УДК 519.688 ***DOI: 10.46960/1810-210X_2020_4_26***

А.Д. Леушкин, Е.А. Неймарк

КВАДРАТИЧНАЯ ЗАДАЧА О НАЗНАЧЕНИИ. ОБЗОР МЕТОДОВ, ГЕНЕРАЦИЯ ТЕСТОВЫХ ЗАДАЧ С АПРИОРНО ИЗВЕСТНЫМ ОПТИМУМОМ

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Рассматривается подход к решению задачи оптимизации работы в системе конвейерного типа на основе метода ветвления и связывания с целью уменьшения затрат на вычислительные потери без потери оптимального решения. Рассмотрен без доказательства алгоритм построения задачи с априорным оптимальным решением, на его основе проведен эксперимент по получению и последующему решению задачи с априорно известным оптимумом для сравнения с апостериорным оптимумом. С помощью разработанного алгоритма получено решение тестовых задач. Показано, что априорно заданный оптимум является близким к глобальному оптимуму задачи. В большинстве случаев перестановка, построенная алгоритмом, может быть усовершенствована. Для генерации тестовых задач реализован генератор Палубецкиса, проведен эксперимент по его верификации при помощи локальной оптимизации**.**

Ключевые слова: комбинаторные задачи, задача квадратичного назначения, оптимизация, локальный поиск, генерация тестовых задач.

Введение

Основной целью настоящей работы является рассмотрение основ квадратичной задачей назначения (далее КЗН), ее постановки, вычислительной сложности, основных методов решения. Также рассмотрен алгоритм для построения тестовой задачи с априорно известным оптимальным решением. Заранее известно, что данная задача относится к классу задач комбинаторной оптимизации, поэтому достаточно удобно реализовывать программные комплексы для решения переборными методами. Наличие активности по данной теме на популярных академических площадках показывает, что данные исследования являются актуальным [1-3].

Общая постановка

В соответствии с формулировкой, представленной Купмансом и Бекманом [4], имеется набор из N объектов и N локаций для их размещения, для этого набора определены:

- матрица$C: N \times N$**,**состоящая из элементовc_{ij}-стоимостей расположения *i-ого* объекта в *j-ой* локации;
- матрица$F: N \times N$**,** состоящая из элементов $\begin{cases} f_{ij}, \text{при} \boldsymbol{i} \neq \text{ji}, \text{j} = 1, \dots, N \\ 0, \text{при} i = j \end{cases}$, характеризующих количество ресурсов к транспортировке из *i-ого* в *j-ый* объект**;**
- матрица $D: N \times N$**,** состоящая из элементов $\begin{cases} d_{ij}, \text{при} \boldsymbol{i} \neq ji, j = 1, \dots, N \\ 0, \text{при} i = j \end{cases}$, характеризующих стоимость транспортировки единицы ресурса из *i-ой* в *j-ую* локацию**.**

Тогда полная стоимость транспортировки ресурса из *i-ого* объекта в *j-й* выглядит как произведение $(f_{ij} \cdot d_{lk})$**,** где l – расположение *i-ого* объекта в пространстве локаций, а k– расположение *j-ого* объекта в пространстве локаций.

Зададим некоторую перестановку в виде вектора $P_N = (p_1, p_2, \dots, p_N)$**,** $p_i \in \mathbb{N}$ – соответсвует назначению в i- ую локацию некоторого объекта.

Просуммируем все транспортировки между объектами, расположенными в локациях соответственно перестановке P_N , и добавим к этому стоимость расположения объектов в этих локациях при перестановке P_N. Получим полную стоимость для работы всех объектов при перестановке P_N. Необходимо найти такое назначение всех объектов на локации, чтобы эта сумма была минимальной. В прикладном смысле решение этой задачи будет являться наименьшим размером издержек.

$$\min_{p \in PN} \sum_{i=1}^{N} \sum_{j=1}^{N} f_{ij} \cdot d_{p_i p_j} + \sum_{j=1}^{N} c_{ip_j} \quad (1)$$

Математическая модель

На основе постановки задачи построим математическую модель [5]. Мы можем представить варьируемый параметр, являющийся перестановкой P_N, как некоторую матрицу $X : N \times N$, состоящую из элементов x_{ij}, при этом она должна гарантировать, что каждый объект назначен только на одну локацию (2) и каждая локацию назначена только один объект (3).

