Pareto-optimal Algorithms for Metric TSP: Experimental Research

Ekaterina N. Beresneva (Chirkova) and Prof. Sergey M. Avdoshin

***Abstract*—The Travelling Salesman Problem (TSP) is a fundamental task in combinatorial optimization. A special case of the TSP is Metric TSP, where the triangle inequality holds. Solutions of the TSP are generally used for costs minimization, such as finding the best tour for round-the-world trip or construction of very large-scale integration schemes. Since the TSP is NP-hard, heuristic algorithms providing near optimal solutions will be considered. The objective of this article is to find a group of Pareto optimal heuristic algorithms for Metric TSP under criteria of run time efficiency and qualitative performance as a part of the experimental study. Classification of algorithms for Metric TSP is presented. Feasible heuristic algorithms and their prior estimates are described. The data structure and the details of the research methodology are provided. Finally, results and prospective research are discussed.**

Keywords—travelling salesman problem, resource-efficient algorithm, heuristic algorithm, posterior estimate, computational experiment.

I. 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 heuristic algorithms for Metric TSP by criteria of run time and qualitative performance as part of the experimental investigation.

Manuscript received April 18, 2017.

Ekaterina N. Beresneva is with the Department of Software Engineering, Faculty of Computer Science, National Research University Higher School of Economics, Moscow, Russia (e-mail: enchirkova@edu.hse.ru).

Prof. Sergey M. Avdoshin is with the Department of Software Engineering, Faculty of Computer Science, National Research University Higher School of Economics, Moscow, Russia (e-mail: savdoshin@hse.ru).

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 the mathematical formulation of Metric TSP, the specification of metric types and, at last, definition of Pareto optimization. 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.

II. Theoretical Basis

A. Problem Formulation

In this paper, mathematical formulation of Metric TSP is adopted as follows.

Given a complete weighted undirected graph $G = (V, V^2)$ which contains $N = |V|$ vertices. Graph vertices are indexed as $index = V \to I, I = \{1, 2, \dots, N\}$, and $(\forall v_i \in V)(\forall v_j \in V)$ it is true that if $v_i \neq v_j$ then $i \neq j$, where $i = index(v_i)$. The distance between two vertices v_i and v_j is calculated by distance function $d(v_i, v_j)$. Here a real-valued function $d(\cdot,\cdot)$ on $V \times V$ satisfies [3]:

1. $d(v_i, v_j) \geq 0$ (non-negativity axiom)
2. $d(v_i, v_j) = 0$ if and only if $v_i = v_j$ (identity axiom)
3. $d(v_i, v_j) = d(v_j, v_i)$ (symmetry axiom)
4. $d(v_i, v_k) \leq d(v_i, v_j) + d(v_j, v_k)$ (triangle inequality axiom)

Let S be a set of all Hamiltonian cycles of G. It is defined as $S = \{p: V \to V | (p(1) = 1 \& (\forall i \in V)(\forall j \in V)$ $(p(i) = p(j) => i = j\}$. An example of a Hamiltonian circuit $s \in S$ is $(p_1, p_2, \dots, p_N)$, where p_i is used as abbreviated notation of $p(i)$.

Weight of a Hamiltonian cycle $s \in S$ can be found according to the formula (1):

$$f(s) = d(p_1, p_N) + \sum_{i=1}^{N-1} d(p_i, p_{i+1}) \quad (1)$$

The set of vertices V is determined in Euclidean space R^n by their coordinates. Under these circumstances, the distance between two vertices $v = (x_1, x_2, \dots, x_n)$ and $w = (x_1, x_2, \dots, x_n)$ is defined as (2):

$$d_0(v, w) = \sqrt{\sum_{i=1}^{n} \left(x_i(v) - x_i(w)\right)^2} \quad (4)$$

The formulation for the metric travelling salesman problem is to find such s_0 that $f(s_0) = \min_{s \in S} f(s)$ for the given metric d_0.

B. Resourse-efficient parameters

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

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

Qualitative performance can be calculated using (7):

$$f_\varepsilon(m, N) = \frac{f(s) - f(s_0)}{f(s_0)} * 100\%, \quad (7)$$

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 library TSPLIB as the lengths of the best reported solutions for each of the instances [4].

