http://www.moderntechno.de/index.php/meit/article/view/meit26-01-063
DOI: 10.30890/2567-5273.2023-26-01-063

UDC 519.16

SOFTWARE ARCHITECTURE FOR SOLVING LARGE-SCALE TRAVELING SALESMAN PROBLEMS

Bazylevych R. P.
d.t.s., prof.
ORCID: 0000-0002-7949-1353
Lviv Polytechnic National University, St. Bandery str. 12, Lviv, 79013
Kutelmakh R.K.
c.t.s..
ORCID: 0009-0008-6499-1161
Lviv Polytechnic National University, St. Bandery str. 12, Lviv, 79013
Bhanu Prasad*
Professor
Department of Computer and Information Sciences, Florida A&M University, Tallahassee, Florida 32307, USA.
**Corresponding author*

Abstract. *The architecture of a cross-platform software for solving large-scale Traveling Salesman Problem (TSP) is presented. According to this architecture, the software consists of a central control module and additional modules - input data, input data processing, input/output data visualizers, and algorithm library. The developed software can be used to solve large-scale TSP (more than 100,000 points) while ensuring that the length of the solution is within 0.5% above the optimum and the solution is generated in an acceptable amount of time. The algorithms integrated into the software have computational complexity close to linear-logarithmic, which makes it possible to reduce the runtime of the solutions.*

Key words: *Traveling salesman problem, decomposition, large-scale, software, architecture, combinatorial optimization, algorithm*

Introduction

Traveling Salesman Problem (TSP) [1, 2] has a wide range of applications in areas such as transportation systems, automated design, integrated circuits testing and manufacturing, printed circuit boards, laser cutting of plastics and metals, protein structure study, continuous line drawing technology, embroidery, welding, X-ray crystallography, etc. An important feature of these TSPs is their large-scale, which, for many of them, is more than several million points. TSP refers to NP-hard problems because TSP has factorial computational complexity. This makes it difficult to obtain exact solutions in a reasonable amount of time for large-scale TSPs.

Existing software for solving TSP [3, 4] is based on algorithms that have a quadratic and higher computational complexity and cannot be used for large-scale problems. Helsgaun developed the most effective heuristic algorithm, LKH, for solving the TSP [5 - 7]. As the experiments show, LKH provides a solution which is within 0.1-0.2% length above the optimal one and the solution is generated in an acceptable amount of time. The computational complexity of the LKH algorithm is O($N^{2.2}$), where N is the total number of points in the TSP.

Several decomposition algorithms for solving large-scale TSPs were developed: a macro modeling method for clustered TSP [8], a partial solution expanding method for arbitrary distribution of points [9], initial solution optimization methods –

sequential optimization [10], geometric areas optimization, and triangulation-based optimization [11]. The developed methods can be used for large-scale TSPs, as proved by the experiments [12]. These developed algorithms have a computational complexity close to linear-logarithmic. According to these developed methods, the process of solving a TSP consists of the following stages:

- forming a set of clusters or subsets of points, depending on the distribution of the points;
- finding an initial solution;
- optimization of the initial solution.

It is appropriate to develop an applied software system for solving large-scale TSP based on these methods. In order to have efficient calculations, special data structures and algorithms have been developed in this research.

Problem formulation

TSP is formulated as follows: given a set of points P, described by the coordinates:

$$P = \{p_1, p_2, \ldots, p_N\}, \text{ where } p_i = (x_i, y_i) \text{ for } i \in \{1, 2, \ldots, N\};$$

and the metric dist: $P \times P \to R$ on set P:

$$dist_E\,(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad \text{for } i, j \in \{1, 2, \ldots, N\} \text{ (Euclidean metric)} \qquad (1)$$

$$dist_O\,(p_i, p_j) = |x_i - x_j| + |y_i - y_j| \quad \text{for } i, j \in \{1, 2, \ldots, N\} \text{ (orthogonal metric)} \qquad (2)$$

the problem is to minimize the length of the closed route M, which visits all the points of P:

$$\text{len } M \to \text{minimum}, \qquad (3)$$

where $M = <m_1, m_2, \ldots, m_N>$, where $m_i \in P$ and $|M| = N$; and

$$len\ M = \ \text{dist}(m_i, m_{i+1}) + \text{dist}(m_N, m_1) \text{ is the route length function.} \qquad (4)$$

The problem is considered symmetric TSP if $dist(p_i, p_j) = \text{dist}(p_j, p_i)$, otherwise it is asymmetric TSP. Under the condition of returning to the initial point, the problem is called closed TSP, otherwise it is unclosed TSP.

