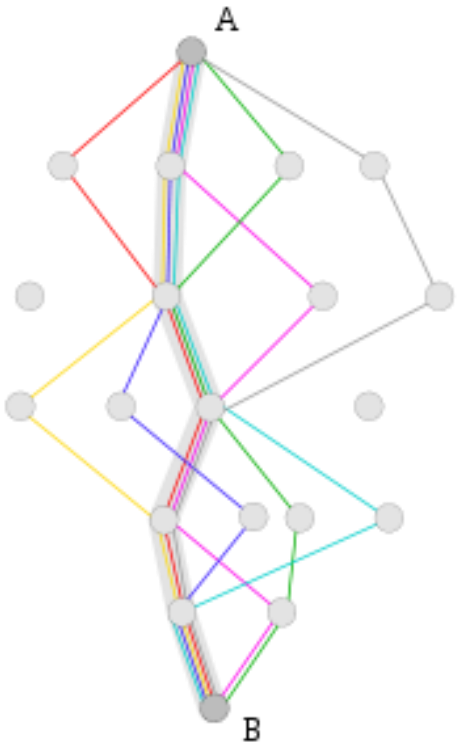

Рисунок 2. Поиск муравьями кратчайшего пути

Алгоритм муравьиной колонии на каждой итерации использует искусственного муравья (их конечное количество) для поиска решения задачи, таким образом искусственный муравей является вероятностной модификацией алгоритма жадного спуска, который на каждом шаге строит решения. При построении таких решений по каждой итерации данные накапливаются и обрабатываются в алгоритме муравьиной колонии. Они влияют на дальнейшие поиски и могут быть интерпретированы как аналог феромона живых муравьев. Таким образом, поведение некоторых муравьев моделируется алгоритмом муравьиной колонии. Использование

статистической информации, накопленной искусственными муравьями, является стержнем алгоритма. Критерий остановки может быть определен количеством итераций, временем вычисления алгоритма и др.

Опишем схему применения алгоритма муравьиной колонии для задачи p-медианы (алгоритм хорошо применим и для прочих задач размещения). I – множество предприятий; $\hat{I}$ – множество открытых предприятий; p – количество отбираемых лучших решений; вектор z будет называться решением задачи p-медианы. Дополним также модель постоянными затратами на открытие объектов $C_i \geq 0$.

$$F(z) = \sum_{i \in I} \sum_{j \in J} g_{ij} z_i + \sum_{i \in I} C_i z_i \rightarrow min$$

При условиях:

$$\sum_{i \in I} z_i = p$$

Статистическая информация α, которая накапливается и хранится в векторе $\alpha_k = \alpha_{ik}, i \in I$ служит феромоном для каждого i-го предприятия на k-ой итерации алгоритма муравьиной колонии.

Δf_{ik} – изменение целевой функции при закрытии предприятия i на k-ом шаге алгоритма искусственного муравья. $\Delta f_{ik} \geq 0 \; \forall \; i \in \hat{I}$ так как при закрытии предприятия значение целевой функции f не убывает.

1. Определить начальный вектор статистической информации (уровень феромонов) α, начальный рекорд F= ∞. Шаг k, k > 1.
2. Построить допустимые решения алгоритмом искусственного муравья:
   a. $\hat{I} = I$;

b. Если | $\hat{I}$ | = p, то End;

c. Сгенерировать множество W (λ)

$$W(\lambda) = \left\{ i \in \hat{I} \middle| \Delta fi \leq (1 - \lambda) \min \Delta f_i + \lambda \max \Delta f_i \right\}$$

Где $\lambda \in [0,1]$ - параметр алгоритма, детерминировано генерируется на каждой итерации алгоритма муравьиной колонии.

d. Выбрать элемент $i_0 \in W(\lambda)$ с вероятностью p_i:

$$p_i = \frac{\alpha_i(\Delta fmax \quad -\Delta \quad f_i \quad + \varepsilon)}{\sum_{k \in w} \alpha_k(\Delta f \max - \Delta f_k + \varepsilon)}, \qquad i \in w(\lambda)$$

Где параметр $\varepsilon > 0$ необходим чтобы любое предприятие имело возможность быть закрытым.

e. Переопределить $\hat{I} = \hat{I} \{i_0\}$.

f. Следующая итерация, k = k + 1.

3. Выбрать среди них t лучших решений по целевой функции f^* - рекорд итерации.