C. Pareto-optimality

Pareto optimization [5] or multi-objective optimization problem for the TSP can be formulated as (8):

$$\min_{m \in M} \{f_\varepsilon(m), f_t(m)\} \quad (8)$$

Pareto optimal solutions are solutions that cannot be improved in any of the objectives without degrading, at least, one of the other objectives. In mathematical terms, a feasible solution $m^1 \in M$ is said to (Pareto) dominate another solution $m^2 \in M$, if

1. $f_i(m^1) \leq f_i(m^2)$, for all indices $i \in \{\varepsilon, t\}$.
2. $f_j(m^1) < f_j(m^2)$, for at least one index $j \in \{\varepsilon, t\}$.

III. 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 [6].

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 [7]:

- 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 [8] and Savings [9] [10].
- 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 [11] [12] and space-filling curves group [13].
- Subtour insertion methods:
  An initial subtour is selected, for example one edge is selected. In each iteration, a node is selected and inserted into the subtour according to some criterion. To insert a node an edge in the subtour is replaced by two edges not in the subtour. The insertion heuristics are described by 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]. Currently, the most simple tour-improving heuristic used in practice is 2-Opt heuristic [20]. It was introduced and described by Flood [21], Croes [18] and Bock [17]. The later algorithm of Lin and Kernighan (LKH) [22] appeared on the basis of k-Opt tour-finding approach. Also this group consists of simulated annealing [23], evolutionary [24] and swarm intelligence methods [25] [26].

IV. 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 of its high computational complexity – $O(N^3 \log N)$ – in comparison with Greedy one – $O(N^2 \log N)$. In addition, 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 two types of tour improving algorithms – the most simple one (2-Opt) and the most perspective one (LKH).

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 [21].

3) Greedy (GRD).

The Greedy heuristic constructs a path by adding the shortest edge to the tour until a cycle with A edges, $A < N$, is created, or the degree of any vertex exceeds two [27].

4) Nearest Addition (NA), Simple Cheapest Insertion (SCI), Simple Nearest Segment Insertion (SNSI).

The fundamental idea of NA, SCI and SNSI is to start with an initial subtour made of the shortest edge and to add repeatedly other vertices using various rules. Depending on the algorithm the vertex not yet in the cycle should be inserted so that:

a) In NA it is the closest to any node in the tour;
b) In SCI its addition to the tour gives a minor increment of its length;
c) In SNSI 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 [19].

5) Nearest Insertion (NI), Cheapest Insertion (CI), Nearest Segment Insertion (NSI).

Algorithms NI, CI and NSI are variations of NA, SCI and SNSI respectively. The feature of modified methods is additional computation that selects the best place for each inserting node.

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 [16].

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 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. Figure 2 shows the order of the cells after one, two and three subdivision steps respectively [28] .

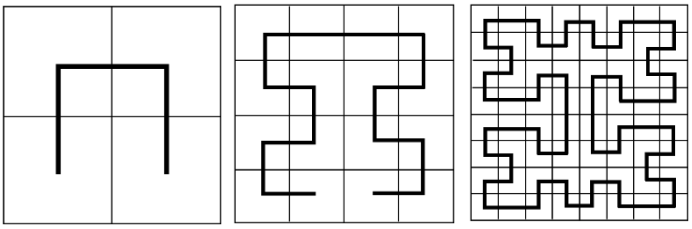

Fig. 1. The order for the Moore curve after 1, 2 and 3 subdivision steps.

10) 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 $d(a,b) + d(c,d) > d(a,c) + d(b,d)$ (Fig. 1) [19].

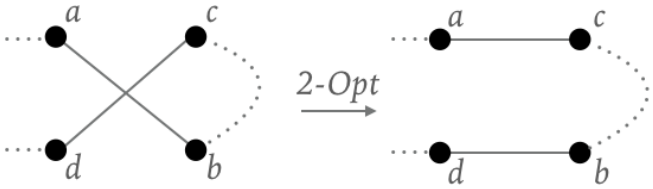

Fig. 2. 2-Opt modification.

11) 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 [22].

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

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 [30], 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 I. UPPER-BOUND ESTIMATES OF ALGORITHMS

#	Algorithm	Upper-bound estimate
1	NN DENN	$0.5\lceil \log_2 N + 1 \rceil$
2	GRD	$0.5\lceil \log_2 N + 1 \rceil$
3	NA, SCI, SNSI NI, CI, NSI	$2 - \frac{2}{N}$
4	DMST, DMST-M	$2 - \frac{2}{N}$
5	CHR	$\frac{3}{2}$
6	2-Opt	≈ 2
7	LKH	≈ 2
8	MC	$\log N$

TABLE II. RUNNING TIME OF ALGORITHMS

#	Algorithm	Running time
1	NN DENN	$O(N^2)$
2	GRD	$O(N^2 \log N)$
3	NA, SCI, SNSI NI, CI, NSI	$O(N^2)$
4	DMST, DMST-M	$O(N^2)$
5	CHR	$O(N^3)$
6	2-Opt	$O(N^2)$
7	LKH	$O(N^{2,2})$
8	MC	$O(N \log N)$

V. WAVE FUNCTION AS THE DATA STRUCTURE IN USE

The key aspect of realization of the algorithms is data structure they make use of. In the study the Hamiltonian circuits are presented as *wave functions*, which are fixed-length one-dimensional arrays [31]. Their sizes are equal to N.

