The metric travelling salesman problem: pareto-optimal heuristic algorithms

Ekaterina Beresneva

Faculty of Computer Science
National Research University Higher School of Economics
Moscow, Russia, +7(925)538-40-58
eberesneva@hse.ru

Sergey M. Avdoshin

Software Engineering School
National Research University Higher School of Economics
Moscow, Russia, +7(985)695-22-75
savdoshin@hse.ru

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 and genetics. Since this problem is NP-hard, heuristic algorithms providing near optimal solutions in polynomial time will be considered. The aim of this article is to 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. There is a classification of algorithms for Metric TSP in the article. Feasible heuristics and their prior estimates are described. The details of the research methodology are provided. Finally, results and prospective research are discussed.

Keywords: pareto-optimal algorithms, metric travelling salesman problem, heuristic algorithms.

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) [2].

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. Next, a classification of algorithms for Metric TSP is given, including a literature review of popular heuristics. 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

2.1 Resource-efficient parameters

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 (1):

$$f_\varepsilon(m, N) = \frac{f(s) - f(s_0)}{f(s_0)} * 100\%, \quad (1)$$

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 Pareto optimal if $(\forall m \in M)$ $\Big((m \neq m_0) \Rightarrow \big(f_\varepsilon(m) > f_\varepsilon(m_0)\big) \vee \big(f_t(m) > f_t(m_0)\big)\Big)$.

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 Related works

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. Both widely known subtypes of exact methods – linear programming and branch-and-bound techniques – are described in details by Applegate [1]. 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 [5].

In this paper, only 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 following are alternative ways of doing this [6]:

- Ordered sequence methods.
  At the start of the heuristic, all the edges are ranked by some suitable criteria. At each iteration the best edge according to the criteria is added to the solution. The most common order sequence methods are Greedy [7] and Savings [8] [9].
- Increasing path methods.
  One node or edge is selected as the starting point of a path. At each iteration one edge is selected according to some criterion, and added to one of the ends of the path. A disadvantage of this approach is that it is possible for the ends of the path to be 'far' from one another when the heuristic terminates, i.e. the last edge needed to change the path to a tour may be an expensive edge. There are two groups of increasing path methods – neighbour group [10] [11] and space-filling curves group [12].
- Subtour insertion methods.
  An initial subtour is selected, for example one edge is selected. At each iteration, a node is selected and it is inserted into the subtour according to some criterion. In order to insert a new node, some edge in the subtour is replaced by two edges not in the subtour. The *insertion* heuristics are described by Rosenkrantz [13], Johnson and McGeoch [14].

- Combined methods.
  These are well-known algorithms based on minimum spanning tree that were introduced by Christofides [15] [16].

The second group consists of *tour-improving algorithms* that have their roots in TSP papers from the 1950s [17] [18]. According to Applegate, *'... These heuristics 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'* [1]. Most of these algorithms are described by Aarts and Lenstra [19].

Tour-improving algorithms consists of:

- Local-optimal methods.
  Such algorithms are aimed at removing one set of edges and inserting the other set. Currently, the most simple tour-improving heuristic used in practice is 2-Opt heuristic, where 2 edges are replaced [20]. It was introduced and described by Flood [21], Croes [18] and Bock [17]. The later algorithm of Lin and Kernighan (LK) appeared on the basis of k-Opt tour-finding approach [22]. Nowadays, the most "successful" local-optimal method is Helsgaun's modification of LK – LKH [23].
- Simulated annealing methods:
  Simulated annealing algorithms are based on imitation of physical process of metal annealing. These methods propose a periodical increase in tour length in order to prevent getting in local optimums. Detailed description is presented by Laarhoven etc [24].
- Evolutionary methods:
  These methods are inspired by the process of biological evolution. They include the following parts: reproduction, mutation, crossover (recombination) and selection. More information can be found in [14].
- Swarm intelligence methods:
  These algorithms are nature-inspired metaheuristics. They are based on decentralized, self-organized systems with simple agents, such as ants, bees, birds, fishes etc. The most popular swarm intelligence methods are Artificial Bee Colony algorithm [25], Ant Colony Optimization [26], Particle Swarm Optimization [27] and Cockoo Search [28].

4 Algorithms

In this paper, most of the methods mentioned in Part III are implemented and assessed through experiments.

It should be noted that Savings algorithm was not chosen because it was mentioned in [1] that for most instances Greedy algorithm gives better results than Savings.

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 following:

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 [21].

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 [10].

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 [29].

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.

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:

a) In NI it is the closest to any node in the tour;
b) In CI its addition to the tour gives a minor increment of its length;
c) In FI it is the farthest to any node in the cycle;
d) IN AI it is the random vertex not yet in the cycle;
e) 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 [13] [30].

