The Metric Travelling Salesman Problem: The Experiment on Pareto-optimal Algorithms

S.M. Avdoshin <savdoshin@hse.ru>
E.N. Beresneva <katrinberesneva@yandex.ru>
Department of Software Engineering,
National Research University Higher School of Economics,
20, Myasnitskaya st., Moscow, 101000 Russia

Abstract. The Metric Travelling Salesman Problem is a subcase of the Travelling Salesman Problem (TSP), where the triangle inequality holds. It is a key problem in combinatorial optimization. Solutions of the Metric TSP are generally used for costs minimization tasks in logistics, manufacturing, genetics and other fields. Since this problem is NP-hard, heuristic algorithms providing near optimal solutions in polynomial time will be considered instead of the exact ones. The aim of this article is to experimentally find Pareto optimal heuristics for Metric TSP under criteria of error rate and run time efficiency. Two real-life kinds of inputs are intercompared - VLSI Data Sets based on very large scale integration schemes and National TSPs that use geographic coordinates of cities. This paper provides an overview and prior estimates of seventeen heuristic algorithms implemented in C++ and tested on both data sets. The details of the research methodology are provided, the computational scenario is presented. In the course of computational experiments, the comparative figures are obtained and on their basis multi-objective optimization is provided. Overall, the group of Pareto-optimal algorithms for different N consists of some of the MC, SC, NN, DENN, CI, GRD, CI + 2-Opt, GRD + 2-Opt, CHR and LKH heuristics.

Keywords: metric travelling salesman problem, heuristic algorithms, Pareto-optimality

DOI: 10.15514/ISPRAS-2017-29(4)-8

For citation: Avdoshin S.M., Beresneva E.N. The Metric Travelling Salesman Problem: Pareto-optimal Algorithms. *Trudy ISP RAN/Proc. ISP RAS*, vol. 29, issue 4, 2017. pp. 123-138. DOI: 10.15514/ISPRAS-2017-29(4)-8

1. Introduction

The Travelling Salesman Problem (TSP) is one of the most widely known questions in a class of combinatorial optimization problems. Essentially, to meet a challenge of the TSP is to find a Hamiltonian circuit of minimal length. A subcase of the TSP is

Metric TSP where all of the edge costs are symmetric, and they satisfy the triangle inequality.

The methods for solving the TSP have been developed for many years, and since the problem is NP-hard, it continues to be topical. The TSP has seen applications in the areas of logistics, genetics, manufacturing, telecommunications and neuroscience [1]. The most common practical interpretation of the TSP relates to the movement of people and vehicles around tours, such as searching for the shortest tour through N cities, school bus route planning, and postal delivery. In addition, the TSP plays an important role in very large-scale integration (VLSI).

The purpose of this study is to determine the group of Pareto-optimal algorithms among the set of selected ones for Metric TSP by criteria of run time and qualitative performance.

Clearly, a study of this type is inevitably restricted by various constraints, in this research only heuristic algorithms constructing near optimal solutions in polynomial time will be considered instead of the exact ones.

The paper is structured as follows. First, the theoretical basis is described. It presents definition of resource-efficient parameters, Pareto optimization and, at last, the formulation of the aim of the project. Then the description of methods to be used is provided with their prior estimates. After that the details of the research methodology and expected results are mentioned.

2. Theoretical basis

In this paper, mathematical formulation of Metric TSP is adopted as stated here [2].

2.1 Parameters for Pareto-optimality

Let M be a set of selected heuristic algorithms for Metric TSP. There are two parameters of resource-efficiency for $m \in M$ for each number of vertices N in data set:

- $f_\varepsilon(m, N)$ — qualitative performance;
- $f_t(m, N)$ — running time.

Qualitative performance can be calculated using:

$$f_\varepsilon(m, N) = \frac{f(s) - f(s_0)}{f(s_0)} * 100\%,$$

where $f(s)$ is the obtained tour length and $f(s_0)$ is the optimal tour length. The values of optimal tour lengths are taken from the open libraries VLSI Data Sets and National TSPs as the lengths of the best found (exactly) or reported solutions for each of the instances [3] [4].

2.2 Pareto-optimality

The algorithm $m_0 \in M$ is said to be Pareto optimal if $(\forall m \in M)\left((m \neq m_0) \Rightarrow \left(f_\varepsilon(m) > f_\varepsilon(m_0)\right) \vee \left(f_t(m) > f_t(m_0)\right)\right)$.

2.3 The aim of the study

The aim is to find a set $M_0 = \left\{(\forall m \in M)\left((m \neq m_0) \Rightarrow \left(f_\varepsilon(m) > f_\varepsilon(m_0)\right) \vee \left(f_t(m) > f_t(m_0)\right)\right)\right\}$ of Pareto-optimal algorithms for Metric TSP by criteria of time and qualitative performance.

3. Algorithms

Algorithms for solving the TSP may be divided into two classes:

- Exact algorithms, and
- Heuristic (or approximate) algorithms.