The wave function data structure decreases the computational time of the algorithms in comparison with standard C++ template *std::vector<T>* (Fig. 3).

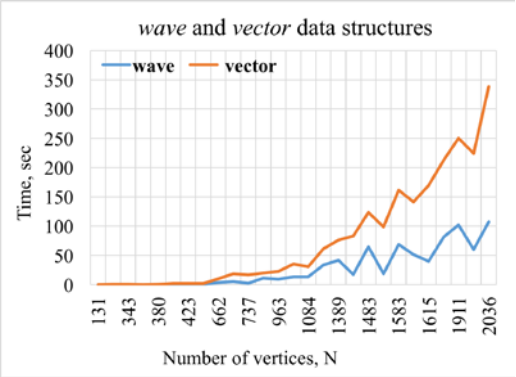

Fig. 3. The time-efficiency comparison of *wave* and *vector* data structures.

VI. 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++. VLSI data sets from an open library TSPLIB [2] are selected as input data for algorithms. There are 102 instances in the VLSI collection that range in size from 131 vertices up to 744,710 vertices. There is one data set for each number of vertices. 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:

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

The computational experiment corresponds to the following scenario:

```
for algorithm m in range [1…9, 11] (see IV)
   for data set N in range [1…102]
      for i in range [1…11]
         f_εi(m,N), f_ti(m,N) are calculated
            if i > 1 then
               f_εmin(m,N) is memorized
               f_tsum(m,N) is calculated
      f_tavg(m,N) = f_tsum(m,N) / 10
      // 2-Opt stage
      if m != 11 (m is not LKH)
         f_ε(m + 2–Opt,N), f_t(m + 2–Opt,N) are
         calculated
   E(f_εmin(m,N)), σ(f_εmin(m,N)) for all N - ?
   max(f_εmin(m,N)), min(f_εmin(m,N)) for all N - ?
```

Metrics used in scenario have following meanings:

- $f_{\varepsilon_{min}}(m,N)$ – best qualitative performance of m,
- $f_{t_{sum}}(m,N)$ – accumulative running time of m (sec),
- $f_{t_{avg}}(m,N)$ – average running time of m (sec),
- $f_{\varepsilon}(m + 2\text{–}Opt, N)$ – qualitative performance of $m + 2\text{–}Opt$,
- $f_t(m + 2\text{–}Opt, N)$ – running time of $m + 2\text{–}Opt$,

- $E\left(f_{\varepsilon_{min}}(m,N)\right)$ – expected value of qualitative performance of m for all N,
- $\sigma\left(f_{\varepsilon_{min}}(m,N)\right)$ – standard deviation of qualitative performance of m for all N,
- $max\left(f_{\varepsilon_{min}}(m,N)\right), min\left(f_{\varepsilon_{min}}(m,N)\right)$ – maximum and minimum values of qualitative performance of m for all N.

Qualitative performance metrics are represented in Table 3. Table color scheme varies from green (the best result in a column) to red (the worst value in a column).

The line graphs that illustrate $f_\varepsilon(m,N)$ and $f_t(m,N)$ of the algorithms are provided (see Fig. 4, Fig. 5). There are no MC algorithms as their key figures are outstanding (Table 3).