4. Найти вектор накопления статистической информации (феромонов) $\alpha_{i,k+1}$, i ∈ I. Компоненты вектора $\alpha_{i,k+1}$ модифицируются следующим образом:

$$\alpha_{i,k+1} = \frac{\alpha min + q^{\gamma_i}(\alpha_{ik} - \alpha min)}{\beta_k}, i \in I$$

Где β_k – коэффициент испарения феромона;

γ_i – частота появления предприятия i в t лучших решениях;

q – параметр принадлежащий [0, 1].

Следовательно, при заданных значениях параметров β и q, чем чаще предприятие оказывается среди t лучших решений в смысле целевой функции, тем меньше соответствующее значение α_i, i ∈ I.

5. Если $f^* < F$, то для ненулевых компонент соответствующего z^*: $\alpha_i = \alpha_{min}$ (параметр минимального значения α_i, I ∈ I. $F = f^*$.
6. Если критерий остановки выполняется, то End.

В качестве критериев остановки можно использовать предельное количество итераций, точность заданной нижней границы целевой функции или повторение агентами тех же решений.

7. Перейти к следующей итерации, k = k + 1.

Следует отметить, что наличие нескольких параметров порождает так называемую проблему настройки алгоритма, то есть выбор таких параметров, которые гарантируют хорошее поведение алгоритма на большинстве задач.

Программная реализация

В приложении приведен разработанный в рамках данной работы программный код на языке Python реализующий описанный и формализованный ранее алгоритм. Код написан без применения сторонних библиотек.

Был произведен ряд вычислительных экспериментов по настройке алгоритма для определения параметров, гарантирующих хорошее поведение

алгоритма на большинстве задач. Для этого на вход алгоритма подавались случайным образом сгенерированные матрицы различных размерностей.

В примере ниже (Таблица 1) наблюдение производилось за временем с которым алгоритм искал оптимальное решение в зависимости от различных значений коэффициента испарения феромона.

	β = 0,3	β = 0,5	β = 0,8	β = 0,9	β = 0,95	β = 1
1	213,31	187,83	163,41	132,95	112,41	137,58
2	207,17	178,11	154,20	136,69	115,53	135,86
3	205,78	174,75	158,48	137,71	117,64	142,79
4	204,32	182,26	162,34	131,66	112,50	130,42
5	195,23	190,87	160,47	134,50	112,95	142,54
6	205,32	192,53	162,27	126,28	116,89	143,71
7	193,56	176,78	160,54	132,43	118,32	138,47
8	197,55	194,18	153,51	135,27	114,32	130,77
9	194,54	181,90	164,15	135,28	112,52	131,61
10	206,46	190,57	165,86	125,58	118,73	134,24
Среднее	**202,32**	**184,98**	**160,52**	**132,84**	**115,18**	**136,80**

Таблица 1. Зависимость времени поиска решения от коэффициента испарения феромона

По результатам эксперимента на анализ чувствительности коэффициента испарения феромона для задачи размерностью 30 x 30 видно, что алгоритм показывает лучшее время работы при значении β = 0,95.

Вместе с тем была получена и зависимость времени работы алгоритма от количества итераций (Таблица 2).

	Размерность задачи				
Количество итераций	10x10	15x15	20x20	25x25	30x30
1	0,1338	0,9449	2,8048	5,3176	11,6199

