Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 44 No. 12
December 2023

On One Algorithm for Approximate Solution of the Traveling Salesman Problem

Mikhail Abramyan[1], Nikolai Krainiukov[2],Boris Melnikov[3]

[1, 2, 3]Shenzhen MSU-BIT University (China), Southern Federal University (Russia)

Abstract

The paper describes the results of numerical investigation of the efficiency of the contour ("onion husk") algorithm for approximate solution of the traveling salesman problem. The algorithm is based on the construction of convex closed contours on the initial set of nodes ("cities") and their subsequent merging into a closed path. Several versions of the contour algorithm are considered. Sets of randomly generated nodes, the number of which varies from 4 to 90, are used to test these versions. The results obtained are compared with the results of two well-known combinatorial optimization algorithms, namely the algorithm based on the branch-and-bound method and the simulated annealing algorithm. The possibility of applying the contour algorithm to the pseudogeometric traveling salesman problem, for which an analog of the distance matrix rather than a set of nodes is given, is also discussed.

Keywords:Branch-and-bound method, contour algorithm, simulated annealing method, traveling salesman problem.

1. Introduction

Combinatorial optimization problems and methods for their solution are constantly developing and improving; one of such problems is the traveling salesman problem (TSP), which belongs to classical NP-complete problems [1]. It should be noted that the combinatorial optimization methods used to solve this problem can be used more widely – for optimization, search and processing of complex data and other combinatorial objects.

2. Contour Algorithm and Its Modifications

The *contour algorithm* ("*onion husk" algorithm*) can be used for approximate solution of the so-called *geometric traveling salesman problem*, for which not only the distance matrix is given, but also the points ("cities") in the plane. The contour algorithm is based on the construction of nested convex contours connecting the initial points (Fig. 1) and the subsequent merging of these contours into a final closed path. The advantage of this algorithm is its high speed, while the obvious disadvantage is the approximate nature of the solution. At the same time, by applying additional heuristics at the main stage of the algorithm – the union of the initial convex contours – it is possible to achieve some improvement of the obtained results.

The simplest way to merge contours into a closed path is the sequential processing of neighboring contours, starting from the two outermost contours, where two adjacent points of each neighboring contour are connected by segments of minimum length (see Fig. 2, where the contour connection segments are represented by lines of greater thickness).

However, this method does not take into account the lengths of deleted segments of contours. Therefore, a more optimal variant is the one that minimizes the *SumLen* expression, in which the lengths of deleted segments are included with the "+" sign, and the lengths of added segments are included with the "–" sign. The result of applying this variant of the algorithm, in which the contours are searched from the outermost to the innermost (the *Up* algorithm), is shown in Fig. 3.

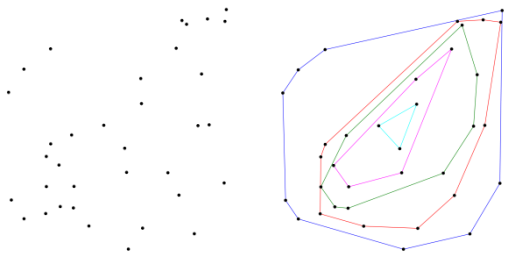

Figure 1. Set of points and their union into convex contours

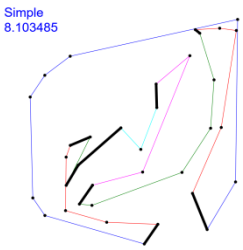

Figure 2. The simplest variant of contours merging

Figure 3. Improved variant of contours merging (the *Up* version)

Note that the choice of connecting segments reduces the number of possible variants for the next pair of contours. This may lead to the fact that for the next pair of contours the optimal connection variant will be lost. Therefore, we can consider two more modifications of the contour algorithm: in one of them (the *Down* modification) connections are built in the reverse direction (from the innermost to the outermost contour), and in the other (the *Mid* modification) at each step all pairs of neighboring contours that have not been merged yet are analyzed and the contours of the pair for which the value of the *SumLen* expression described above is minimal are merged.

Examples of application of *Down* and *Mid* modifications are shown in Fig. 4. For this set of points, all three modifications give different results, with the *Mid* algorithm giving the best result.

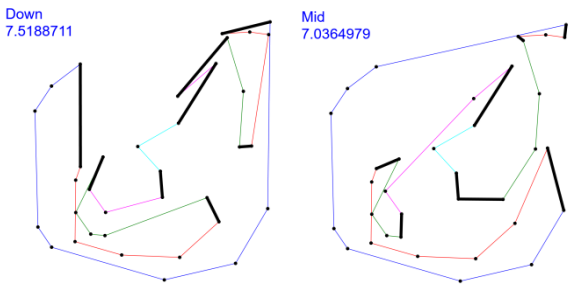

Figure 4.*Down* and *Mid* versions of the contour algorithm

3. Numerical Study of Modifications of the Contour Algorithm

Numerical study of the algorithms was performed on 20 randomly generated sets of 90 points located in a square with side 1. The initial parts of these sets consisting of 4, 6, 8, 10, 12, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80 and 90 points were analyzed. The previous figures showed an example of 35 points for one of the test sets.