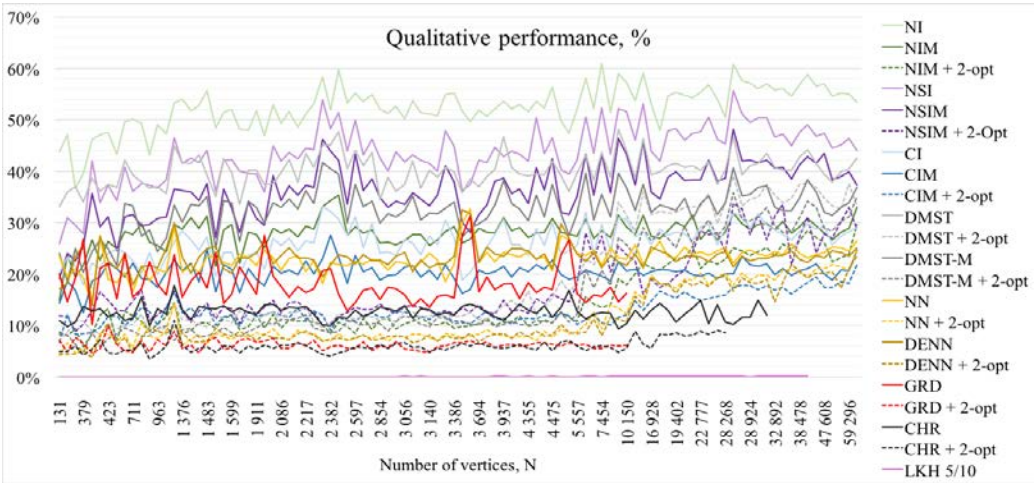

Fig. 4. Qualitative performance of the algorithms.

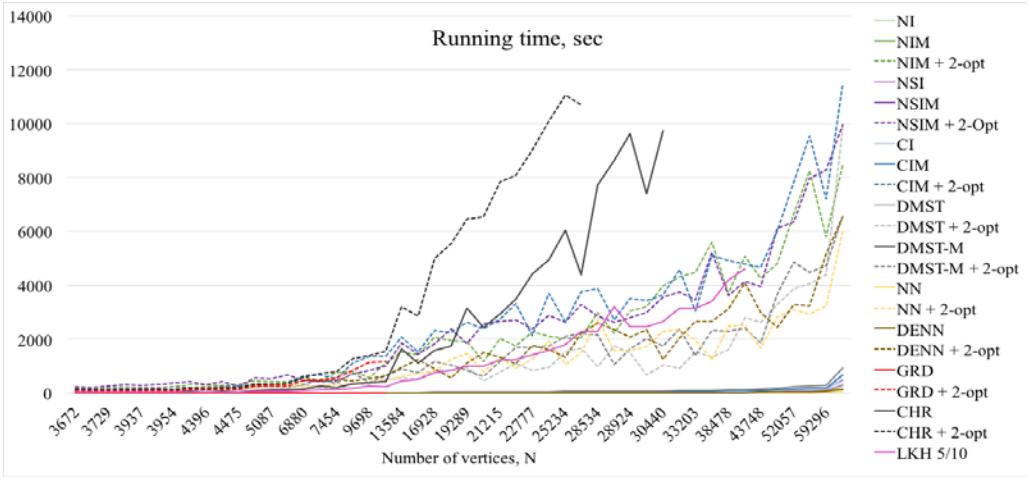

Fig. 5. Running time of the algorithms.

TABLE III. QUALITATIVE PERFORMANCE METRICS OF THE ALGORITHMS

Algorithm	E(ε)	σ(ε)	max ε	min ε
LKH 5/10	0,07%	0,05%	0,23%	0,00%
CHR + 2-Opt	5,77%	0,68%	11,02%	3,47%
GRD + 2-opt	6,22%	0,70%	9,89%	4,69%
DENN + 2-opt	10,95%	4,42%	23,51%	3,82%
NN + 2-opt	11,44%	1,87%	24,77%	4,26%
CHR	12,61%	1,08%	17,79%	9,31%
CIM + 2-opt	13,05%	2,29%	21,86%	6,74%
NIM + 2-opt	14,60%	5,53%	29,66%	5,86%
DMST-M + 2-opt	16,08%	8,39%	40,61%	4,80%
NSIM + 2-opt	17,63%	5,82%	33,65%	8,92%
GRD	18,12%	2,91%	31,34%	10,30%
DMST + 2-opt	19,08%	9,54%	39,12%	6,91%
CIM	20,31%	1,44%	27,54%	12,46%
DENN	23,28%	1,62%	32,53%	13,88%
NN	23,94%	1,64%	30,97%	12,94%
CI	26,25%	2,70%	33,05%	17,94%
NIM	27,98%	1,89%	35,29%	14,89%
DMST-M	32,41%	3,15%	41,68%	18,55%
MC + 2-opt	32,41%	22,54%	177,83%	6,21%
NSIM	36,23%	4,23%	48,17%	19,15%
DMST	40,09%	2,34%	48,88%	33,16%
NSI	43,55%	5,64%	55,61%	25,89%
NI	52,46%	3,19%	60,94%	36,77%
MC	63,93%	25,58%	242,41%	33,07%

VII. Pareto-optimal algorithms

We decided to select six different data sets with N = {1083, 5087, 10150, 30440, 52057, 104815} to plot charts that illustrate Pareto-optimal algorithms.