Exact algorithms are aimed at finding optimal solutions. However, a major drawback is connected with their time efficiency. It is a common knowledge that there are no exact algorithms running in polynomial time. Thus, only small datasets can be solved in reasonable time. For example, the 4410-vertex problem is believed to be the largest Metric TSP ever solved with respect to optimality [3].

In this paper, some algorithms from a class of heuristic search algorithms will be taken into account. They are designed to run quickly and to get an approximate solution to a given problem.

Heuristic algorithms are subdivided into two groups. The first group includes tour construction algorithms that have one feature in common — the tour is built by adding a new vertex at each step. The second group consists of tour-improving algorithms that, according to Applegate, '…take as input an approximate solution to a problem and attempt to iteratively improve it by moving to new solutions that are close to the original'. Full classification of heuristic algorithms has already been presented in [2].

In order to restrict our investigation, it was decided to choose only three types of tour improving algorithms — the most simple local-optimal method (2-Opt), the most perspective one (LKH) and one of the best swarm intelligence methods — algorithm qCABC based on bee colony agents.

The list of used algorithms for Metric TSP is as follows.

3.1 Nearest Neighbour (NN)

The key to NN is to initially choose a random vertex and to add repeatedly the nearest vertex to the last appended, unless all vertices are used [5].

3.2 Double Ended Nearest Neighbour (DENN)

This algorithm is a modification of NN. Unlike NN, not only the last appended vertex is taken into consideration, so the closest vertex to both of endpoints in the tour is added [6].

3.3 Greedy (GRD)

The Greedy heuristic constructs a path by adding the shortest edge to the tour until a cycle with K edges, $K < N$, is created, or the degree of any vertex exceeds two [7].

3.4 Nearest Addition (NA)

The fundamental idea of NA is to start with an initial subtour made of the shortest edge and to add repeatedly other vertices which are the closest to the vertices being already in the cycle. It should be noted that insertion place is not specially calculated. It is always added after the nearest vertex in the cycle. Algorithm is terminated when all vertices are used and inserted in the tour.

3.5 Nearest Insertion (NI), Cheapest Insertion (CI), Farthest Insertion (FI), Arbitrary Insertion (AI), Nearest Segment Insertion (NSI)

The start step of these algorithms is similar to NA (except for FI, where the longest edge is found). Next, other vertices are added repeatedly using various rules. Depending on the algorithm the vertex not yet in the cycle should be inserted so that:

- In NI it is the closest to any node in the tour;
- In CI its addition to the tour gives a minor increment of its length;
- In FI it is the farthest to any node in the cycle;
- IN AI it is the random vertex not yet in the cycle;
- In NSI distance between the node and any edge in the tour is minimal.

The previous step should be repeated until all vertices are added to the cycle.

The feature of these methods is additional computation that selects the best place for each inserting node [6] [8].

3.6 Double Minimum Spanning Tree (DMST)

DMST method is based on the construction of a minimal spanning tree (MST) from the set of all vertices. After MST is built, the edges are doubled in order to obtain an Eulerian cycle, containing each vertex at least once. Finally, a Hamiltonian circuit is made from an Eulerian circuit by sequential (or greedy) removing occurrences of each node [9].

3.7 Double Minimum Spanning Tree Modified (DMST-M)

This algorithm is a modification of DMST. Unlike DMST, it is necessary to remove duplicate nodes from an Eulerian cycle using triangle inequality instead of greedy method.

3.8 Christofides (CHR)

This method is a modification of DMST that was proposed by Christofides [10]. The difference between CHR and DMST is addition of minimum weight matching calculation to the first algorithm.

3.9 Moore Curve (MC)

This is a recursive geometric method. Vertices are sorted by the order they are located on the plane. Only the two-dimensional example of Moore curve is implemented [11]. Figure 1 shows the order of the cells after one, two and three subdivision steps respectively [11].

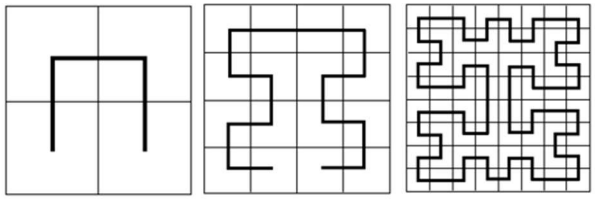


Fig. 1. The order for the Moore curve after 1, 2 and 3 subdivision steps

3.10 Sierpinski Curve (SC)

This algorithm is also included in the family of Space-Filling Curves combinatorial algorithms as MC. SC is more symmetric than MC [12]. Figure 2 shows the order of the cells after one, two and three subdivision steps respectively.

