Some approaches to solving the NP-hard Mixed Chinese Postman Problem

Maria Gordenko

Software Engineering School
National Research University Higher School of Economics
Moscow, Russia
mkgordenko@edu.hse.ru

Sergey Avdoshin

Software Engineering School
National Research University Higher School of Economics
Moscow, Russia
savdoshin@hse.ru

Abstract. The routing problems are important for logistic and transport sphere. Basically, the routing problems related to determining the optimal set of routes in the multigraph. The Chinese postman problem (CPP) is a special case of the routing problem, which has many potential applications. We propose to solve the MCPP (special NP-hard case of CPP, which defined on mixed multigraph) using the reduction of the original problem into General Travelling Salesman Problem (GTSP). The variants of CPP are pointed out. The mathematical formulations of some problems are presented. The algorithm for reduction the MCPP in multigraph into GTSP is shown. The experimental results of solving MCPP in multigraph through the reduction into GTSP are presented.

***Keywords*:** Mixed Chinese Postman Problem, General Routing Problem, Vehicle Routing Problem, Arc Routing Problem, Asymmetric Traveling Salesman Problem, Generalized Traveling Salesman Problem, Traveling Salesman Problem, heuristic algorithm.

1 The General Routing Problem

The routing is one of the most important problem in the optimization researches. The GRP is to define a minimum cost set of routes (one route also is possible) in a multigraph, that must include some required vertices and pass through some required edges and arcs of the original multigraph [1].

Formally, the GRP is defined on multigraph $G =< V, E, A, C >$, where

V is a set of vertices,

A is a multiset of directed edges (arc);

E is a multiset of undirected edges (edges);

C: $E \cup A \to R_+$ is a cost function giving non-negative weights of arcs and edges between vertices.

In the routing problems, it is not necessary to visit all vertices, edges and arcs of the multigraph. Two subsets of edges and arcs $A_R \subseteq A$ and $E_R \subseteq E$ are defined. The arcs and edges from A_R and E_R must necessarily appear in the solution. Let the subset of vertices $V_R \subseteq V$ consist of those vertices that must appear in the route.

The goal of all routing problems is to define a minimum cost set of routes, that traverses all the arcs and edges from the multisets A_R and E_R and includes all vertices of the set V_R.

2 The Vehicle Routing Problem

The VRP is a special case of the GRP with $A_R = \emptyset$ and $E_R = \emptyset$, i.e. the restrictions on the edges and arcs, which must necessarily appear in the route, are absent. The VRP is to determine the Hamiltonian cycle of minimum cost, which traverse all vertices of the subset V_R [1].

In the case, when $V_R = V$, the problem reduces to one of the most famous problem of combinatorial optimization – the classical Traveling Salesman Problem (TSP).

3 The Arc Routing Problem

Another special case of the GRP is the Arc Routing Problem (ARP), it is to determine the minimum cost set of routes, that traverses all required edges E_R and all required arcs A_R of original multigraph. In the ARP, there are no restrictions on the presence of vertices in the route, i.e. $V_R = \emptyset$. The CPP is the variant of ARP. In the original formulation, the CPP is the problem, where the postman should traverse through every street in the given area.

4 The Variants Of Chinese Postman Problem

The CPP was originally studied by the Chinese mathematician Kwan Mei-Ko in 1960. A problem is called the Chinese Postman Problem after him. Kwan Mei-Ko defined the problem on undirected graph. Today, there are many various classifications of CPP, including classifications based on the graph type, on the type of solution route and other restrictions and additions [2].

In the modern world, the number of companies and industries that are interested in building an optimal route of product delivery is growing. For example, the postman delivering letters or leaflets wants to know the optimal route that traverses every street in the given area, starting and ending at the office. Apart from the traditional application of the CPP to solving the routing problems such as path planning of snowplows or serving teams, there are a wide range of applications including robot exploration, testing web site usability and finding broken links [3].

Below, the most popular variants of the CPP are presented.

4.1 The Undirected Chinese Postman Problem

The formulation of the Chinese Postman Problem in undirected graph (UCPP) is an original formulation of the CPP problem. The UCPP is a special case of ARP, where $A = \emptyset$ and $E_R = E$. The UCPP belong to class of P problems.

4.2 The Directed Chinese Postman Problem

The Chinese Postman Problem in directed graph (DCPP) is the modification of UCPP, where every arc (directed edge) can be traversed in one direction. Another name of problem is the New York Street Sweeper Problem. The DCPP is a special case of ARP, where $A_R = A$ and $E = \emptyset$. The DCPP belong to class of P problems.

4.3 The Windy Chinese Postman Problem.

The Windy Chinese Postman Problem (WCCP) is the interesting generalization of classical CPP, which has many real uses. In WCPP the cost of traversing some edge depends on way of traversing. The WCPP is a special case of ARP, where $E_R = E$, $A = \emptyset$ and at least for one edge the cost of traversing in direct way is differ from cost of traversing it in opposite way. The DCPP belong to class of NP-hard problems, which cannot be solved in polynomial time.

4.4 The Rural Chinese Postman Problem.

The Rural Chinese Postman Problem (RCPP) is a special case of ARP, where $A_R \subseteq A$, $E_R \subseteq E$. Another name of RCPP is the Selecting Chinese Postman Problem. In all above defined CPP problems, it is necessary to find a closed shortest route, that traverses all edges or arcs of the original multigraph at least once. In the real world, it is not always necessary to traverse all roads (edges or arcs). It is enough to traverse only a set of requires arcs and edges (A_R and E_R). The RCPP belong to class of NP-hard problems, which cannot be solved in polynomial time.