The charts are shown below (see Fig. 6, Fig. 8, Fig. 10, Fig. 12, Fig. 14, Fig. 16). 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.

There are five charts (see Fig. 7, Fig. 9, Fig. 11, Fig. 13, Fig. 15), except one with $N = 104814$, 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.

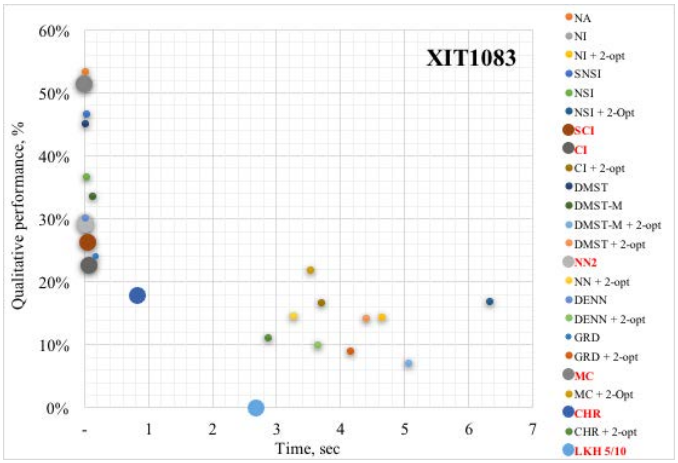

Fig. 6. Pareto-optimal algorithms for XIT1083.tsp, N = 1083.

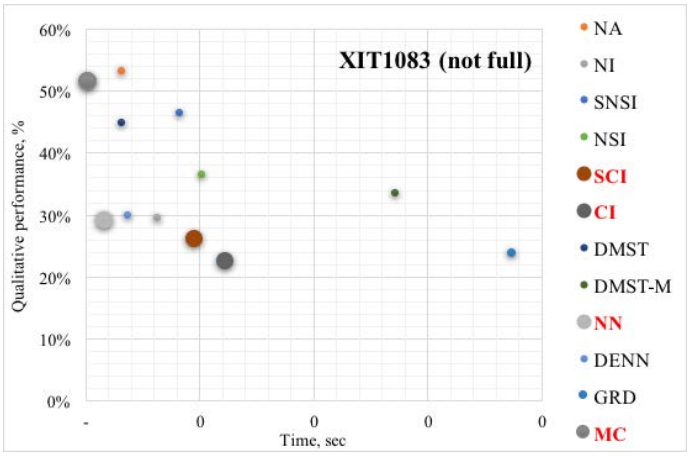

Fig. 7. Pareto-optimal algorithms, for XIT1083.tsp, N = 1083 (scale-up).

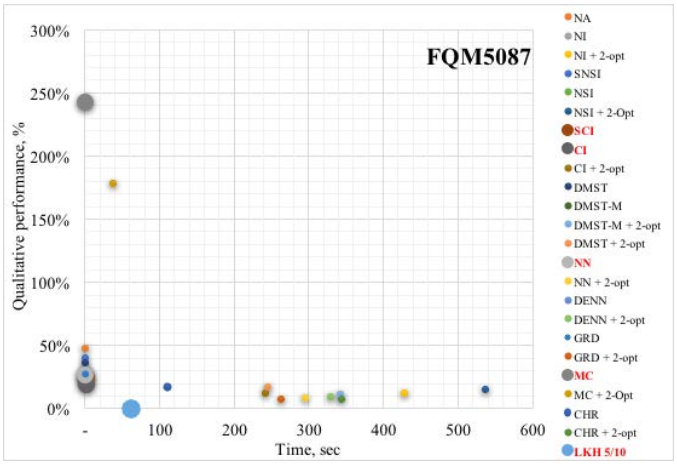

Fig. 8. Pareto-optimal algorithms for FQM5087.tsp, N = 5087.