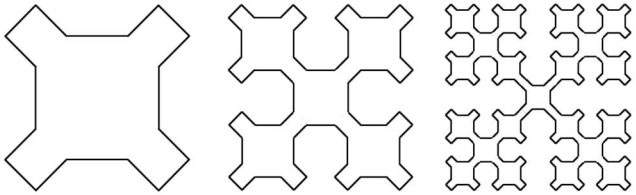


Fig. 2. The order for the Sierpinski curve after 1, 2 and 3 subdivision steps

3.11 2-Opt

The main idea behind 2-Opt is to take a tour that has one or more self-intersections and to remove them repeatedly. In mathematical terms, edges ab and cd should be

deleted and new edges ac and bd should be inserted, if $d(a,b) + d(c,d) > d(a,c) + d(b,d)$ [13].

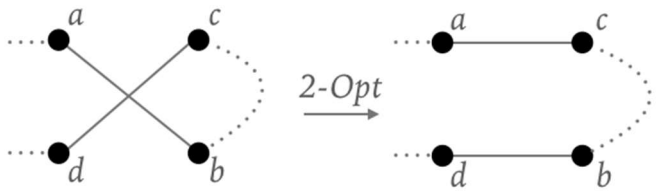


Fig. 3. 2-Opt modification

3.12 Helsgaun's Lin and Kernighan Heuristic (LKH)

LKH uses the principle of 2-Opt algorithm and generalizes it. In this heuristic, the k-Opt, where $k = \overline{2..\sqrt{N}}$, is applied, so the switches of two or more edges are made in order to improve the tour. This method is adaptive, so decision about how many edges should be replaced is taken at each step [14].

It should be noted that because of complexity of LKH algorithm, it was not implemented by the authors of research. The original open source code [15] was used to carry out experiments. All the parameters were not changed, so they were used by default.

3.13 Quick Combinatorial Artificial Bee Colony (qCABC).

This is one of the Swarm Intelligence methods, which is based on colony of bees. Algorithm suggests that all agents are divided into the three groups: scout bees (looking for new random solutions), employed bees (keeping information and sharing it) and onlooker bees (choosing the solution to explore) [16].

3.14 Estimates

Estimated upper bounds for the algorithms can be calculated as are the ratio of $\frac{f(s)}{f(s_0)}$.

According to [1], for any k-Opt algorithm, where $k \leq N/4$, problems may be constructed such that the error is almost 100%. So 2-Opt and LKH algorithms have approximate upper bound 2. Upper-bound estimates and running times of the algorithms are represented in Table 1.

Table 1. Upper-bound estimates and running time of algorithms

#	Algorithm	Upper-bound estimate	Running time
1	NN	$0.5\lceil \log_2 N + 1 \rceil$	$O(N^2)$
2	DENN		
3	GRD		$O(N^2 \log N)$
4	NA	$2 - \frac{2}{N}$	$O(N^2)$
5	NI		
6	CI		$O(N^2 \log N)$
7	FI		$O(N^2)$
8	AI		
9	NSI		
10	DMST		
11	DMST-M		
12	CHR	$\frac{3}{2} - \frac{1}{N}$	$O(N^3)$
13	2-Opt	≈ 2	$O(N^2)$
14	LKH		$O(N^{2.2})$
15	MC	$\log N$	$O(N \log N)$
16	SC		
17	qCABC	?	$O(N^2)$

4. Experimental research

This section documents details of the research methodology. The experiment is carried out on a 1.3 GHz Intel Core i5 MacBook Air. It includes the qualitative performance and the run time efficiency of the current implementations.

Heuristics are implemented in C++. Two types of data bases from an open library TSPLIB are selected. The first one is VLSI data sets [3]. There are 102 instances in the VLSI collection that range in size from 131 vertices up to 744,710 vertices. All of these instances are tested. The first dataset is National TSPs, which includes 25 instances that vary from 29 to 71009 points [4].

There is one data set for each number of vertices for all input data. The integer Euclidean metric distance is used, so coordinates of nodes and distances between them have integer values. The distance *d* between some nodes *v* and *w* is calculated as follows:

$$d(v,w) = \left\lfloor \sqrt{|x(v) - x(w)|^2 + |y(v) - y(w)|^2} + 0.5 \right\rfloor$$

The computational experiment corresponds to the following scenario:

```
Input: Algorithms, input datasets (VLSI Data Sets, National TSPs)
    1:  foreach tour construction and composite algorithm m
    2:      foreach tour improving algorithm m'
    3:          foreach dataset type DT from input datasets
    4:              foreach dataset D form DT
```



```
    5:                  for i ∈ {1…11} // tour construction stage
    6:                      solution s = run algorithm m on D
    7:                      if (i > 1)
    8:                          calculate f_εi(m,D), f_ti(m,D)
    9:                  calculate f_εmin(m,D)
   10:              remember best solution s0
   11:              calculate σ(Σ_{i=1}^{10} f_ti(m,D))
   12:              calculate f_tavg(m,D) = (f_t1(m,D)+…+f_t10(m,D))/10
   13:              if (m is composite) continue
   14:              for i ∈ {1…11} // improvement stage on s0
   15:                  solution s = run algorithm m' on D
   16:                  if (i > 1)
   17:                      calculate f_εi(s0+m',D), f_ti(s0+m',D)
   18:              calculate f_εmin(s0+m',D)
   19:              calculate σ(Σ_{i=1}^{10} f_ti(s0+m',D))
   20:              calculate f_tavg(s0+m',D) = (f_t1(s0+m',D)+…+f_t10(s0+m',D))/10
   21:          calculate E(f_εmin(m,D)), E(f_εmin(s0+m',D)) for all D
   22:          calculate σ(f_εmin(m,D)), σ(f_εmin(s0+m',D)) for all D
   23:          calculate max(f_εmin(m,D)), max(f_εmin(s0+m',D)) for all D
   24:          calculate min(f_εmin(m,D)), min(f_εmin(s0+m',D)) for all D
```