2	0,3180	1,8983	6,3696	11,5116	24,9411
3	0,4787	2,8853	9,6564	17,6186	38,2909
4	0,6743	3,8443	12,8759	23,4735	51,0739
5	0,8757	4,8253	16,1642	29,1553	64,3394
6	1,0355	5,7789	19,7312	35,0683	76,9322
7	1,2011	6,7851	23,2039	40,6227	89,8454
8	1,3677	7,7634	26,6639	46,2757	102,9510
9	1,5556	8,7025	30,0400	51,9006	116,3778
10	1,7395	9,6824	33,3571	57,6596	129,4946
11	1,9408	10,6698	37,0047	63,3735	142,8020
12	2,1242	11,5960	40,3603	69,0136	155,5289
13	2,3155	12,5592	43,5260	74,7408	169,4668
14	2,4849	13,6036	47,0659	80,3027	182,2090
15	2,6552	14,5618	50,6101	86,3024	195,0345
16	2,8489	15,5085	53,7638	92,1759	207,5543
17	3,0187	16,4584	56,6411	97,9065	220,6801
18	3,1830	17,4610	59,4950	104,0882	233,6473
19	3,3726	18,3945	62,3072	109,4882	246,3449
20	3,5694	19,3192	65,4072	115,2489	258,9505
21	3,7812	20,2676	68,0749	120,8753	271,9334
22	3,9758	21,1982	71,0279	126,5433	284,9736
23	4,1586	22,1150	73,8157	132,1033	297,2285
24	4,3571	23,0712	76,7098	138,0892	310,7288
25	4,5372	24,0205	79,3357	143,9003	323,5461
26	4,7317	24,9422	82,1653	149,5180	336,2374
27	4,9154	25,8737	84,9585	155,2488	349,0374
28	5,0904	26,7394	87,5573	160,9867	361,5624
29	5,2723	27,6899	90,3050	166,9542	374,3760
30	5,4799	28,6238	93,0114	172,8246	388,3142
31	5,6579	29,6186	95,8769	178,4353	401,6464
32	5,8449	30,5649	98,5623	184,3071	414,6999
33	6,0221	31,4671	101,2240	190,2677	426,9235
34	6,2087	32,2764	104,0430	195,8244	440,0740
35	6,3969	33,2286	106,7466	201,6023	453,4710
36	6,6199	34,1259	109,4381	207,3275	466,6160
37	6,8223	35,0356	112,2100	212,8948	479,3160
38	7,0081	35,9149	115,1454	218,7195	492,3817
39	7,1621	36,8217	118,0976	224,4008	504,9371

40	7,3093	37,7838	120,9621	230,3010	518,4842

Таблица 2. Зависимость времени работы алгоритма от количества итераций.

Также был проведен вычислительный эксперимент, исследующий степень улучшения решения в зависимости от количества итераций и количества муравьев. Поскольку каждый муравей в алгоритме муравьиной колонии является вероятностным жадным алгоритмом, то при малом их количестве может быть найдено неоптимальное решение. Эксперимент проводился для нескольких задач одинаковой размерности. В таблице указаны средние значения.

	10 муравьев	20 муравьев	30 муравьев	40 муравьев	50 муравьев
10	864,7841	854,1308	854,0306	852,1502	849,3573
20	860,58	855,4024	853,2504	846,5348	843,3337
30	846,4982	844,7863	846,3731	841,7648	824,7844
40	841,1924	840,0439	830,8657	827,2101	814,3598
50	838,5238	832,2139	820,7518	818,2958	792,2995
60	834,4536	824,3808	805,153	792,3508	789,3947
70	833,3145	817,2893	795,1822	789,5307	791,7588
80	821,4813	793,4665	790,159	787,0202	790,0954
90	818,535	796,6443	787,2599	788,0661	784,0566
100	816,4448	791,7299	789,4459	785,8372	785,0657
110	797,6011	788,1361	786,3792	784,2585	780,1074
120	799,3504	788,7129	789,4841	785,5541	779,5203
130	797,0029	789,4437	789,0724	782,5481	781,0621
140	795,757	784,2401	788,1703	781,1605	775,0764
150	794,2218	786,6875	783,7745	777,4586	774,4961
160	798,7838	786,6061	777,7358	777,4975	775,7467
170	796,33	779,2153	774,2344	777,1675	775,8265
180	794,2349	778,5769	774,1459	775,4502	773,5563
190	789,2908	777,4313	775,1566	777,1946	773,1905
200	788,5313	777,3867	778,0209	777,0358	776,6366

Таблица 3. Зависимость решения от количества итераций и количества муравьев

В виде графика (Рисунок 3):