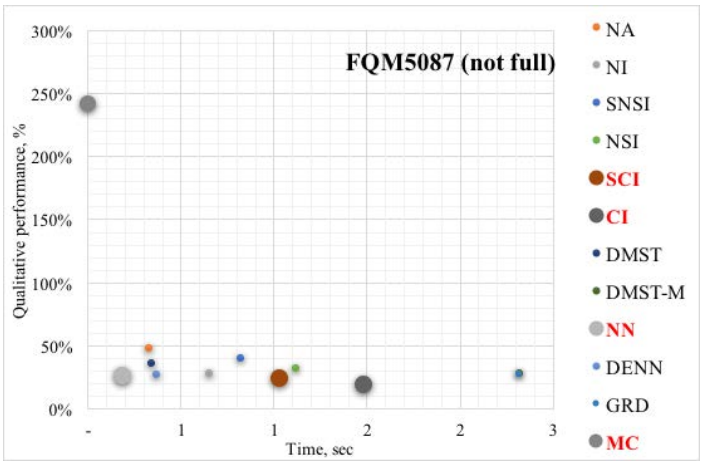

Fig. 9. Pareto-optimal algorithms for FQM5087.tsp, N = 5087 (scale-up).

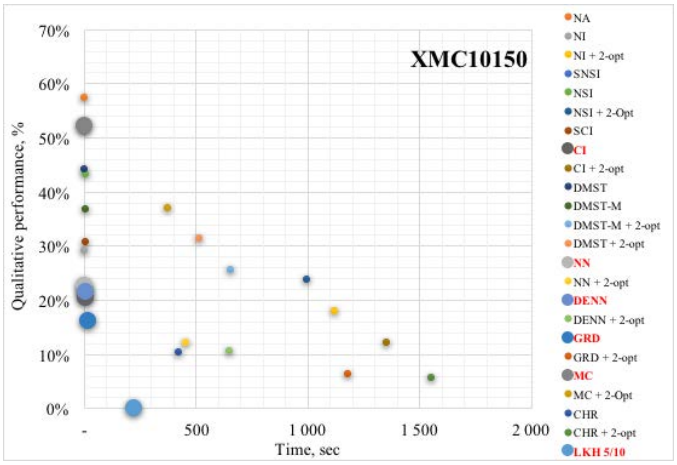

Fig. 10. Pareto-optimal algorithms for XMC10150.tsp, N = 10150.

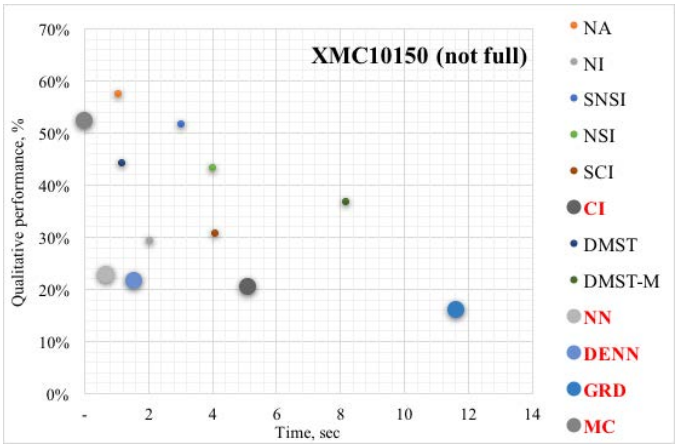

Fig. 11. Pareto-optimal algorithms for XMC10150.tsp, N = 10150 (scale-up).

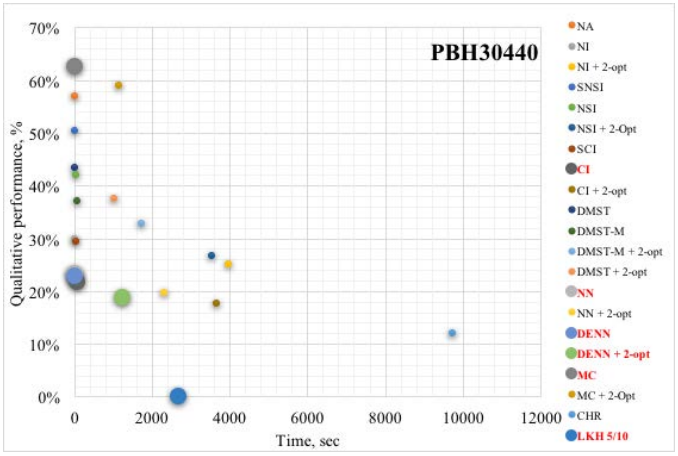

Fig. 12. Pareto-optimal algorithms for PBH30440.tsp, N = 30440.

