# Exact time-efficient combined algorithm for solving the asymmetric traveling salesman problem[1]

**Galina N. Zhukova**
*Associate Professor, School of Software Engineering*
*National Research University Higher School of Economics*
*Address: 20, Myasnitskaya Street, Moscow, 101000, Russia*
*E-mail: galinanzhukova@gmail.com*

**Mikhail V. Ulyanov**
*Leading Researcher, Laboratory of Scheduling Theory and Discrete Optimization*
*V.A. Trapeznikov Institute of Control Sciences, Russian Academy of Sciences;*
*Professor, Department of Algorithmic Languages*
*Lomonosov Moscow State University*
*Address: 65, Profsoyuznaya Street, Moscow, 117997, Russia*
*E-mail: muljanov@mail.ru*

**Mikhail I. Fomichev**
*Graduate of System and Software Engineering MSs Program*
*National Research University Higher School of Economics*
*Address: 20, Myasnitskaya Street, Moscow, 101000, Russia*
*E-mail: mikhail.fomichev94@gmail.com*

**Abstract**

For practical, important tasks in the fields of economics and logistics, as well as in a number of technical applications, it becomes necessary to solve the traveling salesman problem (TSP). Quite often, the features of these problems lead to the traveling salesman problem in asymmetric formulation (asymmetric traveling salesman problem, ATSP). Moreover, in some practical applications it is desirable to obtain an exact solution. One of the known exact algorithms for solving the ATSP is an algorithm that implements the well-known branch and bound method. The known experimental estimates of its complexity on the average are exponential. However, this does not mean that for small dimensions of the problem (currently, no more than 70–75), the expected time for solving the individual problem is unacceptably high. The need to reduce the time for solving individual problems dictated by practice is associated with the use of various modifications of this algorithm, of which a modification that involves storing truncated matrices in the search decision tree is one of the most effective. In this article, the authors rely on this modification. Other possible improvements in the time efficiency of the software implementation of the branch and bound method are related, among other things, to obtaining the initial approximation by heuristic algorithms. As a result, we get a combined algorithm, in which, at the first stage, some heuristics works to obtain the initial solution, from which the branch and bound method starts. This idea has been discussed for a long time, but the problem is that to reduce time, such a heuristic algorithm is needed that delivers a solution close to optimal which will be found quite fast. One of the possible solutions to this problem is the subject of this article.

[1] This work was supported by the Russian Foundation for Basic Research (project No. 16-07-00160 "Forecasting of time characteristics of efficient implementations of the branch and bound method for the traveling salesman problem relying on characteristics of random matrixes and identification of generated times distribution")

The subject of the research in this article is the choice of the best heuristic algorithm which, when applied, leads to an increase in temporal efficiency in combination with the algorithm of the branch and bound method, and an experimental study of its software implementation in order to obtain an average time for solving individual problems. On the basis of the results obtained, recommendations are given on the limiting dimensions of the problem that allow for an acceptable solution time, something which is of interest in the practical application of this combined algorithm in the tasks of business informatics and logistics.

**Key words:** travelling salesman problem; branch and bound method; combined algorithm; time efficiency; experimental research.

**Citation:** Zhukova G.N., Ulyanov M.V., Fomichev M.I. (2018) Exact time-efficient combined algorithm for solving the asymmetric traveling salesman problem. *Business Informatics*, no. 3 (45), pp. 20–28. DOI: 10.17323/1998-0663.2018.3.20.28.