4.5 The Mixed Chinese Postman Problem

The Mixed Chinese Postman Problem (MCPP) is a well-known version of the CPP, where multigraph contains both arcs and edges. The MCPP is a special case of ARP, where $E_R = E$ and $A_R = A$. The MCPP belong to class of NP-hard problems.

There are other variants of the problem, such as Hierarchical Postman Problem (HCPP), k-Chinese Postman Problem (k-CPP), Chinese Postman Problem with Time Windows and others.

5 Mathematical formulation of the MCPP

The MCPP is one of the most important problem of the ARP. The MCPP is a special case of ARP, for which $A_R = A \neq \emptyset, E_R = E \neq \emptyset$.

An edge $\{v_i, v_j\}$ (an unordered pair of vertices) from the set E is fixed. Let (v_i, v_j) is ordered pair of vertices (this mean, that should traverse $\{v_i, v_j\}$ from vertex v_i to vertex v_j). Note, that, $\forall\{v_i, v_j\} \in E,\ c((v_i, v_j)) = c(\{v_j, v_i\})$ and $C((v_j, v_i)) = C(\{v_j, v_i\})$, it means that the cost of traversing the edge in any directions is the same.

The mathematical formulation of the MCPP problem is presented below.

Let $I = \{1, 2, \dots, |E + A|\},\ L = \{1, 2, \dots, |V|\}$. Indexation on the set of vertices V is defined as $inv: V \to L,\ \forall v_i \in V\ \forall v_j \in V\ v_i \neq v_j \Rightarrow i \neq j,\ i = inv(v_i)$. On the multiset $E \cup A$ indexation is defined as $inea: E \cup A \to I,\ \forall e_i \in (E \cup A)\ \forall e_j \in (E \cup A)\ \ e_i \neq e_j \Rightarrow i \neq j, i = inea(e_i)$.

Route $\mu = (v_{l_1}, e_{p_1}, v_{l_2}, e_{p_2}, \dots, v_{l_n}, e_{p_k})$ is the solution of the MCPP that satisfies to the following properties:

- $e_{p_i} = (v_{l_i}, v_{l_{i+1}}), i = 1,2, \dots, k-1$;
- $e_{p_k} = (v_{l_k}, v_{l_1})$;
- $E \cup A \,/\{e_{p_1}, e_{p_2}, \dots, e_{p_k}\} = \emptyset$.

Let $C(\mu) = \sum_{i=1}^{k} C(e_{p_i})$ is the cost of the MCPP route. Let $\mathcal{M}$ be a set of solutions of the MCPP. It is necessary to find a route $\mu_0 \in \mathcal{M}$ that satisfies the following property $\forall \mu \in \mathcal{M}\ C(\mu_0) \leq \mathrm{C}(\mu)$ or $C(\mu_0) = \min_{\mu \in \mathcal{M}}(\mathrm{C}(\mu))$.

6 Reduction the MCPP into asymmetric GTSP

The MCPP can be reduced to an equivalent ARP. When problem defined in directed graph (DCPP), it can be reduced to the asymmetric TSP. When problem defined in mixed or undirected graph, it can be reduced to GTSP [4-6].

6.1 Description of reduction algorithm

Originally, the reduction algorithm was presented for the graph [3, 4]. The algorithm modification, applicable to the multigraph, is given below [7].

The process of reduction the MCPP to GTSP is to transform the original graph $G =< V, E \cup A, C >$ into equivalent GTSP on complete graph $\tilde{G} =< \tilde{V}, \tilde{A}, \tilde{C} >$.

Each arc $a_{ij}^k \in A$ between to vertices $v_i \in V,\ v_j \in V$ is represented as vertex $v_{ij}^k \in \tilde{V}$, which must be used in the solution at least once, where k is the serial number of parallel arc. Each edge $e_{ij}^k \in \tilde{E}$ between to $v_i \in V,\ v_j \in V$ is represented as two vertices $v_{ij}^{k_1} \in \tilde{V}, v_{ij}^{k_2} \in \tilde{V}$, one of which must be used in the solution, another may not be used, where k is a serial number of parallel edge, k_1, k_2 are the serial numbers of parallel arcs between two vertices. After replacing the arcs and edges in vertices, the cost from each pair of vertex $v_{ab}^{k_1} \in \tilde{V}, v_{cd}^{k_2} \in \tilde{V}$ in graph $\tilde{G}$ compute, as $\widetilde{c_{ij}} = d_{bc} + c_{cd}^{k_2}$,