$$\sum_{j=1}^{N} x_{\text{ij}} = 1, i = 1, \dots, N \quad (2)$$

$$\sum_{i=1}^{N} x_{\text{ij}} = 1, j = 1, \dots, N \quad (3)$$

$$x_{ij} \in \{ 0, 1 \}$$

$$x_{ij} = \begin{cases} 1, & \text{если } p_i = j \text{ (объект } i \text{ расположен в локации } j) \\ 0, & \text{иначе} \end{cases}_{i,j = 1, \dots, N} \quad (4)$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} f_{\text{ij}} \cdot d_{\text{k;}} \cdot x_{\text{ik}} \cdot x_{\text{jl}} + \sum_{i,j=1}^{N} c_{\text{ij}} \cdot x_{\text{ij}} \Longrightarrow min \quad (5)$$

Тогда наименьшее значение критерия (5) при выполнении ограничений (2), (3) и (4) будет достигаться с некоторой матрицей X, которая соответствует оптимальной перестановке P_N. При данной постановке явно видно, что критерий зависит как от расположения истока, так и от расположения стока потока ресурсов. В общей постановке (1) этот факт отображается в сложности индексов при суммировании.

Вычислительная сложность

В 1976 г. Sahniand и Gonzalez показали NP-полноту [6]. Определение сложности задачи показало несостоятельность поиска алгоритма для нахождения оптимального решения задачи за полиномиальное время. Было также показано, что поиск ε-оптимального решения также является NP-полным**.** Зная это, можно назвать задачу «сложнейшей из сложных» [7]. Легко убедиться в NP-полноте проблемы, поскольку к ней можно свести другие известные NP-полные проблемы [5].

- Задача о коммивояжере: матрица *F* строится в соответствии с расстояниями между пунктами задачи коммивояжера; матрица *D* строится как матрица смежности задачи коммивояжера. Тогда оптимальная перестановка КЗН будет являться оптимальным циклом обхода для исходной задачи о коммивояжере.
- Задача о максимальной клике: для поиска клики размера *k* необходимо чтоб матрица *F* формировалась как матрица смежности графа исходной задачи; матрица *D* строится как матрица смежности клики размера *k*. Тогда максимальная клика может быть

найдена решением набора из N КЗН, каждое из этих решений будет являться кликой размера k $(1 \leq k \leq N)$если она существует в исходном графе.

Возможность полиномиального решения задачи

В прошлом пункте было показана NP-полнота проблемы в общем случае, но существуют примеры входных данных, когда возможно получение решения за полиномиальное время. Одно из таких решений описали Cristofides и Gerrard [8]. Они показали, что если матрицы D и F являются взвешенными матрицами смежности, описывающие граф типа-дерево, то проблема может быть разрешена с помощью динамического программирования за полиномиальное время. Условие является необходимым, и, если хотя бы одна из матриц под него не подходит, то задача превращается вновь в NP-полную**.**

Также существуют возможности решения за полиномиальное время задач, в обе матрицы являются матрицами смежности последовательно-параллельных графов, не содержащих двураздельный граф $K_{2,2}$.

Обзор методов решения

Как известно, для NP-трудных задач найти полиномиальный алгоритм нахождения оптимального решения не представляется возможным. Поэтому для нахождения решения используют разнообразные методики динамического программирования и техники «ветвей и границ».

Точные методы. Метод ветвей и границ

Наиболее эффективной себя показала техника ветвей и границ. Для решения КЗН наиболее используемыми являются три алгоритма [5, 9].

1. Алгоритм одиночного назначения (на каждом листе дерева поиска «ветвей и границ» объект назначается на одну локацию).
2. Алгоритм парного назначения (на каждом листе дерева поиска «ветвей и границ» фиксированная пара объектов назначается на пару локаций).
3. Алгоритм взаимной позиции (уровень дерева поиска «ветвей и границ» не соответствует назначению объекта на локацию. Частичная перестановка на каждом уровне определена с точки зрения дистанции между объектами).