The value $D_i = 100\ (S_i - S_0)/S_0$ was used to characterize the efficiency of the contour algorithm modifications, where S_0 is equal to the length of the path obtained by the simplest algorithm, and S_i is the length of the path obtained by one of the modifications of this algorithm (*Up*, *Down*, *Mid*). Thus, this value shows by how many percent the result has improved (i. e., *decreased*) compared to the simplest algorithm. The average values of D_i for the 20 test sets are summarized in Table 1; the results are also illustrated in Fig. 5.

On average, the *Mid* algorithm gives the best results, and the *Up* algorithm is the next most efficient. But the differences in the efficiency of these algorithms are insignificant even for a large number of points (no more than 1-2%).

4. Comparison of the Contour Algorithm with Other Approximate Algorithms for TSP

It has already been noted above that for the contour algorithm one cannot expect results close to the optimal ones. For a more accurate estimate of the applicability of this algorithm, we compared it with two known and quite effective approximate algorithms for TSP solving. The first one is based on a variant of the branch-and-bound method [2], [3], and the second one uses the simulated annealing algorithm [4]. Both of these algorithms are any-time algorithms and allow us to obtain the current pseudo-optimal solution at any time of their execution.

Table 1.Improvement (in percent) of the results of *Down*, *Up*, *Mid* algorithms compared to the simplest contour algorithm

Number of points	*Down*	*Up*	*Mid*
4	0.00	0.00	0.00
6	0.00	0.00	0.00
8	–2.95	–3.09	–3.09

10	–2.97	–3.40	–3.40
12	–2.44	–4.01	–4.01
15	–2.70	–3.56	–4.07
20	–5.51	–5.52	–5.52
25	–6.63	–7.07	–7.46
30	–6.07	–5.94	–6.91
35	–7.58	–7.02	–8.15
40	–8.40	–8.51	–9.16
50	–9.99	–10.68	–11.00
60	–10.60	–10.33	–10.93
70	–11.86	–12.38	–12.79
80	–11.67	–11.94	–12.30
90	–11.22	–11.87	–12.28

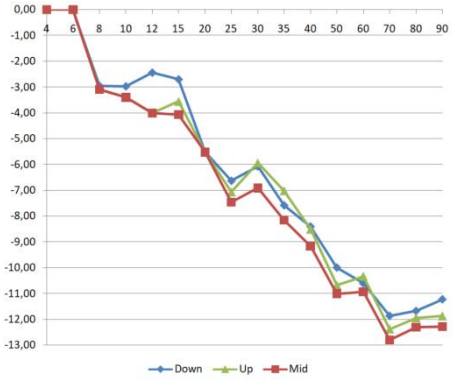

Figure 5. Comparative efficiency of *Down*, *Up*, *Mid* algorithms

Each of these algorithms was applied to the same test sets of points as the contour algorithm. For the algorithm based on the branch-and-bound method (*Branch-bound*), we used a limit on the maximum number of branches equal to 100000. A software version of the branch-and-bound method was implemented by one of the authors of this paper (B. Melnikov).

Simulated annealing is a stochastic minimization method based on random wandering through space at successively lower temperatures, where the probability of performing a step is determined by the Boltzmann distribution [4]. For the simulated annealing algorithm (*Anneal*), we used the function *gsl_siman_solve* of the GNU Scientific Library (GSL) standard library with the following parameters: initial temperature T_INITIAL = 5000, final temperature T_MIN = 5.0e–6, Boltzmann constant K_B = 1.0, temperature reduction factor MU_T = 1.03.

Fig. 6 shows the results of these algorithms for the set of 35 points previously considered for the contour algorithm.

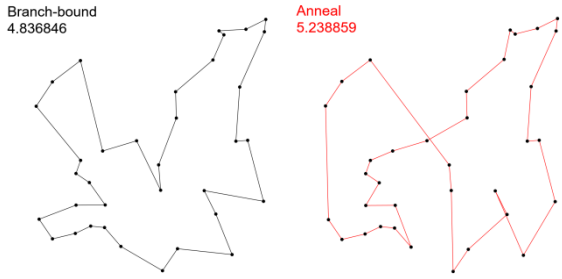

Figure 6. Paths obtained by *Branch-bound* and *Anneal* algorithms

The *Branch-bound* algorithm was used as the base algorithm, which showed the best results. The *Mid* algorithm was selected among the contour algorithms. To determine the relative efficiency of the *Anneal* and *Mid* algorithms compared to the *Branch-bound* algorithm, the previously described characteristic D_i was used, for which in this case the final path length obtained by the *Branch-bound* algorithm was used as S_0 and the path length obtained by the *Anneal* and *Mid* algorithms was used as S_i. Here, this characteristic is positive and shows by how many percent the results deteriorated (i.e., *increased*) compared to the *Branch-bound* algorithm. The average values of D_i for the 20 test sets are summarized in Table 2; the results are also illustrated in Fig. 7.