Table 1. Formulas for computing arc costs of Asymmetric GTSP

		1	2	3	4	5	6	7	8
		v_{12}^1	v_{13}^1	v_{13}^2	v_{21}^1	v_{23}^1	v_{23}^2	v_{31}^1	v_{32}^1
1	v_{12}^1	-	$s_{21}+c_{13}^1$	$s_{21}+c_{13}^2$	$s_{22}+c_{21}^1$	$s_{22}+c_{23}^1$	$s_{22}+c_{23}^2$	$s_{23}+c_{31}^1$	$s_{23}+c_{32}^1$
2	v_{13}^1	$s_{31}+c_{12}^1$	-	$s_{31}+c_{13}^2$	$s_{32}+c_{21}^1$	$s_{32}+c_{23}^1$	$s_{32}+c_{23}^2$	$s_{33}+c_{31}^1$	$s_{33}+c_{32}^1$
3	v_{13}^2	$s_{31}+c_{12}^1$	$s_{31}+c_{13}^1$	-	$s_{32}+c_{21}^1$	$s_{32}+c_{23}^1$	$s_{32}+c_{23}^2$	$s_{33}+c_{31}^1$	$s_{33}+c_{32}^1$
4	v_{21}^1	$s_{11}+c_{12}^1$	$s_{11}+c_{13}^1$	$s_{11}+c_{13}^2$	-	$s_{12}+c_{23}^1$	$s_{12}+c_{23}^2$	$s_{13}+c_{31}^1$	$s_{13}+c_{32}^1$
5	v_{23}^1	$s_{31}+c_{12}^1$	$s_{31}+c_{13}^1$	$s_{31}+c_{13}^2$	$s_{32}+c_{21}^1$	-	$s_{32}+c_{23}^2$	$s_{33}+c_{31}^1$	$s_{33}+c_{32}^1$
6	v_{23}^2	$s_{31}+c_{12}^1$	$s_{31}+c_{13}^1$	$s_{31}+c_{13}^2$	$s_{32}+c_{21}^1$	$s_{32}+c_{23}^1$	-	$s_{33}+c_{31}^1$	$s_{33}+c_{32}^1$
7	v_{31}^1	$s_{11}+c_{12}^1$	$s_{11}+c_{13}^1$	$s_{11}+c_{13}^2$	$s_{12}+c_{21}^1$	$s_{12}+c_{23}^1$	$s_{12}+c_{23}^2$	-	$s_{13}+c_{32}^1$
8	v_{31}^2	$s_{21}+c_{12}^1$	$s_{21}+c_{13}^1$	$s_{21}+c_{13}^2$	$s_{22}+c_{21}^1$	$s_{22}+c_{23}^1$	$s_{22}+c_{23}^2$	$s_{23}+c_{31}^1$	-

where d_{bc} is the shortest distance between vertices $v_b \in V$, $v_c \in V$ in original multigraph G.

Then, the compete graph $\tilde{G}$ is partitioned into clusters as follows: each arc and each edge is separate cluster. The number of clusters is equal to $|A \cup E|$. The graph partitioned into clusters because edge can be traverse in two ways, for solving the MCPP any way is appropriate and the problem transforms into GTSP [5-7].

The GTSP is a variation of the Traveling Salesman Problem in which all vertices are divided into clusters, and solution consist from only one vertices from each cluster.

7 The example of reduction

The example of original multigraph is shown on Fig. 1. Each arc and edge has the cost of traverse. Each vertex has the serial number.

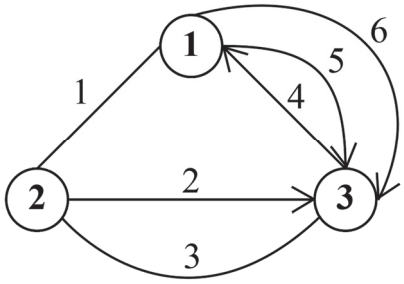

Fig. 1. The original MCPP problem in multigraph

We replace each edge by a pair of two oppositely directed arcs and specify the numbering of parallel arcs between each pair of vertices (see Fig. 2). In multigraph only one arc a_{12}^1 or a_{21}^1 is required, because these arcs represent one edge. The same applies to arc a_{23}^2 or a_{32}^1.

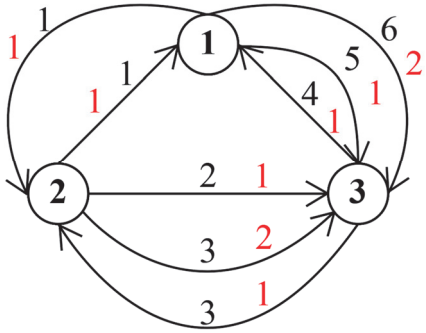

Fig. 2. The results of numbering each parallel arc

After that, should replace each arc and edge as vertex. We received new graph $\tilde{G}$ with 8 verteces. The $\tilde{V}$ can be calculates according to the formula $|\tilde{V}| = |\tilde{A}| + 2|\tilde{E}|$.

The cost from each pair of vertices is calculated by formulas (see Table 1, see Table 2). The vertices represent the edge are marked with a color in the table (different colors for different edges).

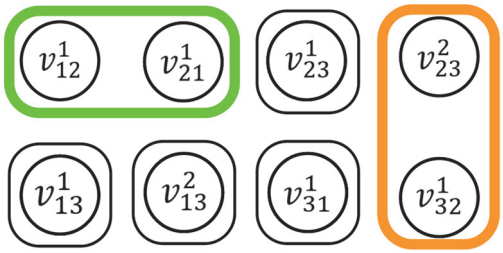

Fig. 3. The vertices and clusters of transformed problems

Table 2. The Cost Matrix

		1	2	3	4	5	6	7	8
		v_{12}^1	v_{13}^1	v_{13}^2	v_{21}^1	v_{23}^1	v_{23}^2	v_{31}^1	v_{32}^1
1	v_{12}^1	-	6	7	1	2	3	6	5
2	v_{13}^1	5	-	10	4	5	6	4	3
3	v_{13}^2	5	9	-	4	5	6	4	3
4	v_{21}^1	1	5	6	-	3	4	7	6
5	v_{23}^1	5	9	10	4	-	6	4	3
6	v_{23}^2	5	9	10	4	5	-	4	3
7	v_{31}^1	1	5	6	2	3	4	-	6
8	v_{32}^1	2	6	7	1	2	3	6	-

Then verteces from $\tilde{V}$ are partitioned into clusters. Fig. 3 depicts the vertices and clusters of transformed graph. The reduction of the MCPP to Asymmetric GTSP is received. After transformation of the original MCPP into the GTSP, the existing algorithm for GTSP can be applied.

8 Algorithms of GTSP solving

In the work, the nearest neighbor heuristic algorithm (NN) and its modifications were applied to solve the GTSP problem.

8.1 Nearest Neighbor Heuristic (NN)

Nearest Neighbor heuristic belongs to the group of tour construction heuristics. In the tour construction methods, the route is built by adding new vertices at each step, according to some rules, while the already existing tour does not improve.