## Introduction

The traveling salesman problem (TSP) can be formulated as follows: the salesman is to find the cheapest tour between $n$ cities, visiting each city once and only once (except origin) and returning to the city of origin. We call a tour the sequence of all cities which the salesman is to visit. Instead of 'the travel cost,' one can use distance, time or other indices. The cost of travel is known for all pairs of cities. In the mathematical field of graph theory, the TSP is defined in the following way: what is a minimal Hamiltonian cycle in a complete weighted graph. This graph is represented by an adjacency matrix $C = (c_{ij})$, which is called the cost matrix. The diagonal elements of the adjacency matrix are infinitely large numbers, because the graph of the road net does not contain self-loops. In the general case, the graph is directed. The Hamiltonian cycle of a graph we call a tour. The solution of the TSP is a tour with a minimum sum of arc weight. In such a formulation, the Traveling Salesman Problem belongs to the class of NP-hard problems. That is why it relates to the intractable problems of combinatorial optimization.

The TSP admits a variety of practical important interpretations. For example: scheduling the operation of equipment with reconfigurations, optimization of crane operations, sequencing of burning slots in the manufacture of chips [1]. The cabling of computer networks, predicting protein function and representation of black and white images by a continuous line without intersections can also be reduced to the TSP [2].

Presented in 1963 [3], the first exact algorithm for solving TSP was based on the branch and bound (B&B) method [4]. A detailed description of the earlier works of the Traveling Salesman Problem can be found in [5]. The present methods and approaches to the TSP are described in [6]. The issues of increasing the accuracy of the lower estimate of the cost of the tour can be found in article [7]. We do not give a description of the classical branch and bound algorithm for the TSP which can be found, for example, in [3; 8; 9].

Since a number of practical problems in the field of business informatics, logistics and economic optimization are reduced to the TSP, there are an abundance of heuristic methods for solving it. But this does not mean that there is no need for exact solutions to the problem. Consequently, one may ask:

✦ what is the biggest dimension value of the problem that can be solved exactly within an acceptable time?

✦ how can we increase this dimension value by the use of a modified exact algorithm with better time efficiency?

Since the 1960s, researchers have used a heuristic algorithm for finding optimal or near-optimal solutions (tours) for the individual TSP and then a start branch and bound algorithm with the

initial solution. It is intuitively clear that such an approach should reduce the number of vertices of the generated decision tree and therefore the time of finding the exact solution. In this case, we get a combined algorithm which contains a heuristic algorithm and B&B. The present article is devoted to the description and the computational analysis of the combined algorithm with the Linn–Kernigan method as a heuristic part.

## 1. Definition of the task

We need to find a heuristic algorithm for the asymmetric traveling salesman problem that can help to reduce the total time of solving TSP by starting the branch and bound algorithm with the initial solution, hereinafter called the precomputed tour.

We introduce the following notations:

$n$ – dimension of the problem (number of vertices of the complete graph);

$C = (c_{ij})$ – cost matrix of the individual asymmetric traveling salesman problem that is an adjacency matrix of a directed graph without loops;

$T_e$ – cost of a precomputed tour which is found by a heuristic algorithm;

$t_{BB}(C, T_e)$ – the running time of the software implementation of the branch and bound algorithm (in a certain hardware configuration) for the individual TSP, which is defined by cost matrix $C$ with a precomputed tour of cost $T_e$;

$t_E(C)$ – the running time of a heuristic algorithm;

$N_0(C)$ – the number of vertices of the search decision tree generated by the classical branch and bound algorithm (without a precomputed tour);

$N_1(C)$ – the number of vertices of the search decision tree generated by the branch and bound algorithm with the precomputed optimal tour;

$\overline{t}_{BB0}(n)$ – sample average running time for software implementation of the classical branch and bound algorithm without a precomputed tour when the sample consists of individual problems with the same dimension $n$;

$\overline{t}_{BB1}(n)$ – sample average running time for software implementation of the branch and bound algorithm with a precomputed tour when the sample consists of individual problems with the same dimension $n$;

$\overline{t}_E(n)$ – sample average running time for software implementation of the heuristic algorithm.

Our goal is to find such a heuristic algorithm that delivers the initial tour that decreases the running time of the B&B so that

$$t_{BB}(C, T_e) + t_E(C) < t_{BB}(C, \infty) \qquad (1)$$

for most cost matrices $C$.