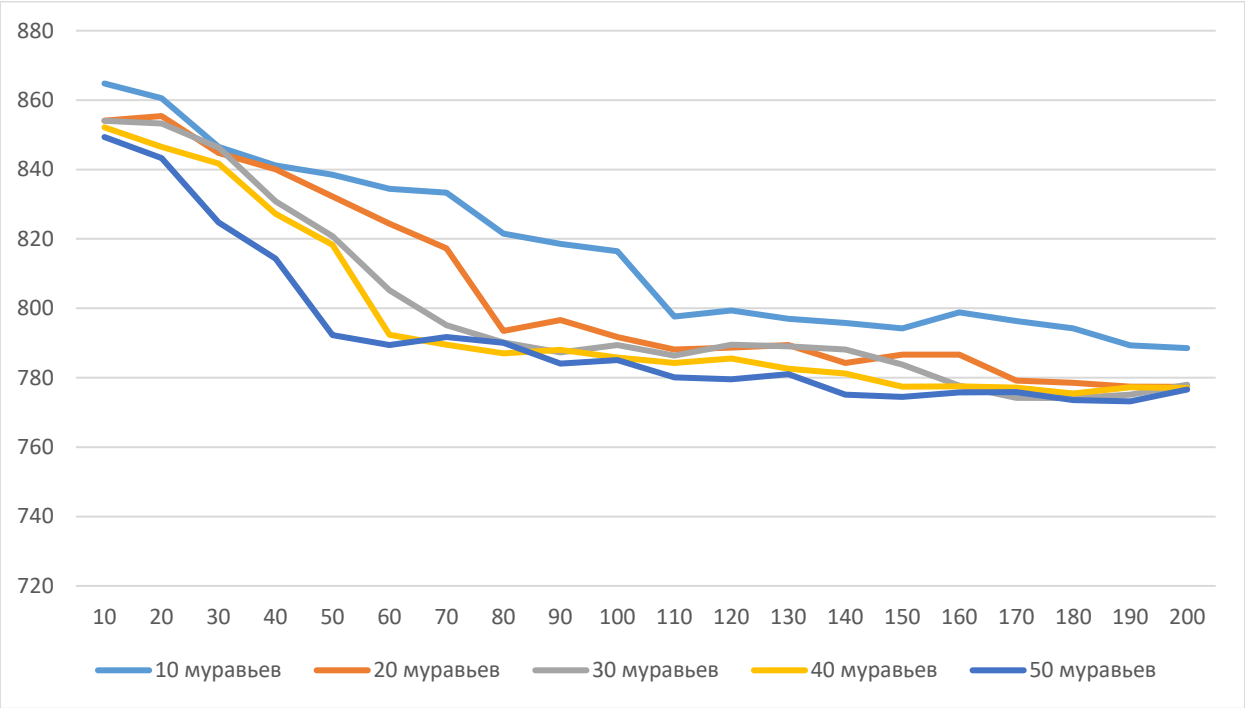

Рисунок 3. Зависимость решения от количества итераций и количества муравьев

Исходя из графика можно сделать вывод, что количество итераций в большей степени влияет на нахождение оптимального решения, чем количество муравьев в алгоритме. В данном случае имеет смысл использовать 30 муравьев для 160 итераций, что позволяет найти баланс между точностью найденного решения и времени работы алгоритма. Однако с увеличением размерности задач эти значения могут меняться.

Также проведен ещё один вычислительный эксперимент для сравнения решений найденных алгоритмом муравьиной колонии и точных решений найденных методом ветвей и границ.

Метод ветвей и границ был реализован в программном пакете Matlab. На вход были сгенерированы матрицы размерностей 10 x 10, 20 x 20, 30 x 30, 40

x 40, …, 100 x 100. Получены значения ошибок приближенного алгоритма в сравнении с точным. Результаты представлены в Таблице 4.

	Задача 1	Задача 2	Задача 3	Задача 4	Задача 5	% отклонения
ММК 10x10	1108	735	912	869	736	0,00%
МВиГ 10x10	1108	735	912	869	736	
ММК 20x20	756	994	914	1081	830	0,00%
МВиГ 20x20	756	994	914	1081	830	
ММК 30x30	784	954	737	767	747	0,00%
МВиГ 30x30	784	954	737	767	747	
ММК 40x40	766	915	941	983	1117	0,00%
МВиГ 40x40	766	915	941	983	1117	
ММК 50x50	922	710	942	926	976	0,20%
МВиГ 50x50	922	708	939	926	972	
ММК 60x60	872	823	969	1024	1060	0,13%
МВиГ 60x60	871	821	967	1024	1059	
ММК 70x70	927	1072	877	984	1062	0,30%
МВиГ 70x70	925	1068	871	983	1060	
ММК 80x80	984	1058	925	1090	968	0,80%
МВиГ 80x80	982	1047	913	1082	961	

ММК 90x90	1015	945	1121	893	1130	1,43%
МВиГ 90x90	999	923	1113	879	1117	
ММК 100x100	1029	1039	1083	960	890	2,30%
МВиГ 100x100	1001	1020	1065	929	871	

Таблица 4. Сравнение решений ММК и МВиГ

На графике (Рисунок 4) показана зависимость % отклонения решения приближенного алгоритма от решения точным алгоритмом.