Все три алгоритма объединяет то, что они начинаются с пустой перестановки и в конце гарантированно получается некоторая перестановка, являющаяся решением задачи. На практике же алгоритм одиночного назначения оказался наилучшим. Алгоритм парного назначения показал себя неэффективным в вычислении, а алгоритм взаимной позиции подходит в основном для проблем с разреженными матрицами. В методе «ветвей и границ» главным критерием для применения к задачам комбинационной оптимизации является нижняя граница. Она будет отчетливо видна поэтому ее легко вычислить. Рассмотрим наиболее ранний метод определения нижней границы КЗН.

Граница Гилмора-Лоулера

Для вычисления границы Гилмора-Лоулера используется [5, 9] минимальный вектор продукции$\langle x, y\rangle_- = \min_{P\in\Pi}\langle x, Py\rangle$и максимальный вектор продукции $\langle x, y\rangle_+ = \max_{P\in\Pi}\langle x, Py\rangle$.

В результате вычислений$\langle x, y\rangle_-$могут быть получены продукции из x^+ – содержит возрастающие компоненты хи y^- содержит убывающие компоненты у. Обозначим f_i, d_i, $i = 1, \dots, N$ – вектора матриц F и D соответственно**.** Тогда $\hat{f}_i$, $\hat{d}_i$– векторы, состоящие из *(n-1)* компонент f_i, d_i т.е. без компонент f_{ii}, d_{ii}.

Составим матрицу L состоящую из $l_{ij}=f_{ij}d_{ij}+\langle \hat{f}_i, \hat{d}_j \rangle$, $i,j=1,\dots,N$. Получим определение границы Гилмора-Лоулера для КЗН с значениями F и D как линейную задачу о назначении:

$$GLB(F,D)=\min_{P\in\Pi}\sum_{j=1}^{N} l_{i\,p(j)}=LAP(L) \qquad (6)$$

Граница, основанная на собственных числах

Данный метод [5, 9] основывается на собственных значениях входных матриц потока F и стоимости D. Из-за итеративности процесса поиска собственных чисел вычислительная сложность сильно повышается с увеличением размерности, что затрудняет построение границ в большинстве случаев.

Граница, основанная на переформулировании

Еще один итеративный метод, не отличающийся эффективностью [5]. Каждую итерацию решается (n^2+1) линейных задач назначения размерности n.

Субоптимальные алгоритмы

Алгоритмы, направленные на получение некоторого решения. В отличие от «точных методов», наиболее часто такие алгоритмы основаны на стохастических принципах и могут с некоторой вероятностью сойтись в глобальном оптимуме. Существует большое количество разновидностей получения субоптимального решения, рассмотрим концепции наиболее распространенных из этих алгоритмов.

Метод ограниченного перечисления

Перебор всех перестановок для нахождения оптимального решения КЗН для задач с размерностью, превышающей отметку в 15 локаций (и объектов), занимает чересчур много времени. Поэтому существует множество механизмов ограничения перебора. Одним из наиболее простых способов является явное ограничение времени процесса [5] поиска. В этом случае поиск заканчивается строго в назначенное время и лучшим решением считается достигнутое до окончания поиска. Также можно рассматривать случаи, в которых найденное решение некоторое время не получало улучшений. Тогда верхняя граница опускается несколько ниже, что уменьшает дерево возможного перебора и увеличивает скорость перебора, хотя и есть возможность потерт некоторых решений. Это нивелируется за счет предположения, что оптимальное и около-оптимальное решение в КЗН отличаются незначительно.

Метод улучшения

Основные эвристические методы для КЗН – это методы улучшения [5]. Они начинают работу с некоторого решения и дерева возможных перестановок и в процессе работы пытаются улучшить решение.

Локальный поиск (LSA)

Итерационный алгоритм улучшения решения [5], т.е., на каждом шаге выбирается наилучшее решение из допустимой окрестности. Для КЗН хорошо подходит метод, при котором в решении перебираются решения от изменения позиций всех возможных пар элементов решения и для следующего шага выбирается наилучший. Алгоритм заканчивает работу в местах локального оптимума – случаях, когда ни одна перестановка не улучшает критерий для решения.

Табу-поиск

Табу-поиск [5] – это техника для преодоления локальной оптимальности в комбинационном поиске. Фундаментальная идея заключается в ограничении направления поиска на каждом шаге для получения наибольшего качества решения с наибольшей эффективностью.