Since the branch and bound algorithm is highly sensitive to the features of individual problems, condition (1) does not mean the time reduction for every individual task. Thereafter, we apply the sample average:

$$\overline{t}_{BB1}(n) + \overline{t}_E(n) < \overline{t}_{BB0}(n). \qquad (2)$$

It is also of interest to find the threshold value $N$ of the dimension of the problem such that a combined algorithm is more effective than the classical B&B when the TSP dimension $n$ is more than $N$. In order to predict the time efficiency, it is also necessary to approximate a dependence of the average running time on the dimension of the TSP.

## 2. How to reduce the running time of the branch and bound algorithm

In order to decrease the running time of algorithms that implement the idea of the branch and bound method for solving the traveling salesman problem, various approaches are proposed. Some of them use storing reduced matrices in the nodes of the search decision tree [10]. There are also several combinations of exact and heuristic algorithms [11–13].

The paper [10] presents an experimental study of the impact of additional memory allo-

cation for the storage of truncated cost matrices in the nodes of the search decision tree in the range of TSP of dimension from 25 to 45. *Table 1* presents a forecast based on the experimental data [10]. This means that the software implementation of the algorithm with the storing matrices can be effectively used for finding the exact solution of the TSP of a dimension that is not more than 70 at an available RAM of 16 GB with the expected average calculation time of modern personal computers on the order of magnitude of one minute.

In developing a combination of the branch and bound algorithm and the heuristic algorithms that deliver a precomputed tour, we will further use an implementation that does not include the storage of truncated cost matrices at the vertices of the search decision tree.

The authors of the study [12] sought to level out the shortcomings of the branch and bound algorithm with heuristic algorithms and techniques for parallelizing program flows. However, they also pointed out that they clearly understood the fact that the results obtained are not reliable and had to be treated with great care.

### 3. The influence of the quality of a precomputed tour on the number of generated vertices

In the classical branch and bound algorithm for solving the TSP, there is no opportunity to begin cutting out the subtrees of the search decision tree until a tour is found. This is because the classical algorithm does not assume the presence of any tour at the time of the initial launch. In view of the particular nature of the problem, the decision tree can seriously grow until the first tour is found. Using a precomputed tour (which is known before the B&B algorithm is launched) can reduce the size of the search decision tree. Under the condition that a precomputed tour is close to the optimal one, a combined approach allows for less time to be spent on finding an optimal tour. That is because there is no need to create and visit unfavorable vertices of the search decision tree. The amount of additional memory required also decreases, since there is no need to store unfavorable vertices of the search tree.

Thus, the combined approach aims at reducing both the running time and the amount of memory required. However, the implementa-

*Table 1.*

**Resource characteristics forecast**

| Dimension | Prediction of optimal tour calculation time without additional memory | Prediction of optimal tour calculation time with additional memory | Time ratio prediction | Prediction of the average amount of additional memory required |
|---|---|---|---|---|
| 45 | 1 s | 0.2 s | 5 | 30.71 MB |
| 54 | 7 s | 1 s | 7 | 172.3 MB |
| 70 | 11.7 min | 1 min | 11.7 | 12.47 GB |
| 80 | 2.5 h | 10 min | 15 | 136.37 GB |
| 88 | 19.6 h | 1 h | 19.6 | 924.26 GB |
| 102 | 29.5 days | 1 day | 29.5 | 25.69 TB |

tion of the combined algorithm causes some problems. Obviously, the closer a precomputed tour is to an exact solution, the less time it takes to find the optimal solution by B&B. On the other hand, a precomputed tour must be found quickly. This means that the running time of the classical branch and bound algorithm should at least be no more than the total running time of the heuristic algorithm and the branch and bound algorithm with the precomputed tour. In other words, the use of a precomputed tour should be justified and rational.