Fig. 3. Computational scenario

Metrics used in scenario have following meanings:

- $f_{\varepsilon_i}(m,D)$ — qualitative performance of m (one iteration),
- $f_{t_i}(m,D)$ — running time of m (one iteration),
- $f_{\varepsilon_{min}}(m,N)$ — best qualitative performance of m,
- $f_{t_{avg}}(m,N)$ — average running time of m (sec),
- $\sigma\left(\sum_{i=1}^{10} f_{t_i}(m,D)\right)$ — standard deviation of running time estimates through 10 iterative runs,
- $E\left(f_{\varepsilon_{min}}(m,N)\right)$ — expected value of qualitative performance of m for one DT,
- $\sigma\left(f_{\varepsilon_{min}}(m,N)\right)$ — standard deviation of qualitative performance of m for one DT,
- $max\left(f_{\varepsilon_{min}}(m,N)\right), min\left(f_{\varepsilon_{min}}(m,N)\right)$ — maximum and minimum values of qualitative performance of m for one DT.

Qualitative performance metrics are represented in Table 2. Table color scheme varies from green (the best result in a column) to red (the worst value in a column).

Table 2. Running time of algorithms

Algorithms	$E(f_\varepsilon)$	max f_ε	min f_ε	$\sigma(f_\varepsilon)$
LKH	0,08%	0,23%	0,00%	0,07%
CHR + 2-Opt	6,14%	12,14%	3,47%	1,59%
GRD + 2-Opt	6,79%	10,82%	4,69%	1,57%
DENN + 2-Opt	11,06%	22,26%	4,39%	5,30%
NN + 2-Opt	11,89%	24,91%	3,90%	2,36%
CHR	12,60%	16,82%	9,31%	1,41%
CI + 2-Opt	13,04%	21,86%	6,74%	2,83%
NI + 2-Opt	14,60%	29,66%	5,86%	6,33%
DMST-M + 2-Opt	16,08%	35,78%	4,80%	9,29%
GRD	17,31%	31,34%	10,30%	3,83%
NSI + 2-Opt	17,63%	33,65%	8,92%	6,87%
DMST + 2-Opt	19,08%	39,12%	6,91%	10,52%
CI	20,28%	25,05%	12,46%	1,96%
DENN	22,82%	33,38%	11,97%	2,47%
NN	25,38%	32,68%	13,94%	2,62%
NA + 2-Opt	27,04%	57,90%	6,79%	17,84%
NI	28,07%	35,29%	14,89%	2,87%
FI + 2-Opt	28,90%	58,05%	4,01%	17,68%
DMST-M	32,46%	41,68%	18,55%	4,32%
SC + 2-Opt	36,20%	166,45%	8,11%	31,69%
NSI	36,23%	48,17%	19,15%	5,46%
MC + 2-Opt	36,47%	177,83%	6,21%	39,70%
DMST	40,09%	48,88%	33,16%	3,07%
AI + 2-Opt	50,23%	77,85%	5,26%	23,43%
NA	51,30%	59,23%	35,38%	4,67%
FI	56,88%	66,09%	31,59%	5,98%
MC	64,49%	242,41%	33,07%	41,08%
SC	66,16%	246,64%	30,76%	42,13%
AI	85,20%	100,92%	65,78%	6,81%

The time limit on algorithm's running time is introduced. It is 11 800 seconds ≈ 3 hours and 20 minutes, at the maximum. That means computational time for one experiment cannot exceed 11 800 seconds * 11 runs ≈ 36 hours ≈ 1.5 days.

5. Results

Experimental results showed that algorithm qCABC takes a large amount of time (more than 'the slowest' CHR) and gives improvement in accuracy even less than

'the most rough' 2-Opt. So qCABC as tour improving algorithm is admitted to be "unviable".

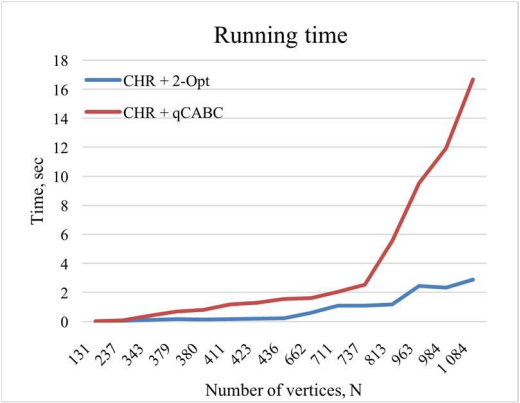


Fig. 4. Running time comparison of CHR + 2-Opt and CHR + qCABC algorithms.

