УДК

РАЗРАБОТКА И АНАЛИЗ МЕТОДОВ ОПТИМИЗАЦИИ С ПОМОЩЬЮ МАТЕМАТИЧЕСКОГО АНАЛИЗА

Руденя Д.А. Кужовник В.С., студенты гр.351001

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Баркова Е.А. – канд. физ.-мат. наук

Аннотация. Главной целью данной научной работы является анализ существующих методов оптимизации и разработка дополнений к методам. В данной работе был проанализирован метод Венгерского алгоритма, который является решением задач о назначениях. Было разработано дополнение Венгерского алгоритма, рекурсивно решающая задачу о назначениях. Наше дополнение позволяет программно решить задачу о назначениях с помощью рекурсии.

Ключевые слова. Оптимизация, задача о назначениях, Венгерский алгоритм, рекурсия, методы оптимизации, назначение, соответствие, графы, матрицы.

Оптимиза́ция (от лат. *optimus* — «наилучший») — процесс, имеющий целью направить развитие какого-либо объекта или метода к наиболее лучшему состоянию.

Существуют различные задачи оптимизации, которые решаются различными методами, например, Задача о назначениях, Кластерный анализ, Оптимальное управление, Методы оптимизации функций.

Задача о назначении (Assignment Problem) является задачей, которая заключается в определении наилучшего сочетания между двумя группами объектов, например, работниками и задачами.

В наиболее общей форме задача формулируется следующим образом:

Имеется некоторое i число *работ* и некоторое число i *исполнителей*. Любой исполнитель может быть назначен на выполнение любой (но только одной) работы, но с неодинаковыми затратами. Нужно распределить работы так, чтобы выполнить работы с минимальными затратами или с максимальной эффективностью.

Для решения задачи о назначении существует множество алгоритмов, например, Метод Венгерского алгоритма.

Венгерский алгоритм — алгоритм оптимизации, решающий задачу о назначениях. Он был разработан и опубликован Гарольдом Куном в 1955 году. Алгоритм является строго полиномиальным. С того времени алгоритм известен также как алгоритм Куна — Манкреса или алгоритм Манкреса. Временная сложность оригинального алгоритма была O(n^4), однако со временем его модифицировали до времени выполнения O(n^3). Вскоре этот метод метод на транспортные задачи. Однако в 2006 году было обнаружено, что Якоби нашёл идентичное решение задачи о назначениях в XIX веке.

Алгоритм ищет значения, которые надо вычесть из всех элементов каждой строки и каждого столбца (разные для разных строк и столбцов), такие, что все элементы матрицы останутся неотрицательными, но появится нулевое решение.

В дальнейшем мы под **назначением** понимаем такой набор нулей, каждый из которых будет задавать соответствие работник/работа (каждый такой ноль будет единственным в своей строке и столбце).

Под **соответствием** мы понимаем элемент матрицы который описывает эффективность выполнения работником работы.

Решение будет найдено, если мы сможем выделить комбинацию нулей таких, что в каждой строке и в каждом столбце будет выделен только 1 ноль.

Описание алгоритма

1 Шаг. Для решения задачи с помощью алгоритма мы уменьшаем элементы построчно. Если мы ищем максимальное назначение, то находим наибольший из элементов первой строки, и вычитаем его из всех элементов первой строки. Если мы ищем минимальное назначение, то находим наименьший из элементов первой строки и вычитаем его из элементов этой строки. При этом хотя бы один из элементов первой строки обнулится. То же самое выполняем и для всех остальных строк. Далее возьмем каждый элемент матрицы по модулю. Теперь в каждой строке матрицы есть хотя бы один ноль. Иногда нулей уже достаточно, чтобы найти назначение.

Этим шагом, мы подготавливаем матрицу таким образом, чтобы новые значения на позициях наибольших/наименьших элементов изначальной матрицы были построчно по модулю меньше остальных элементов строки новой матрицы.

2 Шаг. Если назначить нули всё ещё нельзя, мы отнимаем минимальный элемент каждого столбца из этого же столбца и вновь проверяем, возможно ли назначение.

Этим шагом мы просто приводим наименьшие элементы каждого столбца к нулю.