According to [14], if a precomputed tour is optimal, then the search decision tree size is reduced by approximately 40%. The average number of generated vertices of the search decision tree for problems (of different dimensions), obtained experimentally in [14] is presented in *Table 2*. The characteristic $\eta = (1 - N_1 / N_0)$ shows the decrease of the size of the search decision tree by percentage if the precomputed tour for the branch and bound algorithm is optimal.

However, the sensitivity of the Branch and Bound algorithm to the quality of a precomputed tour is sufficiently high: if a precomputed tour is more than 5% larger than the optimal one, then, as was shown in [14], the search decision tree cannot be significantly reduced.

*Table 2.*

**The influence of the precomputed tour on the number of generated vertices**

| $n$ | $\eta$ |
|---|---|
| 35 | 38% |
| 40 | 38% |
| 45 | 39% |

## 4. Heuristic algorithms for calculating a precomputed tour

Heuristic algorithms are algorithms which do not necessarily deliver an exact solution. However, the solutions which can be found by these algorithms are usually quite close to an optimal one; moreover, they are available in a "reasonable" time [15]. Unlike exact algorithms, heuristic algorithms are usually fairly simple to implement, and they work faster. All heuristic algorithms can be divided into the following three types:

- ✦ greedy [16];
- ✦ swarm intelligence [17–19];
- ✦ improvement [20, 21].

In addition, there are many different algorithms that are designed to solve particular cases of the TSP (for example, the metric traveling salesman problem [22]).

Preliminary analysis of the literature sources [14, 23, 24] allows us to conclude that representatives of greedy and swarm intelligence heuristic algorithms cannot provide a sufficiently high-quality solution for the time required in our formulation of the problem. Therefore, the use of the initial tours obtained by these algorithms is not reasonable. The situation is different with a lot of heuristic algorithms that improve solutions.

In 1973, S. Lin and B. Kernigan presented an effective heuristic algorithm for the traveling salesman problem (the Lin–Kernighan algorithm) [25]. It is based on the idea of an iterative improvement of a randomly found tour. As shown by experimental results, this algorithm often even finds globally optimal solutions. At the same time, the complexity of the algorithm is approximately $O(n^{2,2})$ [25].

Later, in 2000, K. Helsgaun proposed a modified implementation of the algorithm (the Lin–Kernighan–Helsgaun algorithm) [26]. This algorithm often provides an optimal solution in an acceptable time, even for problems of large dimension.

The algorithms considered are designed to solve the symmetric traveling salesman problem. However, using the improved method described in paper [27], any asymmetric trave-

ling salesman problem (dimension $n$) can be reduced to the symmetric traveling salesman problem (dimension $2n$). Unfortunately, this transformation also affects the time of solving the asymmetric traveling salesman problem by the Lin–Kernighan–Helsgaun algorithm.

The main idea of the Lin–Kernighan–Helsgaun algorithm is to transform a feasible solution by the replacement of some set of its arcs to another, which delivers a better feasible solution. The process goes on until there is an existing set to replace it with. All details of the algorithm are given in [26].

The efficiency of the Lin–Kernigan–Helsgaun algorithm is achieved, first of all, due to the effective strategy of the search sets of arcs described above. The search is based on the restriction of 5-opt replacements inside the set of possible candidates [26].

The author of [28] attempted to develop an algorithm that provides an optimal or very close to optimal solution relatively quickly. However, he did not pay attention to the complexity of the implementation, and as a result, the last software implementation presented by the author consisted of about 10,000 lines of source code [28].

## 5. Combined algorithm and experimental results

To analyze the combination of the branch and bound algorithm with a heuristic Lin–Kernigan–Helsgaun algorithm, an experimental study of asymmetric traveling salesman problems of dimension 35, 37, 40, 43, and 45 was carried out. The sample of TSP's of the same dimension consisted of 100,000 individual problems (for each dimension value).