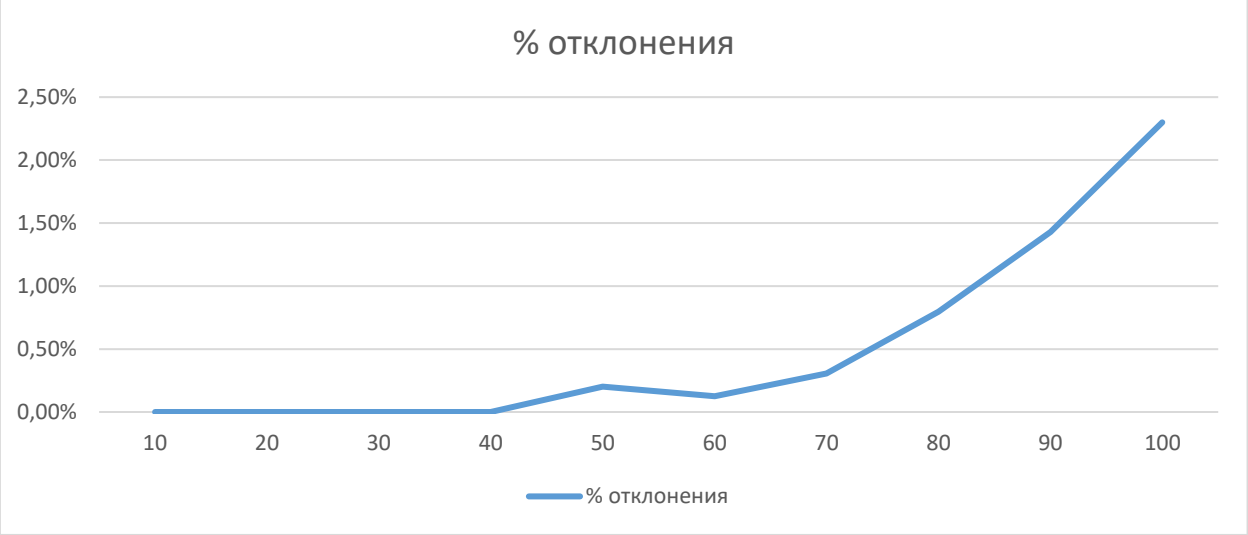

Рисунок 4. Отклонения решений ММК от МВиГ в зависимости от размерности задачи

Анализируя зависимость можно сделать вывод, что при увеличении размерности матрицы точность приближенного алгоритма становится хуже. Однако для матриц с размерностью до 40x40 решения является точными. Также стоит заметить что максимальное отклонение в 2,3% от решений точного алгоритма говорит о хороших результатах, которые показывает

алгоритм муравьиной колонии.

При большой размерности задач точные методы не могут получить оптимальное решение за приемлемое время.

По итогу разработанный алгоритм был применен для решения реальной бизнес-задачи на реальных данных компании работодателя для получения оценки по выбору оптимальных мест расположения элементов цепи поставок с минимизацией сопутствующих издержек, что позволило получить дополнительную информацию по принятию соответствующих решений при выходе на новые рынки присутствия.

6. Заключение

В ходе проведения научно-исследовательской работы, все поставленные цели были достигнуты.

Проведено исследование различных расширений и усложнений моделей размещения элементов цепи поставок и методов их решения (ограничение мощности производства, в условиях ценовой конкуренции, с предписанным/свободным выбором поставщиков и проч.). Точные методы решения и приближенные.

Выявлены факторы модели, проведен сбор, обработка и анализ исходных данных (определены различные варианты размещения производств, складов, распределительных центров, их мощности; стоимость их открытия; определены потребители продукции, проведена оценка спроса и его динамика; определены логистические издержки; определены операционные издержки) для проведения практического эксперимента на основе исследуемых методов.

Разработан программный код, реализующий приближенный алгоритм муравьиной колонии (Ant Colony) для нахождения оптимального решения задачи размещения элементов цепи поставок. Проведен анализ чувствительности параметров, тесты на производительность кода, сравнение полученных результатов с точными методами. Проведены вычислительные эксперименты на реальных данных, что позволило решить реальную прикладную задачу компании по расположению элементов цепи поставки при выходе на новые рынки присутствия.