We decided to select 10 pairs of data sets from VLSI and National TSPs with similar number of vertices (see Table 3) to plot charts that illustrate Paretos.

Table 3. Pairs of input datasets from VLSI and National TSPs

	VLSI	National TSP
Number of vertices, N	737	734
	984	980
	1 973	1 979
	3 386	3 496
	7 168	7 146
	10150	9 976
	14 233	14 185
	16 928	16 862
	22 777	22 775
	33 203	33 708

The charts for pair with $N = 22\,775$ and $N = 22\,777$ are shown below (see Fig. 5, Fig. 6, Fig. 7, Fig. 8). The name of each TSPLIB instance is shown in chart title. The horizontal axis represents the time performance of methods in seconds. The vertical axis shows the gap between optimal and obtained solutions, expressed in percent. Pareto-optimal methods are highlighted in red. The points which are represented by Pareto solutions are bigger than non-Pareto-optimal solutions.

There are two charts (see Fig. 6, Fig. 8) where not all algorithms are compared. These auxiliary charts are enlarged copies of their originals. Their role is to graphically illustrate Pareto-optimal algorithms at scale-up.

Results on VLSI Data sets only are reported in more detail in [2].

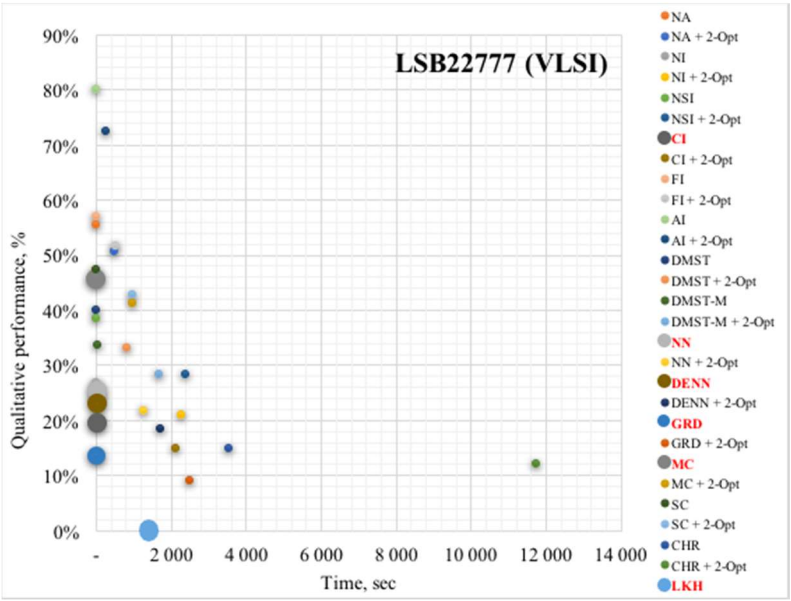


Fig. 5. Pareto-optimal algorithms for LSB22777.tsp (N = 22777)