Table 2.Deterioration (in percent) of *Anneal* and *Mid* algorithms compared to the *Branch-bound* algorithm

Number of points	*Anneal*	*Mid*
4	0.00	0.00
6	0.00	1.46
8	0.00	7.94
10	0.00	14.30
12	0.00	18.71
15	0.00	27.73
20	1.00	36.99
25	2.72	47.05

30	6.71	51.35
35	4.81	58.67
40	3.98	63.07
50	9.45	71.31
60	10.86	77.49
70	11.25	78.21
80	12.34	83.62
90	14.15	85.77

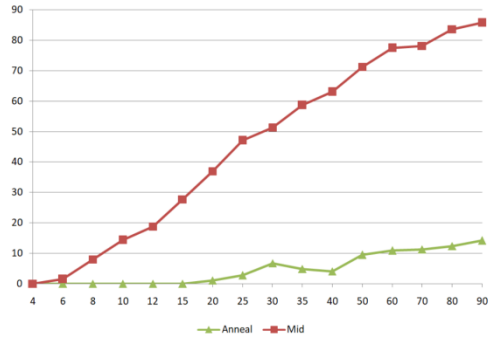

Figure 7. Efficiency of *Anneal* and *Mid* algorithms compared to *Branch-bound* algorithm

Although the *Branch-bound* and *Anneal* algorithms give better results, the deterioration of the *Mid* algorithm results does not exceed 90% even for 90 points. It should be expected that the inclusion of additional heuristics in the contour algorithm will reduce the gap between these algorithms.

5. On the Possibility of Applying the Contour Algorithm to the Pseudogeometric TSP

The contour algorithm uses information about the coordinates of points; thus, it can be applied to solve the *geometric TSP*. In contrast, the *pseudogeometric TSP* requires using only a matrix obtained by multiplying the elements of the "usual" distance matrix by a random variable generated on the basis of a normal distribution with the parameters $\mu = 1$ (mathematical expectation) and some value σ (standard deviation).

At small values of the parameter σ it is possible to implement the following two-stage algorithm for solving the pseudogeometric TSP: at the first stage, a set of points approximating the matrix of the pseudogeometric TSP is generated; at the second stage, the contour algorithm is applied to this set.

This paper does not consider issues related to the implementation of the first stage of this algorithm. However, if it is successfully implemented, we can expect that the results obtained will be quite close in efficiency to those obtained by applying the contour algorithm to the geometric TSP. As an illustration, we present Table 3, in which the *Geom* column gives the relative deterioration of the *Mid* algorithm compared to the *Branch-Bound* algorithm for the geometric TSP (this column coincides with the *Mid-Geom* column of Table 2), and the *Mid-Pseudogeom* column gives the relative deterioration of the solution obtained by the *Mid* algorithm for the original geometric TSP compared to the *Branch-bound* algorithm applied to the corresponding pseudogeometric TSP with the parameter $\sigma = 0.06$. The results are also shown in Fig. 8.

Table 3.Deterioration (in percent) of the *Mid* algorithm for geometric and pseudogeometric TSPs compared to the *Branch-bound* algorithm

Number of points	*Mid-Geom*	*Mid-Pseudogeom*
4	0.00	0.53
6	1.46	2.23
8	7.94	9.14
10	14.30	15.35
12	18.71	19.55
15	27.73	27.84
20	36.99	37.70
25	47.05	47.76
30	51.35	52.03
35	58.67	61.08
40	63.07	65.32
50	71.31	71.85
60	77.49	76.98
70	78.21	80.72
80	83.62	86.63
90	85.77	87.20

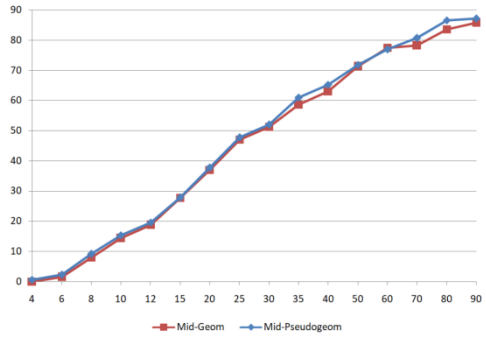

Figure 8. Efficiency of the *Mid* algorithm for the geometric and pseudogeometric TSPs compared to *Branch-bound* algorithm

References

[1] M. Garey, D. Johnson, "Computers and Intractability. A Guide to the Theory of NP-Completeness," San Francisco, CA, USA: W. H. Freeman & Co., 1979.

[2] S. Goodman, S. Hedetniemi, "Introduction to the Design and Analysis of Algorithms," New York, NY, USA: McGraw-Hill, 1977.

[3] B. Melnikov, E. Melnikova, "On the Classical Version of the Branch and Bound Method," Computer Tools in Education, no. 2. pp. 41-58, 2022.

[4] E. H. L. Aarts, J. Korst, "Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing," Wiley, Chichester, 1989.