The experiments were carried out on a personal computer with the following characteristics:

- processor: Intel i7 8700K 4700 MHz;
- RAM: Corsair Vengeance LPX CMK 32GX4M2B3466C16R DDR4 3466 MHz 32 GB;
- motherboard: ASRock Fatal1ty Z370 Gaming K6;
- operating system: Arch with kernel version 4.14.13-1-ARCH.

To minimize operating system noise, background processes (for example, firewalls) unnecessary for the research were disabled, and the distribution did not contain the components of the graphical interface (we used command line interface instead). Moreover, swapping was disabled, so that the speed of the HDD did not affect the running time of the implementation of the algorithm.

The algorithms was implemented in C++ and compiled into an executable program using the compiler gcc 7.2.1 20171224.

The combined algorithm was implemented in C++ and compiled into an executable program using the compiler gcc 7.2.1 20171224.

We introduce the notation $\overline{t}_{LKH}(n)$, which means the average running time of the software implementation of the Lin–Kernighan–Helsgaun algorithm where $n$ denotes the dimension of TSP.

Average, minimal and maximal running time of the B&B algorithm implementation without a precomputed tour are presented in the first part of *Table 3* ($\overline{t}_{BB0}(n)$, $\breve{t}_{BB0}(n)$, $\hat{t}_{BB0}(n)$ correspondingly, the sample size is 100,000).

The same characteristics of the implementation of the combined algorithm with a precomputed tour calculated by the Lin–Kernighan–Helsgaun algorithm are denoted as $\overline{t}_{BB1}(n)+\overline{t}_{LKH}$, $\breve{t}_{BB1}(n)+\breve{t}_{LKH}$, $\hat{t}_{BB1}(n)+\hat{t}_{LKH}$. These figures are presented in the second part of *Table 3* (the sample size is 100,000).

The experimental results obtained do not show such a significant reduction of the running time of the implementation of B&B combined with Lin–Kernighan–Helsgaun algorithm as can be expected based on *Table 2*. The reduction of the number of generated vertices of the decision tree by 38% does not provide a similar decrease of the running time of the

*Table 3.*

**Experimental results**

| $n$ | $\bar{t}_{BB0}(n)$ $\mu s$ | $\breve{t}_{BB0}(n)$ $\mu s$ | $\hat{t}_{BB0}(n)$ $\mu s$ | $\bar{t}_{BB1}(n)+\bar{t}_{LKH}$ $\mu s$ | $\breve{t}_{BB1}(n)+\breve{t}_{LKH}$ $\mu s$ | $\hat{t}_{BB1}(n)+\hat{t}_{LKH}$ $\mu s$ |
|---|---|---|---|---|---|---|
| 35 | 78 009 | 312 | 4 866 390 | 84 080 | 3 629 | 4 200 069 |
| 37 | 128 072 | 357 | 63 857 740 | 133 791 | 3 691 | 39 814 733 |
| 40 | 261 729 | 435 | 34 998 093 | 264 468 | 3 840 | 26 352 357 |
| 43 | 539 085 | 504 | 66 511 234 | 531 160 | 5 127 | 58 208 012 |
| 45 | 859 599 | 578 | 123 629 945 | 831 424 | 6 232 | 78 427 649 |

implementation of B&B. The reason is that calculation of the lower bounds of the vertices of the decision tree takes time, but when the estimates are greater than the value of the precomputed tour no new vertices are created. In this case, the running time is spent without the creation of a new vertex of the search decision tree.