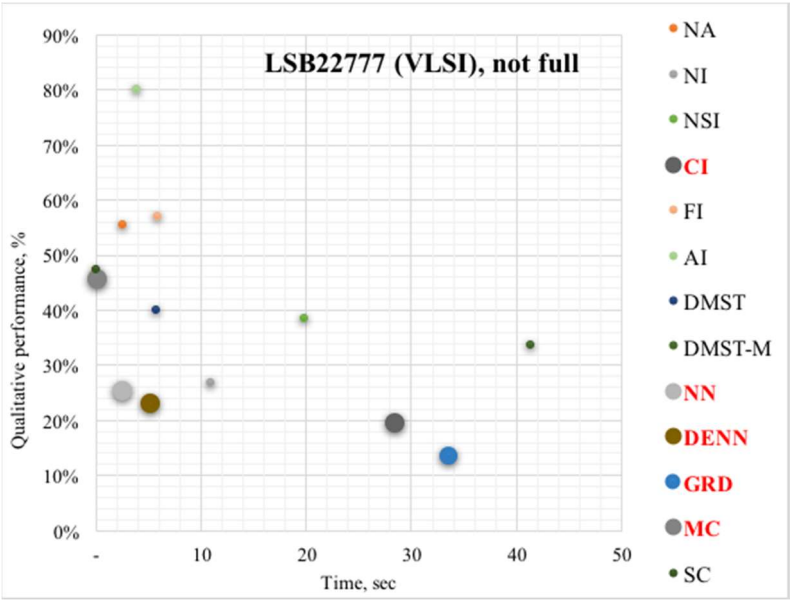


Fig. 6. Pareto-optimal algorithms for LSB22777.tsp (N = 22777), scaled-up

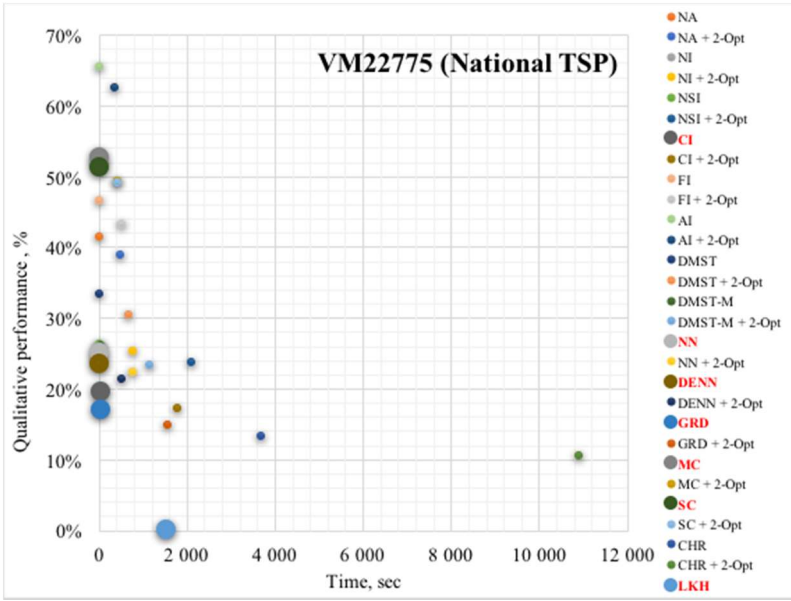


Fig. 7. Pareto-optimal algorithms for VM22775.tsp (N = 22775)

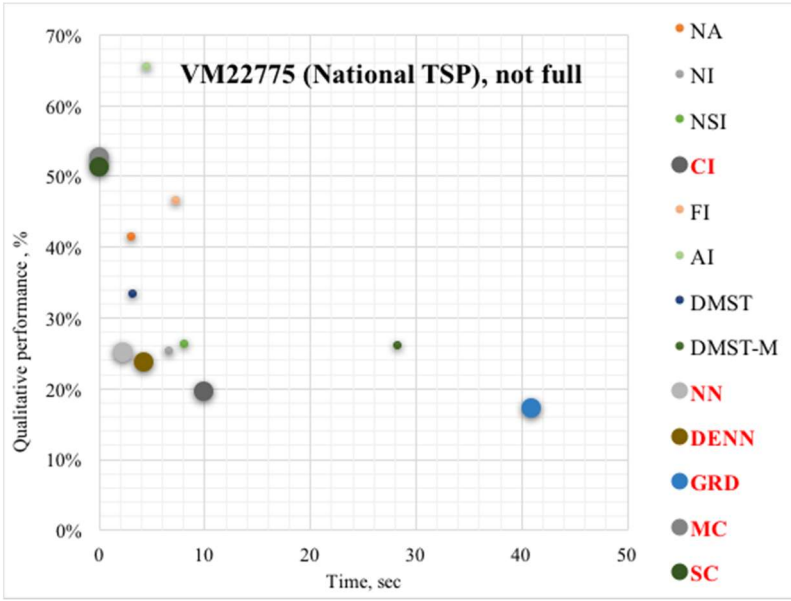


Fig. 8. Pareto-optimal algorithms for VM22775.tsp (N = 22775), scaled-up

Pareto-optimal solutions, that can be suggested on the basis of both data sets only, are shown in Table 4 and they are sorted in the order of increase of running time:

- Moore Curve (MC);
- Sierpinski Curve (SC) — this algorithm depends on type of input data, so qualitative performance estimates are unstable;
- Nearest Neighbour (NN);
- Double Ended Nearest Neighbour (DENN);
- Cheapest Insertion (CI) is Pareto-optimal if $N \lesssim 400\,000$ because of introduced time limit; if $N \lesssim 3\,500$ CI's behavior fluctuates;
- Greedy (GRD) — is Pareto-optimal if $N \lesssim 30\,000$ because of memory limits — $\frac{N(N-1)}{2}$ pairs of edges are needed to be kept simultaneously ;
- Cheapest Insertion and 2-Opt (CI + 2-Opt) — is Pareto-optimal if $30\,000 \lesssim N \lesssim 100\,000$;
- Greedy and 2-Opt (GRD + 2-Opt) — is Pareto-optimal if $N \lesssim 800$;
- Christofides (CHR) — is Pareto-optimal if $N \lesssim 2\,000$;
- Helsgaun's Lin and Kernighan Heuristic (LKH) — this algorithm works excellent if $N \lesssim 55\,000$, however if input data size exceeds $55\,000$ than time limit is met.