Метод отжига

Является рандомизированным методом локального поиска [5], позволяющего избежать плохих локальных оптимумов. Исходит из моделирования физического процесса отжига. Благодаря подражанию процессам, происходящим с веществом при постепенном охлаждении есть возможность поучать сравнительно хорошие решения, являющиеся локальным оптимумом. Для алгоритма имитации отжига доказана асимптотическая сходимость к глобальному оптимуму что делает его привлекательным для использования.

Генетический алгоритм

Генетический алгоритм является стохастическим методом [5], который основывается на естественных свойствах существ к адаптации в природе и принципах естественного отбора. Использование генетических операторов и определенной стратегии мы получим улучшение приспособленности популяции и в итоге кодировка особи с наилучшей приспособленностью будет являться решением задачи. Для эффективного использования рекомендуется использование параллельных вычислений.

Муравьиная система

Муравьиная система [10] – это эволюционный алгоритм поиска, основанный на популяционных схемах строящихся на поведении муравьев в природе. Базовой идеей является принцип взаимодействия муравьев при построении кратчайшего пути от муравейника до цели. Это взаимодействие происходит в некоторой среде с помощью определенного химического соединения – феромона, оставляемого на земле каждой особью. Кратчайшим путем будет считаться тот, на котором феромона было оставлено наибольшее количество, что соответствует наибольшему количеству муравьев, следующих этому маршруту. Таким образом, феромон выступает в роли памяти системы, которая указывает на наличие ранее построенного «хорошего» решения, что можно использовать и для решения КЗН.

Жадный рандомизированный адаптивный поиск (GRASP)

GRASP – техника итеративного рандомизированного отбора [5**]**, каждая итерация которого обеспечивает приближенное к оптимальному решение проблемы. При этом наилучшее решение из всех проделанных итераций сохраняется как финальное. Алгоритм состоит из двух шагов.

1. Конструкция первичного решения некоторой стохастической функцией.
2. Использование техники локального поиска над построенным решением в надежде его улучшения.

В рамках КЗН GRASP в сочетании с методом ветвей и границ позволил получить оптимальное или близкое к оптимальному решение для большинства задач КЗН.

Генерация тестовых задач

В парадигме вычислительной оптимизации генерация тестовых задач является важной частью. Эффективность алгоритма определяется совокупностью следующих критериев [5]: точность решения, скорость алгоритма, универсальность алгоритма в рамках данного класса задач. Для большинства трудных задач существующая теоретическая база не может обеспечить измерение критериев для оценки алгоритма. Поэтому необходимо иметь инструментарий по получению стандартизированных данных и получению решения, следуя конкретному алгоритму для последующей работы над алгоритмами.

Эволюцию и тестирование алгоритма удобнее реализовывать при наличии конкретных задач с заведомо известным оптимальным решением. Используя такую задачу, мы имеем возможность как получать необходимые данные о конкретном алгоритме, так и сравнивать несколько алгоритмов между собой. В рамках КЗН не всегда удается определить оптимум для задачи большой размерности, что приводит к неточностям при оценке алгоритмов. Наиболее доступным методом решения этой проблемы является построение задачи по некоторому принципу, позволяющему знать оптимальное решение до проведения эксперимента.

Генератор Палубецкиса

Один из первых методов конструирования тестовой задачи с известным решением предложил Г.С. Палубецкис [11]. Предполагается что матрица D строится из решеточного графа. На вход: N – размерность задачи, ω – параметр для инициализации F, $z < \omega$ – верхний предел для случайных чисел

Вывод: матрицы F и D и оптимальная перестановка p^*.

Алгоритм следующий.

1. Конструируем матрицу $D = \|d_{ij}\|$, элементы которой являются длиной пути между двумя вершинами (расстояние между соседними вершинами $= 1$) двумерного сеточного графа размером $r \times s, r \cdot s = N$.

2. Определяем $\omega = (\omega_{ij}), \omega_{ij} = \omega$ и вычислить $g_{ij} = 2 - d_{ij} i, j = \overline{1, N}$.

3. Находим пару (l, m) такую, что $d_{lm} = \max\limits_{i,j}\{d_{ij}\}$ и $g_{lm} \le 0$. Если необходимая пара не найдена, то переходим к шагу (5).