Approximation of the obtained experimental results by the method of least squares can be represented as

$$\bar{t}_{BB0}(n) = 17.783 \cdot e^{0.2399n},\ R^2 = 0.9999, \quad (3)$$

$$\bar{t}_{BB1}(n) + \bar{t}_{LKH}(n) = 27.552 \cdot e^{0.2293n},\ R^2 = 0.9999. \quad (4)$$

The abscissa of the intersection point of these exponents is close to the integer point $n = 43$. Taking into account (3) and (4), we concluded that when the dimension of the TSP is more than $n = 43$, the combination of the branch and bound algorithm with the Lin–Kernigan–Helsgaun algorithm delivers an exact solution faster (on average) than the classical B&B. In other words, the sample average running time of B&B with the Lin–Kernigan–Helsgaun algorithm is less than the sample average running time of B&B without an initial tour. The formula (4) provides the values of the dimension of TSP which can be solved using a combined algorithm in time $t_{\max}$ or less. In this case, the largest dimension of the TSP is the solution of the equation

$$27.552 \cdot e^{0.2293n} = t_{\max} \quad (5)$$

For example, when $t_{\max} = 6 \cdot 10^8$ microseconds (i.e. 5 minutes), a combined algorithm can deliver an exact solution of TSP of dimension up to $n = 73$. However, we should not forget that the method is very sensitive to the features of the individual problems, and the amount of running time of some TSP of dimension 73 can be significantly larger than $t_{\max} = 6 \cdot 10^8$ microseconds. Moreover, the requirement for additional memory can be significant.

The authors are aware that the accuracy of the extrapolation obtained is not high, and we have a fairly rough approximation. In general, we hope that when the dimension of the TSP is more than 70–75, the combined algorithm can solve the TSP significantly faster (on average), but we do not state that it is true for all individual tasks.

## Conclusion

Thus, on the basis of the experimental research we conducted, the following conclusions can be made:

✦ according to the calculated trend, when

the dimension of the TSP is more than $n = 43$, the combined algorithm of B&B and the Lin–Kernigan–Helsgaun algorithm works faster than the classical branch and bound algorithm for the asymmetric traveling salesman problem (on average);

✦ the branch and bound algorithm with a precomputed tour does not provide a solution to the traveling salesman problem in polynomial time, however the use of an initial tour reduces the coefficient at $n$ in the exponent in (3) by 4.4%, which is also a significant result;

✦ it is not efficient to use greedy or swarm intelligence heuristic algorithms for finding an initial tour, because the total time of the operation of the branch and bound algorithm and those heuristic algorithms is much more than the classical branch and bound algorithm (without a precomputed tour).

The authors see the further development of the study in a more detailed analysis of the distributions of running time of software implementations of the algorithms. ■

## References


1. Borodin V.V., Loveckiy S.E., Melamed I.I., Plotinskiy U.M. (1980) Ehksperimentalnoe issledovanie ehffektivnosti ehvristicheskih algoritmov resheniya zadachi kommivoyazhera [Experimental research of the effectiveness of heuristic algorithms for solving the traveling salesman problem]. *Automation and Remote Control*, no. 11, pp. 76–84 (in Russian).
2. Kolesnikov A.V., Kirikov I.A., Listopad S.V., Rumovskaya S.B., Domanitsky A.A. (2011) *Reshenie slozhnyh zadach kommivoyazhera metodami funkcionalnyh gibridnyh intellektualnyh system* [Solution of complex traveling salesman problems using the methods of functional hybrid intelligent systems]. Moscow: Institute of Informatics Problems (in Russian).
3. Little J.D.C., Murty K.G., Sweeney D.W., Karel C. (1963) An algorithm for the traveling salesman problem. *Operations Research*, no. 11, pp. 972–989.
4. Land A.H., Doig A.G. (1960) An automatic method of solving discrete programming problems. *Econometrica*, vol. 28, no. 3, pp. 497–520.
5. Slominski L. (1982) Probabilistic analysis of combinatorial algorithms: A bibliography with selected annotations. *Computing*, no. 28, pp. 257–267.
6. Matai R., Mittal M.L., Singh S. (2010) Travelling salesman problem: An overview of applications, formulations and solution approaches. *Travelling salesman problem: Theory and applications* (ed. D. Davendra). Rijeka: Intech Open Access Publisher, pp. 1–24.
7. Toriello A. (2014) Optimal toll design: A lower bound framework for the asymmetric traveling salesman problem. *Mathematical Programming*, vol. 144, no. 1–2, pp. 247–264.
8. Goodman S.E., Hedetniemi S.T. (1981) *Vvedenie v razrabotku i analiz algoritmov* [Introduction to the design and analysis of algorithms]. Moscow: Mir (in Russian).
9. Korte B., Vygen J. (2007) *Combinatorial optimization: Theory and algorithms*. Springer.
10. Ulyanov M.V., Fomichev M.I. (2015) Resource characteristics of ways to organize a decision tree in the branch-and-bound method for the traveling salesmen problem. *Business Informatics*, no. 4 (34), pp. 38-46.
11. Oliver I., Smith D., Holland J. (1987) A study of permutation crossover operators on the traveling salesman problem. Proceedings of the *2nd International Conference on Genetic Algorithms. Cambridge, MA, USA, 28–31 July 1987*, pp. 224–230.
12. Cotta C., Aldana J., Nebro A., Troya J. (1995) Hybridizing genetic algorithms with branch and bound techniques for the resolution of the TSP. *Artificial neural nets and genetic algorithms 2* (eds. D. Pearson, N. Steele, R. Albrecht). Wien, New York: Springer-Verlag, pp. 277–280.
13. Goldberg D., Lingle R.J. (1985) Alleles, loci, and the travelling salesman problem. Proceedings of the *First International Conference on Genetic Algorithms and Their Applications. Pittsburgh, 24–26 July 1985*,