Table 4. Pairs of input datasets from VLSI and National TSPs

Algorithm	Number of vertices, *N* (thousands)							
	(0; 0.8)	[0.8; 2)	[2; 3.5)	[3.5; 30)	[30; 55)	[55; 100)	[100; 400)	[400; 700)
MC	+	+	+	+	+	+	+	+
SC	±	±	±	±	±	±	±	±
NN	+	+	+	+	+	+	+	+
DENN	+	+	+	+	+	+	+	+
CI	±	±	±	+	+	+	+	
GRD	+	+	+	+				
CI + 2-Opt					+	+		
GRD + 2-Opt	+							
CHR	+	+						
LKH	+	+	+	+	+			

The "+" sign means that the algorithm in the same row is supposed to be Pareto-optimal at the range of vertices defined in the same column. The "±" sign shows that experiments did not clearly define if it is Pareto-optimal or not.

6. Conclusion

The presented study is undertaken to determine what heuristics for Metric TSP should be used in specific circumstances with limited resources.

This paper provides an overview of seventeen heuristic algorithms implemented in C++ and tested on both the VLSI data set and instances of National TSPs. In the

course of computational experiments, the comparative figures are obtained and on their basis multi-objective optimization is provided. Overall, the group of Pareto-optimal algorithms for different N consists of some of the MC, SC, NN, DENN, CI, GRD, CI + 2-Opt, GRD + 2-Opt, CHR and LKH heuristics.

In our future work, we are going to fine-tune parameters of LKH method using genetic algorithms of search optimization. Further, it is possible to increase the number of heuristic algorithms, to transit to other types of test data and to conduct experiments using different metrics in order to ensure that a Pareto optimal group is sustainable.

The practical applicability of our findings is to present Pareto optimal algorithms that lead to solutions with maximum accuracy under the given resource limitations. The results can be used for scientific purposes by other researchers and for cost minimization tasks.

References

[1]. Applegate, D., Bixby, R., Chvatal, V., Cook, W. The Traveling Salesman Problem: A Computational Study, Princeton: Princeton University Press, 2011.

[2]. Beresneva (Chirkova), E.N., Avdoshin, S.M. Pareto-optimal Algorithms for Metric TSP: Experimental Research. International Journal of Open Information Technologies , vol. 5, № 5, pp. 16-24, 2017, ISSN: 2307-8162.

[3]. Department of Combinatorics and Optimization at the University of Waterloo. Status of VLSI Data Sets. University of Waterloo (web-site). Available at: http://www.math.uwaterloo.ca/tsp/vlsi/summary.html, accessed 29.04.2017.

[4]. University of Waterloo. National Travelling Salesman Problems. UWaterloo, (web-site). Available at: http://www.math.uwaterloo.ca/tsp/world/countries.html, accessed 16.04.2017.

[5]. Flood, M.M. The traveling-salesman problem. Operation research, vol. 4, pp. 61-75, 1956.

[6]. Rosenkrantz, D. Stearns, R., Lewis II, P. An analysis of several heuristics for the traveling salesman problem, vol. 6, pp. 563-581, 1977.

[7]. Cook, W.J. Combinatorial optimization, New York: Wiley, 1998.

[8]. Hahsler, M., Hornik, K. TSP — Infrastructure for the Traveling Salesperson Problem, vol. 23, № 2, 2007.

[9]. Christofides, N. Graph theory — An Algorithmic Approach, New York: Academic Press, 1974.

[10]. Christofides, N. Worst-case analysis of a new heuristic for the travelling salesman problem. Graduate School of Industrial Administration, CMU, 1976.

[11]. Buchin, K. Space-Filling Curves. Organizing Points Sets: Space-Filling Curves, Delaunay Tessellations of Random Point Sets, and Flow Complexes, Berlin, Free University of Berlin, 2007, pp. 5-30.

[12]. Bartholdi, J., Platzman, L., Collins R., Warden, W. A Minimal Technology Routing System for Meals on Wheels, vol. 13, № 3, pp. 1-8, 1983.

[13]. Aarts, E., Lenstra, J. Local Search in Combinatorial Optimization, Princeton, New Jersey: Princeton University Press, 2003.

[14]. Helsgaun, K. An effective implementation of the Lin–Kernighan traveling salesman heuristic, EJOR, vol. 12, pp. 106-130, 2000.

[15]. Helsgaun, K. LKH (web-site). Available at: http://www.akira.ruc.dk/~keld/research/LKH, accessed 24.02.2017.
[16]. Gorkemli, B., Karaboga, D. Quick Combinatorial Artificial Bee Colony -qCABC-Optimization Algorithm for TSP, vol. 1, № 5, 2013.

Метрическая задача коммивояжера: экспериментальное исследование Парето-оптимальных алгоритмов

С.М. Авдошин <savdoshin@hse.ru>
Е.Н. Береснева <katrinberesneva@yandex.ru>
Департамент программной инженерии,
Национальный исследовательский университет "Высшая школа экономики",
101000, Россия, г. Москва, ул. Мясницкая, д. 20.