3 Шаг. В случае невозможности назначения нулей после второго шага, мы вычеркиваем из матрицы столбцы и строки с нулями с условием, что количество вычеркиваний будет минимально. Далее из неперечеркнутых элементов мы находим минимальный, отнимаем его от этих же оставшихся элементов и прибавляем к элементам, которые находятся на пересечении вычеркнутых столбцов и строк.

Этим шагом, мы убираем (перечеркиваем) тех работников, которые могут выполнять одинаково хорошо несколько работ, и те работы, которые могут быть выполнены одинаково хорошо несколькими работниками, так как мы не можем найти назначение из менее эффективных соответствий (оставшихся элементов). Следовательно, мы берем минимальный элемент (наиболее эффективное соответствие работник/работа из оставшихся) и задаем наиболее эффективное соответствие среди них. А так как на пересечении вычеркиваний находятся наиболее эффективные соответствия, которые подобраны без учета оставшихся, мы фактически их убираем прибавлением выбранного элемента из оставшихся.

После этого шага мы выделяем нули на каждой строке и столбце так, чтобы на каждой строке и столбце был выделен только 1 ноль. Это и будет нашим решением.

Доказательство алгоритма и его дополнения:

Общую постановку задачи можно сформулировать так: Пусть дан взвешенный полный двудольный граф с целыми весами ребер Kn,n, нужно найти в нем полное паросочетание (или иначе назначение) минимального веса. Вес паросочетания определяется как сумма весов его ребер. Далее будем обозначать левую и правую доли графа за X и Y соответственно, вес ребра xy — как c(xy).

В основе алгоритма лежат следующие утверждения:

Лемма 1: Если веса всех ребер графа, инцидентных какой-либо вершине, изменить (увеличить или уменьшить) на одно и то же число, то в новом графе оптимальное паросочетание будет состоять из тех же ребер, что и в старом.

Доказательство: Полное паросочетание для каждой вершины содержит ровно одно ребро, инцидентное этой вершине. Указанная операция изменит на одно и то же число вес любого паросочетания. Значит, ребро, которое принадлежало оптимальному паросочетанию в старом графе, в новом графе тоже будет ему принадлежать.

Далее будем рассматривать только графы с неотрицательной весовой функцией, так как, согласно этой лемме, задачу о назначениях на остальных графах можно свести к задаче о назначениях на них.

Лемма 2: Выделим в множествах X и Y подмножества X’ и Y’ . Пусть d = min{c(xy)| x∈X\X', y∈Y'}. Если мы прибавим d ко всем весам ребер, инцидентных вершинам из X’, а затем отнимем d от всех весов ребер, инцидентных вершинам из Y’(Далее для краткости эту операцию обозначим X'⇅Y'), тогда:

Веса всех ребер графа останутся неотрицательными.

Веса ребер вида xy, где x∈X’, y∈Y′ или x∈X\X′, y∈Y\Y′, не изменятся.

Доказательство: Рассмотрим матрицу весов графа. Не умаляя общности, можно сказать, что множества X′ и Y′ состоят из первых элементов множеств X и Y соответственно (мы упорядочиваем множества по номерам вершин). Тогда вся матрица делится на 4 блока:

	Y'	$Y \setminus Y'$
X'	$A + d - d$	$C + d$
$X \setminus X'$	$B - d$	D

Рисунок 1 – 4-х-блочная матрица весов графа

Веса группы A будут сначала увеличены, а потом уменьшены на d, поэтому они не изменятся, веса группы D вообще изменяться не будут. Все веса группы B будут уменьшены на d, но d — минимум среди этих весов, поэтому они останутся неотрицательными.

Лемма 3: Если веса всех ребер графа неотрицательны и некоторое полное паросочетание состоит из ребер нулевого веса, то оно является оптимальным.

Доказательство не требуется, так как очевидно, что паросочетание с какими-то другими весами ребер имеет больший вес и оптимальным не является.

На основании всего выше перечисленного, нами было разработано решение данной задачи для распределения k работ между k работниками при количестве работ n и количестве работников n. Доказательство нашего дополнения аналогично доказательству самого венгерского алгоритма, так как мы не меняем шаги алгоритма, а лишь проводим манипуляции с весами ребер.