The algorithm starts building the route from the some starting vertex, and then selects the nearest vertex from another cluster to the start point and adds it to the route. Then, the nearest vertex, belonging to an unused cluster, should appear in the route, until all the clusters are used. After adding the vertices, we return to the starting point.

Time complexity of the algorithm in the best and worst case – $O(|\tilde{\tilde{V}}|^2)$ [5].

8.2 Repetitive Nearest Neighbor Heuristic (RNN)

Since the length of the obtained route in NN depends on the considered starting vertex, another variant of the nearest-neighbor is the repeated nearest neighbor, which calculates the cost of the route, when NN is applied to each vertex as starting vertex, and chooses the best route among all.

Time complexity of the algorithm in the best and worst case – $O(|\tilde{\tilde{V}}|^3)$ [5].

8.3 Improved Nearest Neighbor Heuristic (INN)

Another modification of NN is the heuristic, in which the shortest edge between two vertices from different clusters is selected as the starting edge for the route, and then NN is applied to the found edge (the end vertex of shortest edge is the starting vertex for NN).

Time complexity of the algorithm in the best and worst case – $O(|\tilde{\tilde{V}}|^2)$ [5].

8.4 Repetitive Improved Nearest Neighbor Heuristic (RINN)

This method is a joint modification of the methods RNN and INN, which based on the fact, that in the problem there are several edges with a minimum weight. It is proposed to find the lengths of routes for all minimal edges [6].

Time complexity of the algorithm in the best case is $O(|\tilde{\tilde{V}}|^2)$ and in the worst case is $O(|\tilde{\tilde{V}}|^3)$ [6].

8.5 Double-Ended Nearest Neighbor Heuristic (DENN)

The algorithm starts building the route from some starting vertex, and, then, selects the nearest vertex to the start vertex and adds it to the route. Then the nearest vertex, belonging to an unused cluster, to the first vertex in the solution or the last is added, should appear in the route, until all the clusters are used. Thus, the route grows from both ends, the vertices can be added at the beginning of the route, and at the ending of the route. After adding all the vertices, we return to the starting point.

Time complexity of the algorithm in the best and worst case – $O(|\tilde{\tilde{V}}|^2)$ [5].

8.6 Loneliest Nearest Neighbor Heuristic (LNN)

The main idea of the heuristic is that the vertices most remote from the others should be paid special attention during the construction of the route to avoid their later inclusion in the route with higher cost. To make such heuristic possible, the concept of "loneliness" of the city was introduced. Together with the distance to the nearest neighbor, the closeness of the nearest neighbors will also be the criteria for adding the next vertex to the route. "Lonely" neighbors, i.e. most remote, will be preferable to others. At the preprocessing stage, a new distance matrix is obtained, such that shorter new distances from the vertex to the others are a weighted function of short old distances to these vertices, and a higher loneliness of this city. Then NN is applied to the new matrix [5].

Time complexity of the algorithm in the best and worst case – $O(|\tilde{\tilde{V}}|^2)$.

8.7 Double-Ended Nearest Neighbor Heuristic (DENLN)

The heuristic is a modification of the NLN and DENLN heuristics, the weighted distance function is also calculated here, considering the "loneliness" of the city, but the DENLN algorithm is already applied in the next stages [5].

Time complexity of the algorithm in the best and worst case – $O(|\tilde{\tilde{V}}|^2)$ [6].

9 Methods of testing

The algorithm for graph transformations and solving GTSP was written on C++ in MS Visual Studio 2015.

To test all algorithms, the two databases were used. For multigraph, the test data sets were not found. However, graph is a special case of multigraph (without parallel arcs and edges) and algorithm can be tested on graph data sets. In Bönisch's database the input data for 50, 100, 200 vertices in graph are presented [8]. For each dimension, there are 75 different tasks. The test data from Angel Corberan web-site for 500, 1000, 1500, 2000 and 3000 vertices also was used [9]. For each dimension, there are 25 different tasks.

The time performance and error rate of proposed approach were measured as follows:

- Test data were loaded in console program.
- The measurements for each input data set were carried out 10 times. The results of computational time were obtained as the average of 10 runs of the program: $T_{av} = \frac{T_1+\cdots+T_{10}}{10}$.
- Error rate of the TSP algorithms was evaluated according to the formula $Error = \frac{\mathrm{C}(\mu)-\mathrm{C}(\mu_0)}{\mathrm{C}(\mu_0)}$, where $\mathrm{C}(\mu)$ is the resulting length of the route of the MCPP, $\mathrm{C}(\mu_0)$ is the optimum length of the route of the MCPP given in input data.

All test provides on Mac Book Pro 13 retina 2014 (Intel Core i5, 2.6 GHz).

Table 3. The Pareto-Optimal Algorithms For Solving MCPP Through The Reduction To Gtsp

	$\|V\| = 50$	$\|V\| = 100$	$\|V\| = 200$	$\|V\| = 500$	$\|V\| = 1000$	$\|V\| = 1500$	$\|V\| = 2000$	$\|V\| = 3000$
NN			+		+	+		
INN	+	+	+	+	+	+	+	+
RINN	+							+
DENN								
NLN						+		
DENLN								
RNN	+	+	+	+	+	+	+	+

10 Obtained results

For all tests from test database the time and error rate were computed. On Fig. 4, Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 9, Fig. 10 the results are presented in the form of diagrams, where for each described above algorithm the average computational time and error rate are depicted.

Diagrams allow determine the Pareto-Optimal algorithms on two criteria: computational time and error rate. For all tested dimension this groups are similar and contain RNN and INN algorithms.