Related works

Experimental investigation of the efficiency of the existing TSP solving methods was carried out. The "Concorde" [3] software and LKH [4] heuristic were chosen for the experiments. According to the experimental results, the exact method implemented in “Concorde” software provides a relatively quick optimal solution for the TSPs with no more than 1,000 points. As the experiments show, LKH heuristic does not guarantee optimality (but the solution quality is near-optimal). LKH provided optimal solutions for all TSPs except for those with 1,000 points. For a TSP with 1,000 points, the solution quality of the LKH method is 0.043% above the optimal, which is near-optimal. Fig. 1 illustrates: (a) the characteristics of some existing heuristic methods for solving the TSP, (b) their deviations from the optimal solution, and (c) their computational complexity.

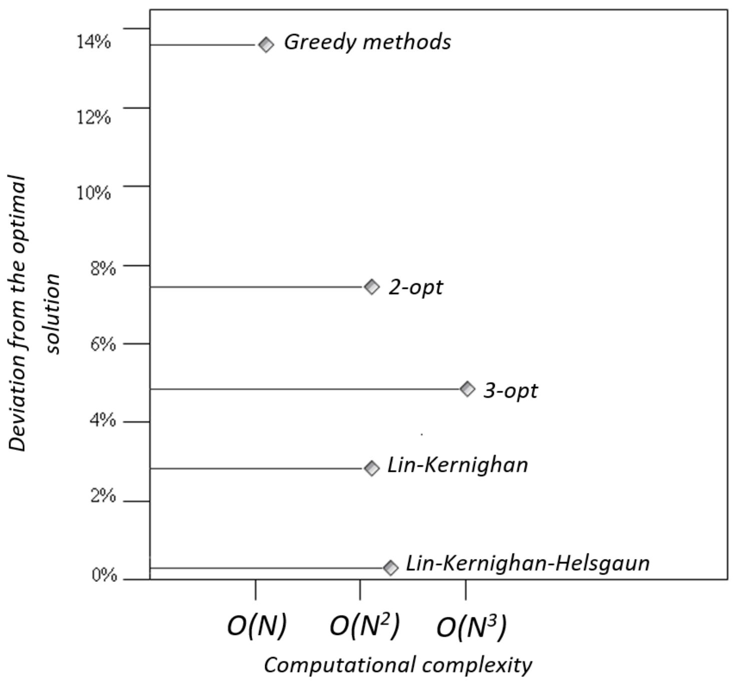

Figure 1 – Runtime characteristics of some heuristic methods for solving TSP

Proposed software architecture for solving large-scale TSP

The software architecture of the system for optimizing large-scale TSPs is presented. The system is named "VRP Modeler" and consists of a central control module and additional modules. The control module is responsible for basic functions for manipulating abstract documents, method libraries, visualization modules, and input data generators. Additional modules contain methods, visualizers, and data handlers.

The main components of the software are:

- central control module;
- modules for visual viewing input data, output data, and intermediate results;
- input data generators module;
- input data processing module;
- method libraries for solving the problem.

Fig. 2 shows the structure of VRP Modeler and the links between its components. The Input Data module accepts the input data. The input can be either in TSPLIB format (TSPLIB is the library that contains known TSP instances with points ranging from 14 to 85,900) or generated automatically by an algorithm according to given settings such as the number of points, clustered or uniform distribution of points, etc. This input is supplied to the Input Data Processing module where the input is converted to an internal format. E.g., in the internal format, the data (i.e., set of points) can be presented as a one-dimensional array of elements, and the route can be presented as a two-level bidirectional linked list.

The input in the internal format is supplied to the Control module that works as a hub between various other modules of the system. Further, the data is submitted to the Input Data Visualization module where it is displayed in the application window. The Input Data Visualization module is a set of C++ classes for configuring and displaying the data of the given TSP. E.g., each point can be visualized as a circle, rectangle, rhombus, etc., and a scale for increasing or decreasing the accuracy of calculations, rotating the problem area, and mirroring the area.