pp. 154–159.

14. Fomichev M.I. (2017) Sravnitelnyi analiz metaehvristicheskih algoritmov resheniya nesimmetrichnoj zadachi kommivoyazhera [Comparative analyses of metaheuristic algorithms of solving asymmetric traveling salesman problem]. *Control Systems and Information Technologies*, no. 3 (69), pp. 88–92 (in Russian).
15. Panteleev A.V. (2009) *Metaevristicheskie algorithmi poiska globalnogo ekstremuma* [Metaheuristic algorithms for search global extremum]. Moscow: MAI-Print (in Russian).
16. Applegate D.L., Bixby R.E., Chvatal V., Cook W.J. (2006) *The traveling salesman problem: A computational study.* Princeton, NJ: Princeton University Press.
17. Colorni A., Dorigo M., Maniezzo V. (1991) Distributed optimization by ant colonies. Proceedings of the *First European Conference on Artificial Life (ECAL 91). Paris, France, 11–13 December 1991*, pp. 134–142.
18. Dorigo M., Gambardella L.M. (1997) Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions of Evolutionary Computation*, vol. 1, no. 1, pp. 53–66.
19. Bonavear E., Dorigo M. (1999) *Swarm intelligence: From natural to artificial systems.* Oxford, UK: Oxford University Press.
20. Gamboa D., Rego C., Glover F. (2005) Data structures and ejection chains for solving large scale traveling salesman problems. *European Journal of Operational Research*, vol. 160, no. 1, pp. 154–171.
21. Kaplan H., Lewenstein M., Shafrir N., Sviridenko M. (2005) Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *Journal of the ACM*, vol. 52, no. 4, pp. 602–626.
22. Mömke T., Svensson O. (2016) Removing and adding edges for the traveling salesman problem. *Journal of the ACM*, vol. 63, no. 1, article no. 2.
23. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. (2009) *Introduction to algorithms.* Cambridge, MA: MIT Press.
24. Stutzle T., Hoos H.H. (2000) MAX-MIN ant system. *Future Generation Computer Systems*, no. 16. pp. 889–914.
25. Lin S., Kernighan B.W. (1973) An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, vol. 21, no. 2, pp. 498–516.
26. Helsgaun K. (2000) An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130.
27. Jonker R., Volgenant T. (1983) Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, no. 2, pp. 161–163.
28. Helsgaun K. (2017) *An extension of the Lin–Kernighan–Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical Report.* Roskilde: Roskilde University.