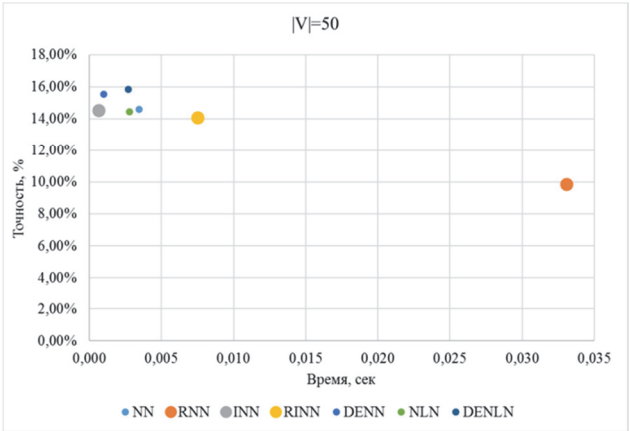

Fig. 4. The results for |V|=50

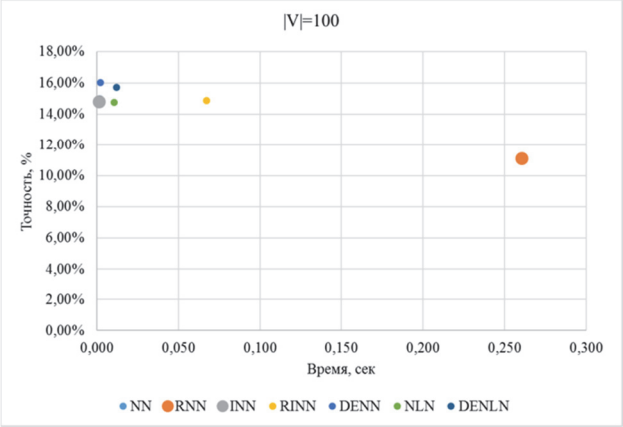

Fig. 5. The results for |V|=100

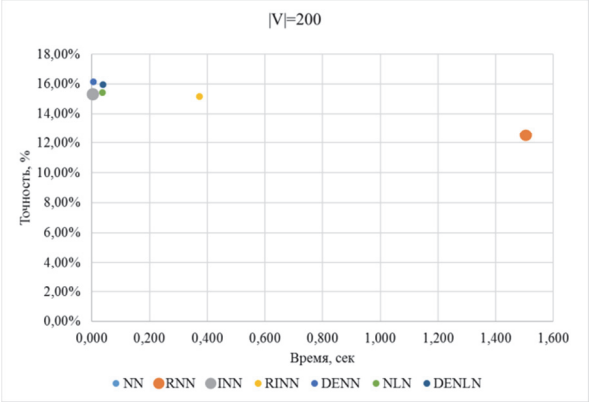

Fig. 6. The results for |V|=200

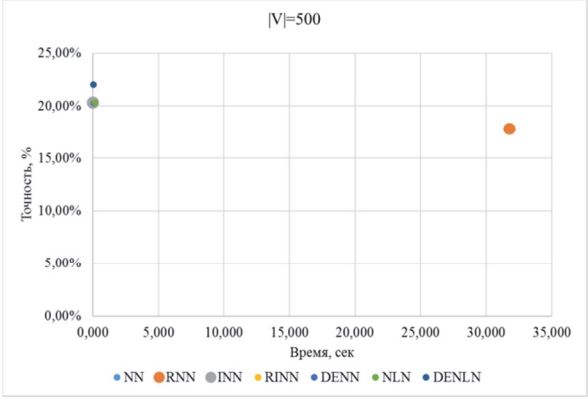

Fig. 7. The results for |V|=500

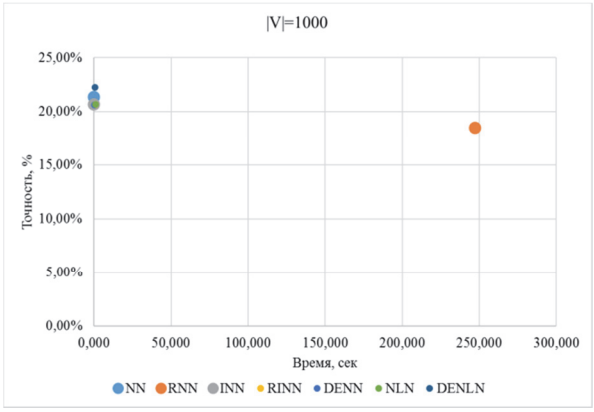

Fig. 8. The results for |V|=1000

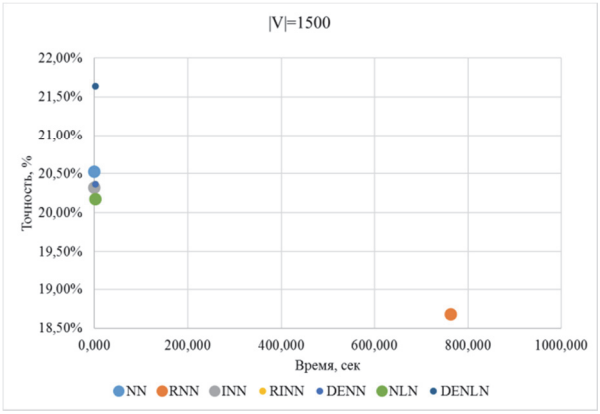

Fig. 9. The results for |V|=1500

The Pareto-Optimal algorithms is shown in Table 3. In Table 4, Table 5, Table 6, Table 7, Table 8, Table 9, Table 10, Table 11, Table 12, Table 13, Table 14, Table 15, Table 16, Table 17 the same results are provided and for both computational time and error rate the standard deviation were calculated. Small values of standard deviation for error rate show that error rate is grouped around the average value. It speaks about the reliability of the results. In tables $min(X)$ – is the minimum of X, $max(X)$ – is the maximum of X, $M(X)$ – is the average of X, $D(X)$ – is the dispersion of X, $\sigma(X)$ – is the standard deviation of X, in last to columns the lower and upper limits of the confidence interval are shown.