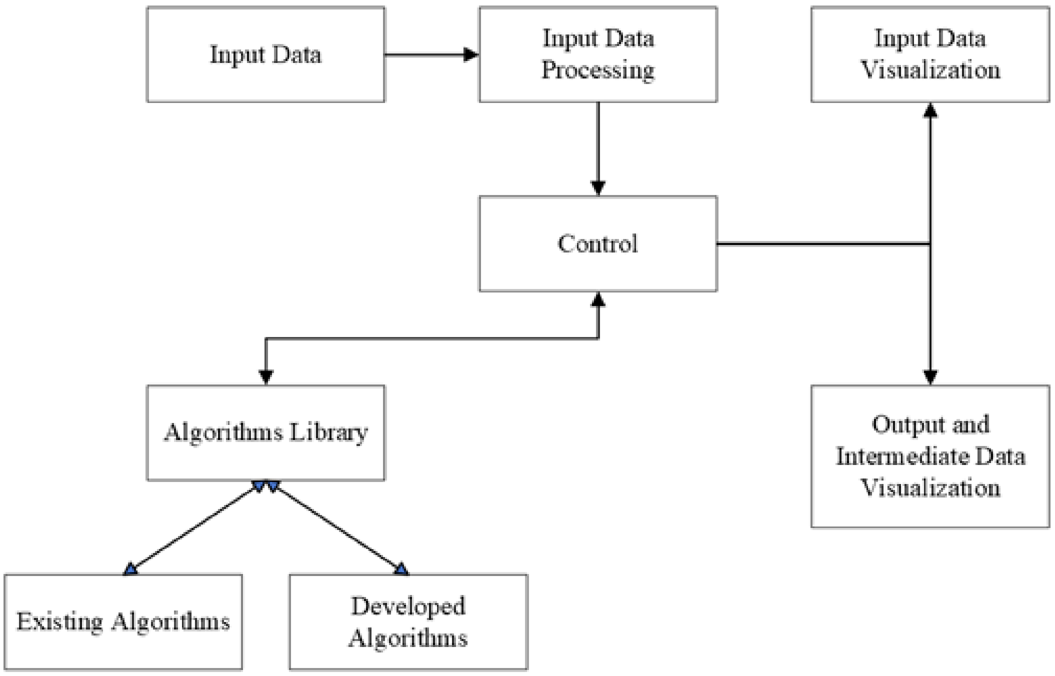

Figure 2 – Schematic diagram of VRP Modeler

To run a specific algorithm for solving the TSP, the Algorithms Library module is used. This library consists of some existing and developed algorithms. Each algorithm has its own settings like the initial route, basic algorithm, number of iterations, scanning area size, overlapping area size, etc. After the execution of an algorithm, the resultant data is returned to the Control module. The system is developed in such a way that new algorithms can be easily added to the system (i.e., add to either the Existing Algorithms module or the Developed Algorithms module).

After the execution of an algorithm, the resultant data is transferred to the Output and Intermediate Data Visualization module, via the Control module. The Output and Intermediate Data Visualization module is an extension of the Input Data Visualization module and uses specific parameters according to the problem-solving algorithm. E.g., for a method based on macromodeling, it "knows" and draws elements such as clusters of points, macroroute, elementary optimization area, etc. Fig. 3 shows further details about the Algorithms Library module of Fig. 2.

Data structures to represent a route is important, especially for the program implementation of large-scale TSPs. Most often, the use of certain data structures affects the execution time. The use of *k*-opt leads to the reversal (i.e., reverse passage of points) in a particular section of the route. The time required for this operation is within the range from $O(\log_2(N))$ to $O(N)$. The reverse operation will take an $O(N)$ time using regular array, which is unacceptable for large-scale TSPs. It is necessary to use another data structure such as two-level tree or two-level double-linked list. In this case, the reverse operation will take $O(\sqrt{N})$ time. This is the best data structure for TSPs with 1,000 to 1,000,000 points. The splay-tree data structure is best for large-scale TSPs; in the worst case, it will take $O(\log_2(N))$ reverse time (but its implementation is more complicated). The best choice is to use arrays for small TSPs, two-level double-linked lists for TSPs with 1,000 to 1,000,000 points, and splay-trees for very large-scale TSPs (i.e., beyond 1,000,000 points).