7. Список литературы

1. Береснев В.Л., Гимади Э.Х., Дементьев В.Т. Экстремальные задачи стандартизации. Новосибирск: Наука, 1978.
2. Гимади Э.Х. Выбор оптимальных шкал в одном классе задач типа размещения, унификации и стандартизации. Новосибирск. Управляемые системы. Вып. 6, 1970
3. Береснев В.Л. Об одном классе задач оптимизации параметров однородной технической системы. Управляемые системы. Вып. 9 (1971), Новосибирск, Ин-т математики Сиб. отд. АН СССР, с. 65-74.
4. Cornuejols G., Fisher M.L., Nemhauser G.L. Location of bank accounts to optimize float. Management Science. v22 (1977), pp 789-810.
5. Khumawala B.M. An Efficient Branch-Bound Algorithm for the Warehouse Location Problem. Management Science. v18 (1972), pp 718-731.
6. Krarup J., Pruzan P.M. The simple plant location problem: Survey and synthesis. European Journal of Operational Research. v12 (1983), pp 36-81.
7. Mirchandani P.B., Francis R.L. Discrete Location Theory. John Wiley & Sons, 1990.
8. Diaz J.A., Fernandez E. Column generation for the single source capacitated plant location problem. Technical report DR 2000/17, UPC Barcelona, 2000.
9. Sridharan R. The capacitated plant location problem. European Journal of Operational Research. v87 (1995), pp 203-213.
10. Горбачевская Л.Е., Дементьев В.Т., Шамардин Ю.В. Двухуровневая задача стандартизации с условием единственности оптимального потребительского выбора. Дискретный анализ и исследование операций. Сер.2, т6 (1999), №2, с.3-11.
11. Береснев В.Л. О задаче выбора оптимальных рядов изделий и комплектующих узлов. Управляемые системы. Вып.16 (1977), с. 35-46.
12. Береснев В.Л. Алгоритмы минимизации полиномов от булевых переменных. Проблемы кибернетики. Вып.36 (1979), с. 225-246.

13. Береснев В.Л., Гончаров Е.Н. Приближенный алгоритм для задачи минимизации полиномов от булевых переменных. Дискретный анализ и исследование операций. Сер.2, т5 (1998), №2, с. 3-19.

14. Гончаров Е.Н., Кочетов Ю.А. Поведение вероятностных жадных алгоритмов для многостадийной задачи размещения. Дискретный анализ и исследование операций. Сер.2, т6 (1999), №1, с.12-32.

15. S. Finkelstein, M. Schkolnik, and P. Tiberio Physical database design for relational databases. ACM Transactions on Database Systems, v.13 (1988), pp.91-128.

16. Hotelling H. Stability in competition // Econom. J. 1929. V. 39. P. 41–57.

17. Eiselt H.A., Laporte G. Sequential location problems // Eur. J. Oper. Res. 1996. V. 96. P. 217–242.

18. Eiselt H.A., Laporte G., Thisse J.-F. Competitive location models: a framework and bibliography // Transportat. Sci. 1993. V. 27. P. 44–54.

19. Hay D.A. Sequential entry and entry-deterring strategies in spatial competition // Oxford Econom. Papers. 1976. V. 28. P. 240–257.

20. Prescott E.C., Vissher M. Sequential location among firms with foresight // Bell J. Econom. Papers. 1977. V. 8. P. 378–393.

21. von Stackelberg H. Marktform und Gleichgewicht. Vienna: Springer, 1934.

22. Kress D., Pesch E. Sequential competitive location on networks // Eur. J. Oper. Res. 2012. V. 217. P. 483–499.

23. Garcia M.D., Fernandez P., Pelegrin B. On price competition in location-price models with spatially separated markets // TOP. 2004. V. 12. P. 351–374.

24. Hanjoul P., Hansen P., Peeters D., Thisse J-F. Uncapacitated plant location under alternative spatial price policies // Market Sci. 1990. V. 36. P. 41–57.

25. Панин А., Плясунов А. Задача ценообразования. Ч. 1. Точные и приближенные алгоритмы решения // Дискрет. анализ и исслед. операций. 2012. Т. 19. № 5. С. 83–100. Panin A., Plyasunov A. The pricing problem. Part I: Exact and approximate algorithms // J. Appl. Indust. Math. 2013. V. 7. No. 2. P. 241–251.

26. Панин А., Плясунов А. Задача ценообразования. Ч. 2. Вычислительная сложность // Дискрет. анализ и исслед. операций. 2012. Т. 19. № 6. С. 56–71. Panin A., Plyasunov A. The pricing problem. Part II: Computational complexity // J. Appl. Indust. Math. 2013. V. 7. No. 3. P. 420–430.

27. А. А. Панин, М. Г. Пащенко, А. В. Плясунов, Двухуровневые модели конкурентного размещения производства и ценообразования, Автомат. и телемех., 2014, выпуск 4, 153–169

28. Davydov I., Kochetov Yu., Plyasunov A. On the complexity of the (r|p)-centroid problem in the plane // TOP 2013. DOI:10.1007/s11750-013-0275-y.