Аннотация. Задача коммивояжера — одна из важнейших задач теории графов и комбинаторной оптимизации, суть которой состоит в нахождении гамильтонова цикла наименьшей длины. Разработка методов для решения задачи коммивояжера осуществляется на протяжении многих лет, и, по-прежнему, остается актуальной, поскольку задача является NP-трудной. Решения применяются, в основном, для минимизации производственных и логистических затрат и издержек. В работе рассматривается частный случай общей постановки задачи коммивояжера, в котором выполняется свойство метрики — метрическая задача коммивояжера. Целью данной работы является определение группы Парето оптимальных алгоритмов решения метрической задачи коммивояжера по критериям времени работы и точности решения в ходе экспериментального исследования. В связи с тем, что задача коммивояжера является NP-трудной, в работе рассматриваются только эвристические алгоритмы, позволяющие получить приближенные решения за полиномиальное время. В статье представлено краткое описание используемых алгоритмов решения метрической задачи коммивояжера, указаны их априорные точностные и временные оценки. Приведено описание плана эксперимента. Данными для экспериментального исследования послужили два набора из открытой библиотеки данных для метрической задачи коммивояжера, основанные на высоко-интегральных вычислительных схемах (VLSI Data Sets) и географических координатах (высоте и широте) городов в различных странах (National TSPs). В результате исследований выявлены оптимальные по Парето алгоритмы для наборов данных различных размерностей — до 700 тысяч вершин. Для каждого N в число Парето-оптимальных алгоритмов входят некоторые из алгоритмов MC, SC, NN, DENN, CI, GRD, CI + 2-Opt, GRD + 2-Opt, CHR и LKH. Приведена таблица, содержащая информацию о результатах экспериментов.

Ключевые слова: метрическая задача коммивояжера, эвристический алгоритм, оптимальность по Парето.

DOI: 10.15514/ISPRAS-2017-29(4)-8

Для цитирования: Авдошин С.М., Береснева Е.Н. Метрическая задача коммивояжера: экспериментальное исследование Парето-оптимальных алгоритмов. *Труды ИСП РАН*, том 29, вып. 4, 2017 г. стр. 123-138 (на английском языке). DOI: 10.15514/ISPRAS-2017-29(4)-8

Список литературы

[1]. Applegate, D., Bixby, R., Chvatal, V., Cook, W. The Traveling Salesman Problem: A Computational Study, Princeton: Princeton University Press, 2011.
[2]. Beresneva (Chirkova), E.N., Avdoshin, S.M. Pareto-optimal Algorithms for Metric TSP: Experimental Research. International Journal of Open Information Technologies , 5, № 5, pp. 16-24, 2017, ISSN: 2307-8162.
[3]. Department of Combinatorics and Optimization at the University of Waterloo. Status of VLSI Data Sets. University of Waterloo (web-site). Доступно по ссылке: http://www.math.uwaterloo.ca/tsp/vlsi/summary.html, дата обращения 29.04.2017.
[4]. University of Waterloo. National Travelling Salesman Problems. UWaterloo, (web-site). Доступно по ссылке: http://www.math.uwaterloo.ca/tsp/world/countries.html, дата обращения 16.04.2017.
[5]. Flood, M.M. The traveling-salesman problem. Operation research, vol. 4, pp. 61-75, 1956.
[6]. Rosenkrantz, D. Stearns, R., Lewis II, P. An analysis of several heuristics for the traveling salesman problem, 6, pp. 563-581, 1977.
[7]. Cook, W.J. Combinatorial optimization, New York: Wiley, 1998.
[8]. Hahsler, M., Hornik, K. TSP — Infrastructure for the Traveling Salesperson Problem, 23, № 2, 2007.
[9]. Christofides, N. Graph theory — An Algorithmic Approach, New York: Academic Press, 1974.
[10]. Christofides, N. Worst-case analysis of a new heuristic for the travelling salesman problem. Graduate School of Industrial Administration, CMU, 1976.
[11]. Buchin, K. Space-Filling Curves. Organizing Points Sets: Space-Filling Curves, Delaunay Tessellations of Random Point Sets, and Flow Complexes, Berlin, Free University of Berlin, 2007, pp. 5-30.
[12]. Bartholdi, J., Platzman, L., Collins R., Warden, W. A Minimal Technology Routing System for Meals on Wheels, 13, № 3, pp. 1-8, 1983.
[13]. Aarts, E., Lenstra, J. Local Search in Combinatorial Optimization, Princeton, New Jersey: Princeton University Press, 2003.
[14]. Helsgaun, K. An effective implementation of the Lin–Kernighan traveling salesman heuristic, EJOR, 12, pp. 106-130, 2000.
[15]. Helsgaun, K. LKH (web-site). Доступно по ссылке: http://www.akira.ruc.dk/~keld/research/LKH, дата обращения 24.02.2017.
[16]. Gorkemli, B., Karaboga, D. Quick Combinatorial Artificial Bee Colony -qCABC-Optimization Algorithm for TSP, 1, № 5, 2013.