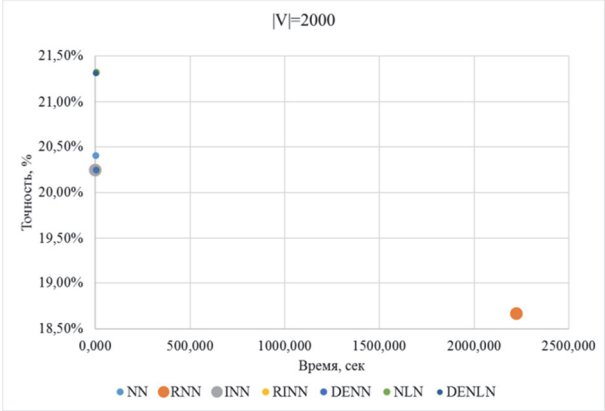

Fig. 10. The results for |V|=2000

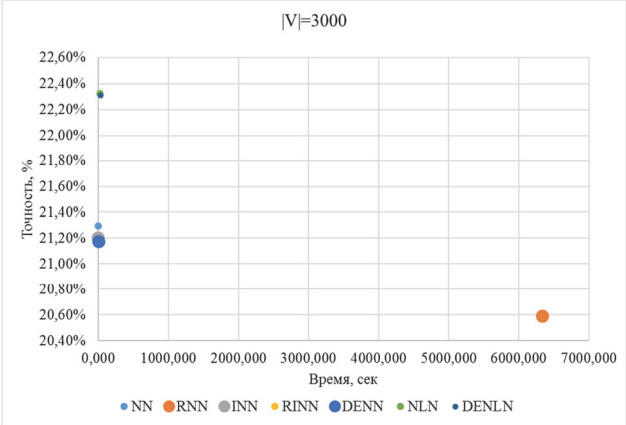

Fig. 11. The results for |V|=3000

The obtained results make it possible to conclude that the proposed approach is applicable to MCPP and gives good results in terms of computational time and error, taking into account the fact, that one of the simplest heuristics was used (the nearest neighbor heuristic).

11 Conclusion

The way of solving the MCPP in multigraph were proposed. Previously, the MCPP were solved only in a mixed *graph*.

The algorithm for transforming the MCPP in multigraph into the equivalent arc routing problem the GTSP were described, implemented in C++ and tested.

The approach to transforming the MCPP in mixed multigraph into equivalent ARP was evaluated on test data from the open databases. Testing has shown that this ap-

proach can be applied to the solve the MCPP in multigraph and should be further investigated with various algorithms for solving the GTSP. In our future work, it is possible to apply and investigated other existing algorithms for GTSP. Next, we are going to develop test data for the mixed multigraph.

Table 4. The Time Characteristics For NN Heuristic

|V|	$\min(T)$	$\max(T)$	$M(T)$	$D(T)$	$\sigma(T)$	$\bar{T} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$	$\bar{T} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$
50	0,001	0,079	0,003	0,000	0,010	0,001	0,006
100	0,001	0,009	0,002	0,000	0,001	0,002	0,002
200	0,002	0,025	0,004	0,000	0,003	0,003	0,005
500	0,007	0,041	0,017	0,000	0,009	0,014	0,021
1000	0,025	0,114	0,062	0,001	0,026	0,052	0,072
1500	0,054	0,264	0,132	0,003	0,056	0,110	0,154
2000	0,098	3,317	0,445	0,428	0,655	0,183	0,707
3000	0,350	3,888	2,510	1,156	1,075	2,080	2,940

Table 5. The Error Characteristics For NN Heuristic

|V|	$\min(L)$	$\max(L)$	$M(L)$	$D(L)$	$\sigma(L)$	$\bar{L} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$	$\bar{L} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$
50	6,65%	23,53%	14,59%	0,12%	3,52%	13,79%	15,40%
100	7,35%	21,25%	15,00%	0,08%	2,84%	14,36%	15,65%
200	8,93%	21,77%	15,24%	0,08%	2,89%	14,58%	15,89%
500	12,18%	35,04%	20,42%	0,38%	6,13%	17,96%	22,87%
1000	12,99%	40,77%	21,33%	0,56%	7,48%	18,34%	24,33%
1500	12,90%	37,38%	20,52%	0,45%	6,69%	17,85%	23,20%
2000	12,28%	39,08%	20,41%	0,46%	6,75%	17,70%	23,11%
3000	12,81%	42,00%	21,29%	0,67%	8,17%	18,03%	24,56%

Table 6. The Time Characteristics For RNN Heuristic

|V|	$\min(T)$	$\max(T)$	$M(T)$	$D(T)$	$\sigma(T)$	$\bar{T} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$	$\bar{T} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$
50	0,011	0,088	0,033	0,000	0,017	0,029	0,037
100	0,079	2,521	0,261	0,081	0,284	0,196	0,325
200	0,630	3,365	1,504	0,420	0,648	1,356	1,652
500	6,678	98,085	31,772	688,350	26,236	21,275	42,269
1000	62,258	695,821	247,218	27517,336	165,884	180,852	313,584
1500	198,014	2009,900	763,293	236970,806	486,796	568,537	958,048
2000	489,153	6731,020	2224,092	2363234,519	1537,282	1609,062	2839,121
3000	4102,820	8901,420	6334,230	2307872,319	1519,168	5726,447	6942,012