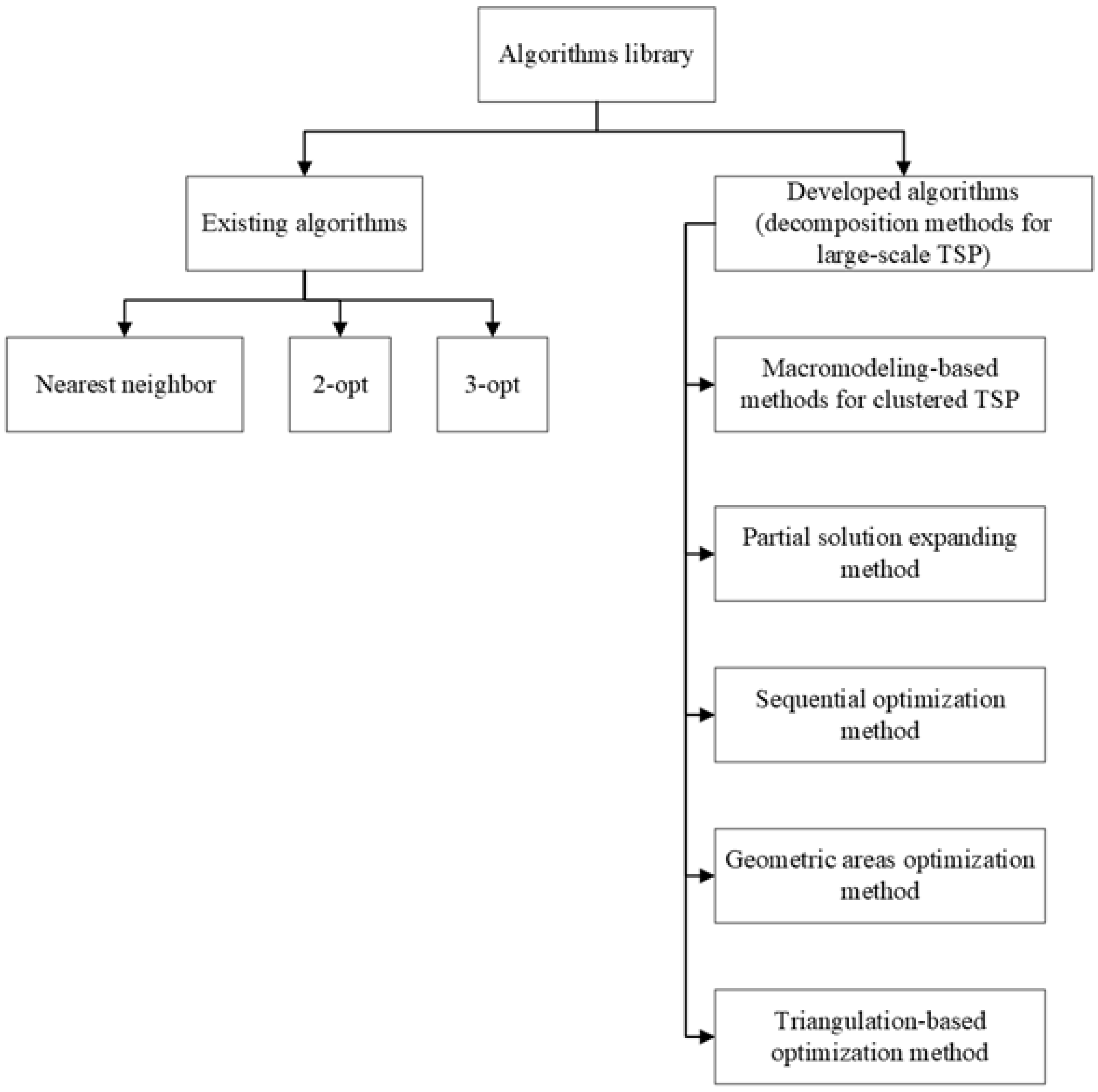
Algorithms library
Existing algorithms
Developed algorithms (decomposition methods for large-scale TSP)
Nearest neighbor
2-opt
3-opt
Macromodeling-based methods for clustered TSP
Partial solution expanding method
Sequential optimization method
Geometric areas optimization method
Triangulation-based optimization method

Figure 3 - Further details about Algorithms Library module

VRP Modeler includes functionally independent components (classes), each of them is designed to perform certain operations. To provide flexibility in expanding its functionality, the software consists of a main executable module VRPModeler and a set of additional dynamic modules. SQLite's database configuration file contains a list of all modules available in the current software version. Every additional module of VRP Modeler contains:

- description of the problems to be solved;
- input data processing module;
- problem solving methods module;
- input/output data visualization module.