4. Случайно выбираем k так, что получим один из кратчайших путей от l до m ($|d_{lk} - d_{mk}| \le 1$); случайно выбираем $\Delta \in [0, z < \omega]$. $\omega_{lm} = \Delta, \omega_{lk} = \omega_{lk} + (\omega - \Delta),\ \omega_{mk} = \omega_{mk} + (\omega - \Delta), g_{lm} = g_{lk} = g_{mk} = 1$.

Возвращаемся к шагу (3).

5. Генерируем случайную перестановку $p^* = p^*(i), i = \overline{1, N}$;формируем матрицу $F = \|f_{ij}\|$, в которой$f_{ij} = \omega_{uv}$, где $i = p^*(u)$и $j = p^*(v)$.

6. Возвращаем F, D и p^*.

Алгоритм позволяет получить задачу и перестановку к ней. Значение критерия этой перестановки будет находиться достаточно близко к оптимальному.

Применение генератора

Сгенерируем задачи разной размерности (6, 10, 16, 20, 26, 30) с фиксированными параметрами и проверим, можно ли улучшить решение с помощью алгоритма локальной оптимизации. Результаты оформим в виде табл. 1 и табл. 2. Сгенерируем задачи с параметрами: $\omega = 1, z = 0$.

Таблица 1

Генерация тестов разной размерности с параметрами: $\omega = 1, z = 0$.

Размерность	Априорное решение	После локальной оптимизацией	Количество шагов локальной оптимизации	Среднее значение при случайной перестановке
6	54	54	1	68,56
10	232	232	1	301,78
16	796	788	3	933,2
20	1480	1426	8	1657,7
26	3662	3634	5	4982,72
30	4244	4114	12	4777,9

Сгенерируем задачи с параметрами: $\omega = 9, z = 3$.

Таблица 2

Генерация тестов разной размерности с параметрами: $\omega = 9, z = 3$.

Размерность	Априорное решение	После локальной оптимизацией	Количество шагов локальной оптимизации	Среднее значение при случайной перестановке
6	480	480	1	582,08
10	2118	2034	3	2529,6
16	7024	6862	6	8002,5
20	12648	12180	6	14311,02
26	32248	31768	11	43093,32
30	36642	35936	11	41185,94

Примечание: первый шаг в алгоритме локальной оптимизации означает, что лучших перестановок не было найдено; среднее значение при случайно перестановке берется как среднее за 100 случайных перестановок.

Прослеживается некоторое отдаление априорного оптимума от оптимума локальной оптимизации при увеличении размерности задачи.

С увеличением параметра ω – увеличивается матрица цен F и соответственно значение итогового критерия. Параметр z влияет случайные процессы и соответственно на вариативность генерации задач.

Вычислительный эксперимент

Попробуем найти решение КЗН с помощью одного из перечисленных методов. Для простоты реализации используем метод локального поиска, для начала работы с которым необходима некоторая перестановка. Попробуем применить метод для случайной перестановки. Будем искать решение для задач, полученных при помощи генератора Палубецкиса. Для усреднения значений отклонения от оптимума при случайном построении исходной перестановки будем использовать *10* запусков для каждой задачи из списка. Полученные результаты оформим в виде таблиц (табл. 3, табл. 4), за оптимум примем значение, полученное после улучшения априорного решения из генерации:

Таблица 3

Результаты вычислительного эксперимента для задач с параметрами: $\omega = 1, z = 0$.

Размерность задачи, $\boldsymbol{\omega = 1\ z = 0}$	Оптимум	Среднее количество шагов алгоритма	Диапазон получаемых значений (min-max)	Медианное значение
6	54	3.5	54-60	55
10	232	5	232-256	243
16	788	10	790-810	800
20	1426	12	1436-1466	1452
26	3634	23	3610-3634	3623
30	4114	24	4100-4190	4145

Таблица 4

Результаты вычислительного эксперимента для задач с параметрами: $\omega = 9, z = 3$

Размерность задачи, $\boldsymbol{\omega = 9\ z = 3}$	Оптимум	Среднее количество шагов алгоритма	Диапазон получаемых значений (min-max)	Медианное значение
6	480	3.5	480-500	486
10	2034	6	2046-2236	2100
16	6862	10	6854-7046	6924
20	12180	16	12196-12574	12372
26	31768	24	31648-31768	31681
30	35936	25	35940-36472	36164

Примечание: За шаг алгоритма принимался переход от одной перестановки к другой после перебора возможных улучшений.