Table 7. The Error Characteristics For RNN Heuristic

|V|	$\min(L)$	$\max(L)$	$M(L)$	$D(L)$	$\sigma(L)$	$\bar{L} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$	$\bar{L} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$
50	4,31%	16,66%	9,84%	0,06%	2,54%	9,26%	10,42%
100	6,07%	15,52%	11,07%	0,04%	2,09%	10,60%	11,55%
200	6,72%	18,41%	12,51%	0,06%	2,45%	11,95%	13,07%
500	10,08%	30,40%	17,83%	0,34%	5,84%	15,50%	20,17%
1000	11,65%	37,44%	18,51%	0,43%	6,56%	15,88%	21,13%
1500	11,40%	34,29%	18,68%	0,40%	6,30%	16,16%	21,20%
2000	11,29%	33,46%	18,67%	0,36%	6,03%	16,26%	21,08%
3000	12,53%	38,94%	20,59%	0,58%	7,59%	17,55%	23,62%

Table 8. The Time Characteristics For INN Heuristic

|V|	$\min(T)$	$\max(T)$	$M(T)$	$D(T)$	$\sigma(T)$	$\bar{T} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$	$\bar{T} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$
50	0,000	0,008	0,001	0,000	0,001	0,000	0,001
100	0,000	0,009	0,001	0,000	0,001	0,001	0,002
200	0,002	0,009	0,004	0,000	0,001	0,004	0,004
500	0,011	0,054	0,025	0,000	0,011	0,020	0,030
1000	0,040	0,178	0,095	0,002	0,042	0,078	0,112
1500	0,091	0,395	0,207	0,007	0,086	0,172	0,241
2000	0,155	1,035	0,404	0,047	0,218	0,317	0,492
3000	0,369	21,215	4,478	34,147	5,844	2,140	6,816

Table 9. The Error Characteristics For INN Heuristic

|V|	$\min(L)$	$\max(L)$	$M(L)$	$D(L)$	$\sigma(L)$	$\bar{L} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$	$\bar{L} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$
50	5,81%	25,32%	14,50%	0,12%	3,46%	13,71%	15,29%
100	7,48%	22,43%	14,79%	0,09%	3,04%	14,10%	15,48%
200	8,43%	22,36%	15,27%	0,10%	3,22%	14,54%	16,00%
500	12,57%	35,96%	20,30%	0,36%	6,01%	17,90%	22,71%
1000	12,64%	40,29%	20,63%	0,58%	7,64%	17,58%	23,69%
1500	12,24%	38,45%	20,31%	0,46%	6,78%	17,60%	23,03%
2000	12,09%	35,56%	20,24%	0,41%	6,37%	17,70%	22,79%
3000	12,65%	42,12%	21,20%	0,64%	8,01%	17,99%	24,40%

Table 10. The Time Characteristics For RINN Heuristic

|V|	$\min(T)$	$\max(T)$	$M(T)$	$D(T)$	$\sigma(T)$	$\bar{T} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$	$\bar{T} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$
50	0,001	0,029	0,008	0,000	0,006	0,006	0,009
100	0,004	0,652	0,067	0,008	0,089	0,047	0,088
200	0,050	1,212	0,372	0,064	0,254	0,314	0,430
500	0,018	0,178	0,072	0,002	0,045	0,054	0,090
1000	0,094	1,177	0,446	0,102	0,319	0,318	0,573
1500	0,212	6,003	1,769	2,847	1,687	1,093	2,444
2000	0,556	28,860	6,816	65,243	8,077	3,585	10,048
3000	1,007	114,590	28,321	801,113	28,304	16,997	39,644

Table 11. The Error Characteristics For RINN Heuristic

|V|	$\min(L)$	$\max(L)$	$M(L)$	$D(L)$	$\sigma(L)$	$\bar{L} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$	$\bar{L} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$
50	6,49%	22,54%	14,07%	0,11%	3,32%	13,31%	14,83%
100	7,17%	20,65%	14,86%	0,08%	2,76%	14,23%	15,49%
200	8,65%	23,28%	15,14%	0,09%	2,97%	14,47%	15,82%
500	12,57%	35,96%	20,30%	0,36%	6,01%	17,90%	22,71%
1000	12,64%	40,29%	20,63%	0,58%	7,64%	17,58%	23,69%
1500	12,24%	38,45%	20,36%	0,46%	6,81%	17,64%	23,09%
2000	12,09%	35,56%	20,24%	0,41%	6,37%	17,70%	22,79%
3000	12,65%	42,12%	21,20%	0,64%	8,01%	17,99%	24,40%

Table 12. The Time Characteristics For DENN Heuristic

|V|	$\min(T)$	$\max(T)$	$M(T)$	$D(T)$	$\sigma(T)$	$\bar{T} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$	$\bar{T} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$
50	0,001	0,002	0,001	0,000	0,000	0,001	0,001
100	0,001	0,016	0,002	0,000	0,002	0,002	0,003
200	0,003	0,014	0,006	0,000	0,002	0,005	0,006
500	0,018	0,193	0,074	0,002	0,048	0,055	0,093
1000	0,080	1,151	0,371	0,084	0,290	0,255	0,487
1500	0,171	7,687	1,668	3,726	1,930	0,896	2,441
2000	0,341	12,420	5,143	15,461	3,932	3,570	6,716
3000	8,018	13,786	11,892	2,512	1,585	11,258	12,526