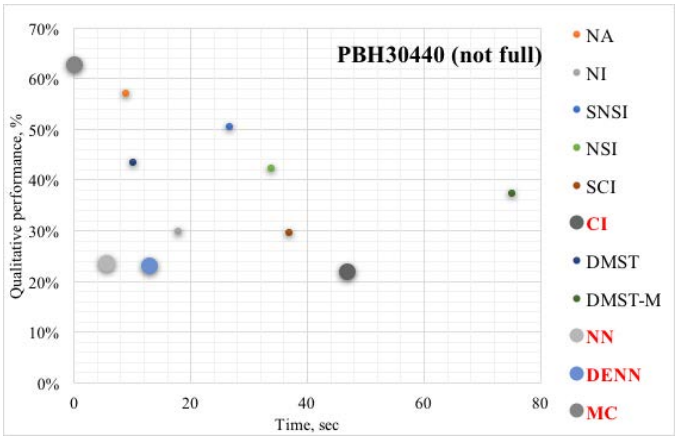

Fig. 13. Pareto-optimal algorithms for PBH30440.tsp, N = 30440 (scale-up).

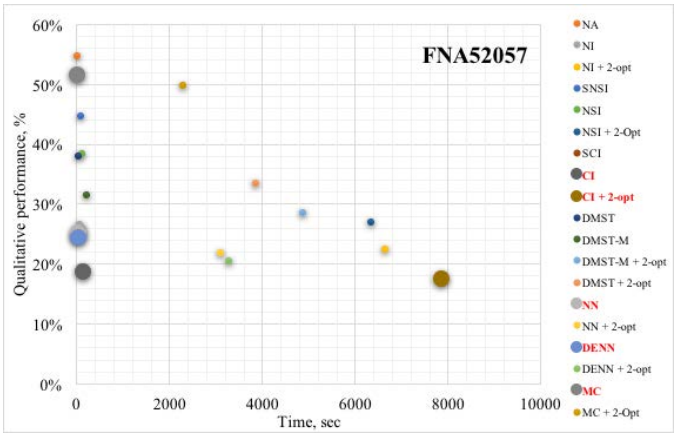

Fig. 14. Pareto-optimal algorithms for FNA52057.tsp, N = 52057.

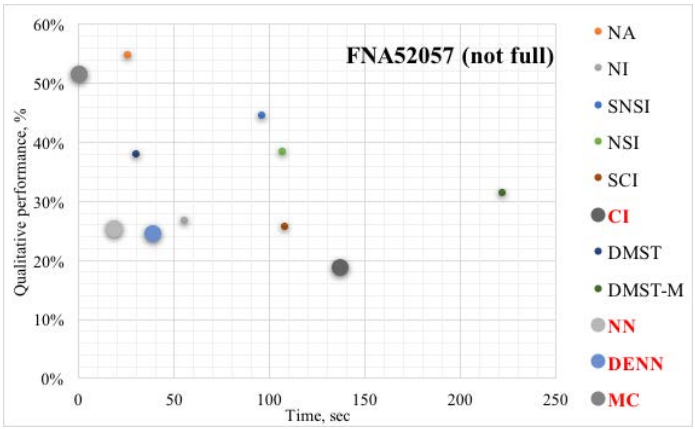

Fig. 15. Pareto-optimal algorithms for FNA52057.tsp, N = 52057 (scale-up).

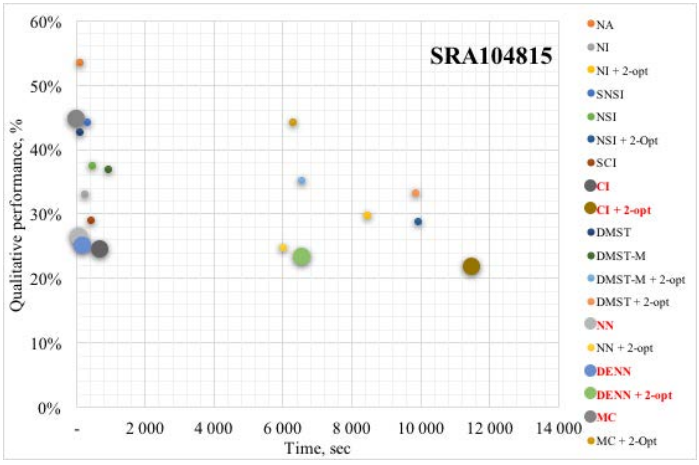

Fig. 16. Pareto-optimal algorithms for SRA104815.tsp, N = 104815.

Full results are presented in Table 4. Dash sign '-' is used because some methods have not tested on large N yet. So we cannot claim whether these algorithms are Pareto-optimal or not.