В данном случае, мы сначала использовали данный алгоритм для распределения 1 работника на 1 работу. Затем, в зависимости от постановки задачи (максимизации или минимизации), мы должны исключить прошлые решения путём изменения их на наибольшее или наименьшее значение в матрице инцидентности таким образом, чтобы они были гораздо больше или меньше (в зависимости от задачи) любого значения из матрицы (в общем случае аналогично изменению их на бесконечности или минус бесконечности). Если наша задача состоит в том, чтобы продолжать искать тот же экстремум, что и на предыдущем шаге, то мы изменяем предыдущие назначения путём подстановки вместо этих значений бесконечно больших чисел в получившуюся на прошлой итерации матрицу (чтобы исключить их из решения на данной итерации). Далее мы идём по алгоритму для нахождения минимальной суммы, повторяя этот алгоритм, пока не будет найдено решение для k работников. Если же нам требуется поменять направление нахождения экстремума, то мы изменяем назначения путём подстановки бесконечно больших отрицательных значений в получившуюся на прошлой итерации матрицу и проводим алгоритм для нахождения максимальной суммы пока не будет найдено решение для k работников.

Однако возникают случаи, когда на промежуточной итерации при вычислении максимальной или минимальной суммы для матрицы $n \times n$ получается несколько различных возможностей назначить нули. В таком случае при последующем нахождении максимальной или минимальной суммы из оставшихся элементов эта сумма будет локально оптимальной. То есть в зависимости от выбора назначенных нулей последующие полученные оптимальные суммы могут отличаться. Это означает то, что решение может не быть глобально оптимальным. Например максимальная сумма для одного назначения нулей может оказаться меньше максимальной суммы для второго назначения нулей.

Таким образом наше исследование показывает рекурсивный алгоритм для нахождения локальных максимальных или минимальных сумм, с условием возможной не глобальной оптимальности, но обязательной локальной оптимальности решения задачи, что может быть полезно в машинном обучении, в транспортной логистике, в образовании, в медицине, в производственном планировании, в сфере телекоммуникаций, в дорожном строительстве, в торговле, в сфере управления проектами, в планировании и других сферах, где часто используются методы оптимизации.

Список использованных источников:

1. R.E. Burkard, M. Dell'Amico, S. Martello: Assignment Problems. SIAM, Philadelphia (PA.) 2009. ISBN 978-0-89871-663-4.

*2. Harold W. Kuhn, «The Hungarian Method for the assignment problem», Naval Research Logistics Quarterly, **2**:83—97, 1955. Kuhn's original publication.*

*3. Harold W. Kuhn, «Variants of the Hungarian method for assignment problems», Naval Research Logistics Quarterly, **3**: 253—258, 1956.*

*4. J. Munkres, «Algorithms for the Assignment and Transportation Problems», Journal of the Society for Industrial and Applied Mathematics, **5**(1):32—38, 1957 March.*

5. M. Fischetti, «Lezioni di Ricerca Operativa», Edizioni Libreria Progetto Padova, Italia, 1995.

6. R. Ahuja, T. Magnanti, J. Orlin, «Network Flows», Prentice Hall, 1993.

7. https://habr.com/ru/articles/422009/ - Электронный ресурс.

8. Бурков, Р. Э., и др. (2009). "Методы решения задачи о назначении". Издательство Санкт-Петербургского университета.

UDC

DEVELOPMENT AND ANALYSIS OF OPTIMIZATION METHODS USING MATHEMATICAL ANALYSIS

Rudzenia D.A., Kuzhovnik V.S.

Belarusian State University of Informatics and Radioelectronics1, Minsk, Republic of Belarus

Barkova E.A. – PhD in Physics and Mathematics

Annotation. The main purpose of this scientific work is to analyze existing optimization methods and develop additions to the methods. In this paper, the Hungarian algorithm method was analyzed, which is a solution to the assignment problem. An addition to the Hungarian algorithm has been developed that recursively solves the assignment problem. Our add-on allows you to programmatically solve the assignment problem using recursion.

Keywords. Optimization, assignment problem, Hungarian algorithm, recursion, optimization methods, assignment, correspondence, graphs, matrices.