29. Dempe S.J. Foundations of bilevel programming. Dordrecht: Kluwer Academ. Publishers. 2002.

30. В. Л. Береснев, А. А. Мельников Приближенные алгоритмы для задачи конкурентного размещения предприятий, Дискретный анализ и исследование операций, Ноябрь—декабрь 2010. Том 17, № 6. С. 3–19.

31. В. И. Сергеев. Управления цепями поставок. 2014

32. Weber A. Über den Standort der Industrie. Teil I: Reine Theorie des Standorts. J.C.B.Mohr, 1909. Tübingen.

33. Brimberg, J., Chen, R., Chen D.: Accelerating convergence in the Fermat-Weber location problem. Operations Research Letters, 22, 151–157 (1998).

34. Fekete, S.P., Mitchell, J.S.B, Beurer, K.: On the continuous Fermat-Weber problem. Operations Research, 53, 61–76 (2005).

35. Hansen, P., Mladenovi´c, N., Taillard, E.: Heuristic solution of the multisource Weber problem as a image-median problem". Operations Research Letters, 22, 55–62 (1998).

36. Drezner Z., Mehrez A., Wesolowsky, G.O.: The facility location problem with limited distances. Transportation Science, 25, 183–187 (1991).

37. E. Weiszfeld, Sur le point pour lequel la somme des distances de n points donn´es est minimum, Tˆohoku Math. J. 43 (1937), 355–386.

38. А. В. Панюков, Р. Э.Шангин, Точный алгоритм решения дискретной задачи Вебера для k-дерева, 2014

39. Alexei Yu. Uteshev, Elizaveta A. Semenova, Geometry and Analytics of the Multifacility Weber Problem, 2019, Faculty of Applied Mathematics, St. Petersburg State University.

40. V. L. Beresnev, A. A. Melnikov, A cut generation algorithm of finding an optimal solution in a market competition, Journal of Applied and Industrial Mathematics, 2019, 13:2, 194–207.

41. В. Л. Береснев, А. А. Мельников, 2016, Задача конкурентного размещения предприятий с ограниченными объемами производства. Дискретный анализ и исследование операций, 127, 35-50.

42. V. L. Beresnev, On the competitive facility location problem with a free choice of suppliers, 2014, Avtom. Telemekh., No. 4, 94–105.

43. V. L. Beresnev and A. A. Melnikov, The branch-and-bound algorithm for a competitive facility location problem with the prescribed choice of suppliers, 2014, Diskretn. Anal. Issled. Oper., 21, No. 2, 3–23.