Поиск оптимума с помощью локальной оптимизации достигает априорного значения из сгенерированной задачи.

Программная система

Программа реализована на языке NetCore 3.1 C#. Среда разработки Microsoft Visual Studio 19. Структура файла данных представляет: размерность задачи, матрицы ***F*, *Du*** опционально *C*.

Система позволяет:

1) решать задачу, импортированную из текстового файла;
2) генерировать задачу по алгоритму из п.3.;
3) экспортировать данные задачи в текстовый файл;
4) находить решение с помощью алгоритма локального поиска.

Выводы

Рассмотрена область задач комбинаторного типа. Выделена и локализована задача квадратичного назначения, которая в силу отсутствия аналитического метода нахождения оптимального решения является достаточно привлекательной для решения эвристиками. Экспериментально подтверждено, что разница между глобальным и локальным оптимумами незначительна. Для корректного оценивания алгоритма был реализован метод конструирования задач с априорным оптимумом.

Библиографический список

1. **Hameed, A.** A new hybrid approach based on discrete differential evolution algorithm to enhancement solutions of quadratic assignment problem / A. Hameed, B. Aboobaider, M. Mutar & N. Choon // International Journal of Industrial Engineering Computations. – 2020. – 11(1). – P 51-72.
2. **Xiang, Rui** Efficient and Robust Shape Correspondence via Sparsity-Enforced Quadratic Assignment / Rui Xiang, Lai Rongjie, Zhao Hongkai // arXiv preprint arXiv:2003.08680, 2020.
3. **Singh, Nirmal Jeet** Simulation and analysis of quadratic assignment problems (QAP) using ant colony optimization / Nirmal Jeet Singh, S.R. Mediratta // Arya Bhatta Journal of Mathematics and Informatics 12.1, 2020. – P. 25-30.
4. **Koopmans, T.C.** Assignment problems and the location of economic activities / T.C. Koopmans, M.: Beckman // Econometric. – 1957. – № 25. – P. 53-76.
5. **Panos, M.P.** The Quadratic Assignment Problem: A Survey and Recent Developments / M.P. Panos, Franz Rendl. Henry Wolkwicz, 1994.
6. **Sahni, S.** P-complete Approximation Problems / S. Sahni, T. Gonzalez // Journal of the ACM 23, 1976. – P. 555-565.
7. **Clayton, W.** Commander. Survey of the Quadratic Assignment Problem, with Applications / W.Clayton. – Darmstadt University of North Carolina at Chapel Hill, 2005.
8. **Christofides, N.** Special cases of the quadrtic assignment problem / N. Christofides, M. Gerrard // Management Science Research Report 391, Carnegie Mellon University, April 1976.
9. **Li, Y.** Lower Bounds for ThequadraticAssigment Problem / Y. Li, P.M. Pardalos, K.G. Ramakrishnan, M.G.C. Resende, 1994.
10. Stützle T. MAX-MIN ant system for quadratic assignment problems / T. Stützle // Research Report AIDA-97-04, Department of Computer Schience, Darmstadt University of Technology, Germany, 1997.
11. **Palubetskis G.S.** A generator of test quadratic assignment problems with known optimal solution / G.S. Palubetskis, Zh. Vychisl. Mat. Mat.Fiz., 1988, Volume 28, Number 11. – P. 1740-1743.

Дата поступления
в редакцию: 03.07.2020

A.D. Leushkin, E.A. Neumark

THE QUADRATIC ASSIGMENT PROBLEM. METHODS OVERVIEW, GENERATION TEST PROBLEM WITH KNOWN OPTIMAL SOLUTION

Lobachevsky State University of Nizhny Novgorod

Purpose: The article is devoted to the consideration of the quadratic assignment problem. A canonical formulation of the problem is given; some methods for obtaining a solution are surveyed.
Approach: An algorithm for constructing a problem with an a priori optimal solution is considered without proof; on its basis, an experiment was conducted to obtain and subsequently solve a problem with an a priori known optimum for comparison with an a posteriori optimum.
Results: We got some test problems using the above algorithm, compared the a priori optimum with the optimum obtained by the local search method. It was shown that the a priori solution has a strong advantage over random permutations.
Originality/value: The Palubetskis generator was implemented in order to generate test problems, it was verified using local optimization.

Key words: combination problem, quadratic assignment problem, optimization, local search,test problem generation.