Every project in VRP Modeler is a set of independent tests. In turn, every test is represented by a set of input data, input and output data visualizer type, and also an algorithm for solving a specific TSP.

The result can be saved as a graphic or text file (TSPLIB route). Saved route in TSPLIB format (.tour file) can be loaded for further optimization. After opening the project, the user can view pre-generated input data, as well as the result from the investigated method. Users can also generate new input data to the existing test, change its number, load input data into an existing test from file, or create new test in the current project with other components (for example, another algorithm, visualizer type, etc.).

Conclusions

The details of a software architecture for solving large scale TSP (more than 100,000 points) is presented. Using this software, (a) specialized decomposition methods for obtaining initial solutions as well as their optimization and (b) known heuristic methods (2-opt, 3-opt, etc.) for solving TSP can be implemented. The developed software can be used to solve large scale TSP while ensuring the solution quality within 0.5% above the optimum, and the solution can be generated in an acceptable amount of time.

References:

1. Applegate D. The Traveling Salesman Problem – A Computational Study / Applegate D.L., Bixby R.E., Chvatal V., Cook W. J. // Princeton Series in Applied Mathematics. – Princeton University Press. – 2006 . – 608 pp.
2. Cook W. In pursuit of the traveling salesman : mathematics at the limits of computation // Princeton University Press. – 2014. – 214 pp.
3. Concorde TSP Solver URL: https://www.math.uwaterloo.ca/tsp/concorde.html
4. LKH Version 2.0.10 (November 2022) URL: http://akira.ruc.dk/~keld/research/LKH/
5. Helsgaun K. An Efficient Implementation of k-Opt Moves for the Lin–Kernighan TSP Heuristic // Datalogiske Skrifter (Writings on Computer Science) . – 2006. – No. 109. – Roskilde University.
6. Helsgaun, K. General k-opt submoves for the Lin-Kernighan TSP heuristic. Math. Prog. Comput. – 2009 .- 1(2-3) .- pp. 119-163.
7. Helsgaun K., An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. Technical Report, Roskilde University, 2017
8. Bazylevych R. Decomposition and scanning optimization algorithms for TSP / Bazylevych R., Kutelmakh R., Prasad B., Bazylevych L. // Proceedings of the International Conference on Theoretical and Mathematical Foundations of Computer Science. – Orlando, USA. – 2008. – pp. 110-116.
9. Bazylevych R. A decomposition algorithm for uniform Traveling Salesman Problem / Bazylevych R., Prasad B., Kutelmakh R., Dupas R, Bazylevych L. // Proceedings of the 4th Indian International Conference on Artificial Intelligence. – Tumkur, India. – 2009. – pp. 47-59.

10. R. Bazylevych, M. Palasinski, R. Kutelmakh, B. Kuz, L. Bazylevych. “Decomposition methods for large-scale TSP”. Artificial intelligence methods and techniques for business and engineering applications”, ITHEA, Rzeszow – Sofia, 2012, pp. 148-157.

11. Roman Bazylevych, Marek Pałasiński, Roman Kutelmakh, Bohdan Kuz, Efficient decomposition algorithms for solving large-scale TSP. In book: G. Setlak, K. Markow. Computational models for business and engineering domains. ITHEA, Rzeszow-Sofia, 2014, pp. 225-234.

12. Bazylevych R., Kutelmakh R., Kuz B., Dupas R., B. Prasad, Y. Haxhimusa, L. Bazylevych, A Parallel Ring Method for Solving a Large-scale Traveling Salesman Problem, I.J. Information Technology and Computer Science, 2016, 5, pp. 1-12

Article sent: 20.04.2023