Table 13. The Error Characteristics For DENN Heuristic

|V|	$\min(L)$	$\max(L)$	$M(L)$	$D(L)$	$\sigma(L)$	$\bar{L} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$	$\bar{L} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$
50	9,13%	27,07%	15,53%	0,14%	3,78%	14,67%	16,39%
100	7,26%	22,85%	16,04%	0,12%	3,40%	15,26%	16,82%
200	11,15%	22,18%	16,14%	0,09%	2,92%	15,47%	16,80%
500	12,57%	35,96%	20,30%	0,36%	6,01%	17,90%	22,71%
1000	12,64%	40,29%	20,63%	0,58%	7,64%	17,58%	23,69%
1500	12,24%	38,45%	20,36%	0,46%	6,81%	17,64%	23,09%
2000	12,09%	35,56%	20,24%	0,41%	6,37%	17,70%	22,79%
3000	12,62%	42,24%	21,17%	0,64%	8,00%	17,97%	24,37%

Table 14. The Time Characteristics For NLN Heuristic

|V|	$\min(T)$	$\max(T)$	$M(T)$	$D(T)$	$\sigma(T)$	$\bar{T} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$	$\bar{T} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$
50	0,001	0,029	0,003	0,000	0,003	0,002	0,004
100	0,004	0,046	0,011	0,000	0,006	0,009	0,012
200	0,017	0,078	0,036	0,000	0,013	0,033	0,039
500	0,078	0,434	0,190	0,009	0,097	0,152	0,229
1000	0,293	2,262	0,814	0,265	0,515	0,608	1,020
1500	0,656	15,192	3,043	14,402	3,795	1,525	4,562
2000	0,356	19,953	3,764	20,551	4,533	1,951	5,578
3000	1,456	120,110	27,402	1160,713	34,069	13,772	41,033

Table 15. The Error Characteristics For NLN Heuristic

|V|	$\min(L)$	$\max(L)$	$M(L)$	$D(L)$	$\sigma(L)$	$\bar{L} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$	$\bar{L} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$
50	6,38%	22,21%	14,44%	0,12%	3,49%	13,65%	15,24%
100	8,03%	20,63%	14,76%	0,08%	2,75%	14,13%	15,38%
200	8,45%	21,76%	15,41%	0,10%	3,17%	14,69%	16,13%
500	12,52%	34,46%	20,36%	0,40%	6,35%	17,82%	22,90%
1000	12,65%	40,48%	20,68%	0,57%	7,52%	17,67%	23,69%
1500	12,81%	36,84%	20,17%	0,41%	6,42%	17,60%	22,74%
2000	12,59%	38,07%	21,32%	0,43%	6,57%	18,69%	23,95%
3000	13,34%	39,91%	22,33%	0,68%	8,27%	19,02%	25,63%

Table 16. The Time Characteristics For DENLN Heuristic

|V|	$\min(T)$	$\max(T)$	$M(T)$	$D(T)$	$\sigma(T)$	$\bar{T} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$	$\bar{T} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(T)}{\sqrt{n}}$
50	0,001	0,015	0,003	0,000	0,002	0,002	0,003
100	0,005	0,088	0,012	0,000	0,010	0,010	0,014
200	0,018	0,089	0,039	0,000	0,014	0,036	0,042
500	0,019	0,192	0,058	0,002	0,039	0,043	0,074
1000	0,101	1,081	0,433	0,079	0,281	0,321	0,546
1500	0,299	22,092	2,174	19,057	4,365	0,428	3,921
2000	0,440	28,828	4,095	33,527	5,790	1,779	6,412
3000	1,498	147,260	35,357	2061,828	45,407	17,191	53,523

Table 17. The Error Characteristics For DENLN Heuristic

|V|	$\min(L)$	$\max(L)$	$M(L)$	$D(L)$	$\sigma(L)$	$\bar{L} - Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$	$\bar{L} + Z_{\frac{1-\alpha}{2}}\frac{\sigma(L)}{\sqrt{n}}$
50	5,22%	28,44%	15,82%	0,19%	4,39%	14,82%	16,82%
100	7,65%	22,59%	15,73%	0,10%	3,17%	15,01%	16,46%
200	9,71%	23,78%	15,97%	0,09%	2,97%	15,29%	16,64%
500	13,14%	38,21%	22,05%	0,47%	6,83%	19,32%	24,79%
1000	12,94%	48,55%	22,28%	0,74%	8,58%	18,85%	25,72%
1500	13,31%	39,38%	21,64%	0,46%	6,76%	18,93%	24,34%
2000	12,59%	38,06%	21,31%	0,43%	6,53%	18,70%	23,92%
3000	13,34%	39,90%	22,31%	0,68%	8,26%	19,00%	25,61%

References

1. C. E. Noon and J. C. Bean, "An Efficient Transformation of The Generalized Traveling Salesman Problem," INFOR Information Systems and Operational Research, vol. 31, no. 1, February 1993.
2. H. Thimbleby, "The directed Chinese Postman Problem," Software Practice and Experience, vol. 33, no. 11, pp. 1081-1096, 2003.
3. Mei-Ko Kwan, «Graphic programming using odd or even points» Acta Mathematica Sinica, p. 263–266, 1960.
4. G. Laporte and M. Blais, "Exact Solution of the Generalized Routing Problem through Graph Transformations," Operations Research, vol. 54, no. 8, pp. 906-910, 2003.
5. F. L. Pimentel, «Double-ended nearest and loneliest neighbour–a nearest neighbour heuristic variation for the travelling salesman problem» Revista de Ciências da Computação, т. 6, № 6, 2016.
6. P. Vreda and P. Black, Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology, 2014.
7. G. Laporte, «Modeling and solving several classes of arc routing problems as traveling salesman problems» Computers & operations research, т. 24, № 11, pp. 1057-1061, 1997.
8. S. Bönisch, «Implementierung der Edmonds-Johnson Heuritik für das Mixed Chinese Postman Problem» 21 December 1999.
9. Á. Corberán, "Arc Routing Problems: Data Instances," [Online]. Available: http://www.uv.es/corberan/instancias.htm. [Accessed 3 April 2017].