TABLE IV. PARETO-OPTIMAL ALGORITHMS FOR 6 GROUPS OF N

XIT1083	FQM5087	XMC10150	PBH30440	FNA52057	SRA104815
SCI	SCI				
CI	CI	CI	CI	CI	CI
				CI+2-Opt	CI+2-Opt
NN	NN	NN	NN	NN	NN
		DENN	DENN	DENN	DENN
			DENN+ 2-Opt		DENN+ 2-Opt
		GRD			
MC	MC	MC	MC	MC	MC
CHR				-	-
LKH	LKH	LKH	LKH	-	-

The rough estimations can be suggested on the basis of VLSI data sets **only**:

1. SCI, and CHR methods are efficient for small $N \lesssim 5000$.
2. GRD is one of the bests for $N \approx 10000$.
3. DENN is high-performance for $N \gtrsim 10000$.
4. CI + 2-Opt combination gives good results for large $N \gtrsim 50000$.
5. DENN + 2-Opt is efficient on $N \approx 30000$ and $N \approx 100000$.
6. There are three heuristics included in each Pareto-optimal group for different N. They are CI, NN and MC.

Despite the lack of information on LKH algorithm for N = 52057 and N = 104814, this method continues to be the most perspective heuristic. As it can be seen from Table 3, expected value of qualitative performance of LKH is 0.07%. From there, we decided to add LKH to the generalized group of Pareto-optimal algorithms.

Overall, the generalized group of Pareto-optimal algorithms consists of **CI**, **NN**, **MC** and **LKH** heuristics.

VIII. 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 consists of CI, NN, MC 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] M. Reed and B. Simon, Methods of Modern Mathematical Physics, London: Academic Press, 1972.

[4] 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].

[5] Wikipedia, "Multi-objective optimization," [Online]. Available: https://en.wikipedia.org/wiki/Multi-objective_optimization. [Accessed 03 02 2017].

[6] University of Waterloo, "Status of VLSI Data Sets," [Online]. Available: http://www.math.uwaterloo.ca/tsp/vlsi/summary.html. [Accessed 08 02 2017].

[7] B. Hock, "An examination of heuristic algorithms for the Travelling Salesman problem," University of Cape Town, Cape Town, 1988.

[8] M. Fisher, G. Nemhauser and L. Wolsey, An analysis of approximations for maximizing submodular set functions, Springer, 1978.

[9] 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.

[10] B. L. Golden, T. Magnanti and H. Nguyen, "Implementing vehicle routing algorithms," *Networks*, vol. 7, no. 2, pp. 113-148, 1977.

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

[12] 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.

[13] E. H. Moore, "On Certain Crinkly Curves," *Trans. Amer. Math Soc.*, vol. 1, pp. 72-90, 1900.

[14] D. Johnson and L. McGeoch, "The Traveling Salesman Problem: A Case Study," in *Local Search in Combinatorial Optimization*, Chichester, 1997, pp. 215-310.

[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] K. Helsgaun, "An effective implementation of the Lin–Kernighan

traveling salesman heuristic," *EJOR*, vol. 12, pp. 106-130, 2000.

[23] P. J. Laarhoven and E. H. Aarts, imulated Annealing: Theory and Applications, Heidelberg, Germany: Springer, 1987.

[24] 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.

[25] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approachto the traveling salesman problem," vol. 1, no. 1, 1997.

[26] B. Gorkemli and D. Karaboga, "Quick Combinatorial Artificial Bee Colony -qCABC- Optimization Algorithm for TSP," vol. 1, no. 5, 2013.

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

[28] 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.

[29] K. Helsgaun, "LKH," Keld Helsgaun, [Online]. Available: http://www.akira.ruc.dk/~keld/research/LKH/. [Accessed 24 02 2017].

[30] 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.

[31] S. M. Avdoshin and V. V. Belov, "Obobschennyi metod "volny" dlya resheniya ekstremal'nykh zadach na grafah," *ZHVM and MF*, vol. 19, no. 3, pp. 739-755, 1979.

[32] C. Nilsson, "Heuristics for the traveling salesman problem," Linkoping University, Linkoping, Sweden, 2013.

[33] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671-680, 13 05 1983.

[34] P. van Laarhoven and E. Aarts, Simulated Annealing: Theory and Applications, Heidelberg: Springer, 1987.

[35] B. Gorkemeli and D. Karaboga, "Quick Combinatorial Artificial Bee Colony -qCABC- Optimization Algorithm for TSP," in *2nd International Symposium on Computing in Informatics and Mathematics*, 2013.

[36] M. Dorigo, Ant colony optimization, Cambridge: MIT Press, 2004.

[37] T. Lust and J. Teghem, "The Multiobjective Traveling Salesman Problem: A Survey and a New Approach," *Studies in Computational Intelligence* , vol. 272, pp. 119-141, 2010.