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 [31].

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.

8) Christofides (CHR).

This method is a modification of DMST that was proposed by Christofides [15]. The difference between CHR and DMST is addition of minimum weight matching calculation to the first algorithm.

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 [32].

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 [33].

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 $d(a,b) + d(c,d) > d(a,c) + d(b,d)$ [19].

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 [23].

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 [34] was used to carry out experiments. All the parameters were not changed, so they were used by default.

13) Quick Combinatorial Artificial Bee Colony (qCABC).

This is one of the Swarm Intelligence methods, which are 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) [25].

Estimated upper bounds for the algorithms can be calculated as are the ratio of $\frac{f(s)}{f(s_0)}$ (see Table 1). According to [13], 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. Running times of the algorithms are represented in Table 2.

Table 1. Upper-bound estimates of algorithms

#	Algorithm	Upper-bound estimate
	NN DENN	$0.5\lceil \log_2 N + 1 \rceil$
	GRD	$0.5\lceil \log_2 N + 1 \rceil$
	NA, NI, CI, FI, AI, NSI	$2 - \frac{2}{N}$
	DMST, DMST-M	$2 - \frac{2}{N}$
	CHR	$\frac{3}{2}$
	2-Opt	≈ 2
	LKH	≈ 2
	MC, SC	$\log N$
	qCABC	?

Table 2. Running time of algorithms

#	Algorithm	Running time
	NN DENN	$O(N^2)$
	GRD	$O(N^2 \log N)$
	NA, NI, CI, FI, AI, NSI	$O(N^2)$
	DMST, DMST-M	$O(N^2)$
	CHR	$O(N^3)$
	2-Opt	$O(N^2)$
	LKH	$O(N^{2,2})$
	MC, SC	$O(N \log N)$
	qCABC	$O(N^3)$

5 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 [2]. There are 102 instances in the VLSI collection that range in size from 131 vertices up to 744,710 vertices. 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 datasets. The integer Euclidean metric distance is used, so coordinates of nodes and distances between them have integer values, thus without loss of generality (4) is transformed into:

$$d_1(v,w) = \left\lfloor \sqrt{|x(v) - x(w)|^2 + |y(v) - y(w)|^2} + \ 0.5 \right\rfloor$$

Experiments showed that LKH algorithm is both tour construction and tour improving because despite using different types of initial subtours this algorithm gives the same results. That is why LKH is represented as an individual algorithm.

Experimental results showed that algorithm qCABC takes large amount of time (more than CHR) and gives improvement in accuracy even less than 2-Opt. So qCABC is admitted to be “unviable”.

6 Pareto-optimal 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	1979
	3 386	3496
	7 168	7146
	10150	9976
	14 233	14185
	16 928	16862
	22 777	22775
	33 203	33708

The charts for pair with $N \approx 10\,000$ are shown below (see 1). The name of 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.

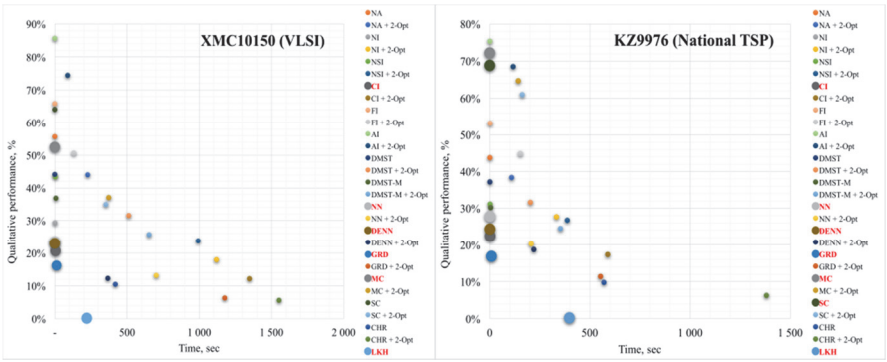

Fig. 1. Pareto-optimal algorithms for XMC10150.tsp (N = 10150) and KZ9976.tsp (N = 9976).

Pareto-optimal solution, which can be suggested on the basis of both data sets only, are:

Algo-rithm	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	+	+	+	+	+			

7 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 eleven heuristic algorithms implemented in C++ and tested on the VLSI data set. In the course of computational experiments, the comparative figures are obtained and on their basis multi-objective optimization is provided. Overall, the generalized group of Pareto-optimal algorithms for all *N* consists of 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. D. L. Applegate, The Traveling Salesman Problem, Princeton: Princeton University Press, 2006.
2. Heidelberg University, "TSPLIB," [Online]. Available: https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. [Accessed 01 02 2017].