8. Приложение

```
import random
from datetime import datetime

# Вводим параметры алгоритма муравьиной колонии
param = {'beta': 0.95, 'iterations': 40, 'nants': 60, 'a_min': 0.3, 'q': 0.5, '_lambda': 0.9, 'np': 4}
a = [1 for x in T]
P = 5
eps = 0.01
stime = datetime.now()
etime = []
rec = 1000000

# Ввод входных данных
input_data = [line.strip() for line in open("in.txt", 'r')]
matrix_index = input_data.index([x for x in input_data if "a" in x][0])
T = [[int(y) for y in x.strip().split()] for x in input_data[matrix_index + 2:] if x]
OC= [x for x in input_data if "b" in x][0] OC= eval(OC[OC.index(":") + 1:])
GT = [x for x in input_data if "c" in x][0] GT = eval(GT[GT.index(":") + 1:])

# Целевая функция
def _F_min(z):
```

```
    total_sum = 0
    for j in range(0, len(T[0])):
        current_min = None
        for i in range(0, len(T)):
            current_way = T[i][j]
            if z[i] and (not current_min or current_way < current_min):
        current_min = current_way
        total_sum += current_min
    return total_sum


# Функция Δf
def _F(z):
    sum_min_all = _F_min(z)
    sum_other = []
    for i in range(0, len(T)): sum_without_i = 0
        for j in range(0, len(T[i])): mins = []
            for k in range(0, len(T)):
            if not i == k and z[k]: mins.append(T[k][j])
        sum_without_i += min(mins) sum_other.append(sum_without_i)
    df_min_sum = min([x - sum_min_all for x in sum_other]) / \ (sum_min_all
    + eps)
    df_max_sum = max([x - sum_min_all for x in sum_other]) / \  sum_min_all
    + eps)
    df_all = [(sum_other[i] - sum_min_all) / (sum_min_all + eps) for i in
    range(0, len(T))]
    return df_min_sum, df_max_sum, df_all
```

```
# Функция искусственного муравья
def _Ant(z):
    opened_count = len([x for x in z if x])
    if opened_count == P:
    return z
    df_min_sum, df_max_sum, df_all = _F(z)
    W = [1 for i in range(0, len(z))]
    for i in range(0, len(T)):
        if z[i] and df_all[i] <= ((1 - param['_lambda']) * df_min_sum +
        param['_lambda'] * df_max_sum):
            W[i] = 0
            if df_all[i] > df_max_sum: df_max_sum = df_all[i]
    for i in range(0, len(T)):
        if not W[i]:
            top_part_p = a[i] * (df_max_sum - df_all[i] + eps)
            bot_part_p = sum(a[i] * (df_max_sum - df_all[i]) + eps for j in
            range(0, len(T)) if not W[j])
            p = top_part_p / bot_part_p
            chance_to_close = random.uniform(0, 1)
            if chance_to_close <= p:
                z[i] = 0
                if len([x for x in z if x]) == P:
                    break
     return z[:]
```

```
# Функция муравьиной колонии
def _P_median():

    global a, rec
    all_best_solutions = []
    for iteration in range(0, param['iterations']): iter_stime = datetime.now()
        print("iter: {0}".format(iteration + 1))
        solutions = []
        for number in range(0, param['nants']): company_states = [1 for i in
range(0, len(T))]
            while True:
                company_states = _Ant(company_states)
                if len([x for x in company_states if x]) <= P: cost =
_F_min(company_states)
                    solutions.append((str(company_states), cost))
                    break
        best_solutions = sorted(solutions, key=lambda x: x[1])[:param['np']]
        for solution in best_solutions:
            if not solution in all_best_solutions:
all_best_solutions.append(solution)
        iter_rec = best_solutions[0][1]
        Y = []
        for i in range(0, len(T)): freqency = 0
            for x in range(0, len(best_solutions)):
```

```
                    _solution = eval(best_solutions[x][0])
                    if _solution[i]: freqency += 1
                Y.append(freqency)
                a = [param['a_min'] + pow(param['q'], Y[i]) * (a[i] -
            param['a_min']) / param['beta']
            for i in range(0, len(a))]
            if iter_rec < ecord: rec = iter_rec best_solution = best_solutions[0]
                a = [param['a_min'] if best_solution[0][i] else a[i] for i
            in range(0, len(a))]
            iter_result_time = (datetime.now() -iter_stime).total_seconds()
            print("times:{0}\n".format(iter_result_time))
            etime.append({ 'iteration': iteration + 1, 'time': iter_result_time
            })
        return all_best_solutions


# Функция нахождения ближайшего пути
def _Find_shortest_way(client, solution):
    all_possible_ways = [T[client][i] for i in range(0, len(T)) if solution[i]]
    shortest_way = min(all_possible_ways)
    return shortest_way


# Main
def main_solution():
```

```
        best_solutions = _P_median()
        solutions_oc= []
        for solution in best_solutions: solution_oc= sum([OC[i] for i in range(0,
        len(OC)) if eval(solution[0])[i]])
            solutions_oc.append(solution_oc)
        solutions_gt_costs = []
        for solution in best_solutions:
            _costs = []
            for i in range(0, len(GT)):
                distance = _Find_shortest_way(i, eval(solution[0]))
                _costs.append(distance * GT[i]) solution_gt_costs =
            sum(_costs) solutions_gt_costs.append(solution_gt_costs)
        best_solutions = sorted(best_solutions, key=lambda x: x[1])[:param['np']]
        return best_solutions, solutions_oc, solu- tions_gt_costs
best_solution, solutions_oc, solutions_gt_costs = main_solution()


# Выводим результаты:
print("\nПустые решения: {0}\n".format(len(best_solution)))
sol_index = 0
for each in best_solution: print("Вектор Z: {0}\nСтоимость решения:
{1}\nСтоимость открытия:{2}\nГрузооборот {3}\n".format(each[0],
each[1],solutions_oc[sol_index],solutions_gt_costs[sol_index]))
        sol_index += 1
end_time = datetime.now()
execution_time = (end_time - stime).total_seconds()
```

```
etime.append({ 'iteration': "all", 'time': execution_time
})
print("times:{0}".format(execution_time))
times_file = open('out.txt', 'w')
times_file.write("\n".join(str(x) for x in etime))
times_file.close()
```