3. Department of Combinatorics and Optimization at the University of Waterloo, "Status of VLSI Data Sets," University of Waterloo, [Online]. Available: http://www.math.uwaterloo.ca/tsp/vlsi/summary.html. [Accessed 29 04 2017].
4. University of Waterloo, "National Travelling Salesman Problems," UWaterloo, [Online]. Available: http://www.math.uwaterloo.ca/tsp/world/countries.html. [Accessed 16 04 2017].
5. University of Waterloo, "Status of VLSI Data Sets," [Online]. Available: http://www.math.uwaterloo.ca/tsp/vlsi/summary.html. [Accessed 08 02 2017].
6. B. Hock, "An examination of heuristic algorithms for the Travelling Salesman problem," University of Cape Town, Cape Town, 1988.
7. M. Fisher, G. Nemhauser and L. Wolsey, An analysis of approximations for maximizing submodular set functions, Springer, 1978.
8. G. Clarke and J. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, no. 12, pp. 568-581, 1964.
9. B. L. Golden, T. Magnanti and H. Nguyen, "Implementing vehicle routing algorithms," Networks, vol. 7, no. 2, pp. 113-148, 1977.
10. D. Rosenkrantz, R. Stearns and P. Lewis II, "An analysis of several heuristics for the traveling salesman problem," vol. 6, pp. 563-581, 1977.
11. F. Glover and A. P. Punnen, "The Travelling Salesman Problem: New Solvable Cases and Linkages with the Development of Approximation Algorithms," vol. 48, no. 5, 1997.
12. E. H. Moore, "On Certain Crinkly Curves," Trans. Amer. Math Soc., vol. 1, pp. 72-90, 1900.
13. D. E. Rosenkrantz, R. E. Stearns and P. M. Lewis II, "An analysis of several heuristics for the traveling salesman problem," SIAM J. Comput, p. 563–581, 1977.
14. D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study," in Local Search in Combinatorial Opti- mization, Chichester, UK, John Wiley & Sons, 1997.
15. N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Graduate School of Industrial Administration, CMU, 1976.
16. N. Christofides, Graph theory - An Algorithmic Approach, New York: Academic Press, 1974.
17. F. Bock, "An algorithm for solving "traveling-salesman" and related network optimization problems," in Unpublished manuscript a-ssociated with talk presented at the 14th ORSA National Meeting, 1958.
18. G. Croes, "A method for solving travelling salesman problems," Operation Resources, vol. 6, pp. 791-812, 1958.
19. E. Aarts and J. K. Lenstra, Local Search in Combinatorial Optimization, Princeton, New Jersey: Princeton University Press, 2003.
20. G. Gutin and A. Punnen, The Traveling Salesman Problem and Its Variations, vol. 12, Kluwer, Dordrecht: Springer US, 2002.
21. M. M. Flood, "The traveling-salesman problem," Operation research, vol. 4, pp. 61-75, 1956.
22. S. Lin and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling-Salesman Problem," vol. 21, no. 2, p. 498–516, 1973.
23. K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," EJOR, vol. 12, pp. 106-130, 2000.
24. P. J. Laarhoven and E. H. Aarts, Simulated Annealing: Theory and Applications, Heielberg, Germany: Springer, 1987.
25. B. Gorkemli and D. Karaboga, "Quick Combinatorial Artificial Bee Colony -qCABC- Optimization Algorithm for TSP," vol. 1, no. 5, 2013.

26. M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approachto the traveling salesman problem," vol. 1, no. 1, 1997.
27. P. Hsiao, "An effective PSO-based algorithm for the traveling-salesman problem," 2013.
28. X. S. Yang and S. Deb, "Cuckoo search: recent advances and applications," vol. 24, no. 1, 2014.
29. W. J. Cook, Combinatorial optimization, New York: Wiley, 1998.
30. M. Hahsler and K. Hornik, "TSP – Infrastructure for the Traveling Salesperson Problem," vol. 23, no. 2, 2007.
31. N. Christofides, Graph theory - An Algorithmic Approach, New York: Academic Press, 1974.
32. K. Buchin, "Space-Filling Curves," in Organizing Points Sets: Space-Filling Curves, Delaunay Tessellations of Random Point Sets, and Flow Complexes, Berlin, Free University of Berlin, 2007, pp. 5-30.
33. J. J. Bartholdi, L. K. Platzman, R. L. Collins and W. H. Warden, "A Minimal Technology Routing System for Meals on Wheels," vol. 13, no. 3, pp. 1-8, 1983.