Study on Solving the TSP Using the Slime Mold Algorithm Including Partial Solutions

Mengchun Xie
Department of Electrical and Computer Engineering
National Institute of Technology, Wakayama College
JAPAN
Koki Furuya
Department of Computer Science and Engineering
Toyohashi University of Technology
JAPAN
Mitsutoshi Murata
Department of Electrical and Computer Engineering
National Institute of Technology, Wakayama College
JAPAN

Abstract
The network constructed by a slime mold in the natural world exhibits the shortest route between several food sources. A method has been proposed that translates the properties of a slime mold into a mathematical model, which is applied to the traveling salesman problem (TSP) to obtain an optimal solution. However, some weaknesses exist, such as that the processing time increases significantly with an increase in the number of nodes in the slime mold algorithm. In this paper, a new slime mold algorithm including partial solutions is proposed to improve the efficiency of searching for solutions to the TSP. The proposed method divides the TSP area into several partial areas and searches for partial solutions, which are then combined into an approximate solution. We perform experiments on a computer for the TSPLIB benchmark problems to compare the proposed method with the conventional approach.

Keywords: Slime Mold Algorithm, travelling salesman problem, dividing search area, integrating partial solutions

Introduction

A combinatorial optimization is a search for an optimal object from among a finite set of objects. Unfortunately, most interesting combinatorial optimization problems are NP-hard. In many such problems, an exhaustive search is not feasible [Cook1997]. For a combinatorial optimization problem such as the travelling salesman problem (TSP), the number of combinations representing possible solutions increases explosively as the scale of problem becomes larger, making it is hard to obtain the optimum solution. An approximation algorithm returns a solution to a combinatorial optimization problem that is provably close to optimal. Approximation algorithms are typically used when finding an optimal solution is intractable, but can also be used in some situations where a nearoptimal solution can be found quickly and an exact solution is not needed [Cook2016, Roberto 2016, Yannakais1994].

There has been an enormous amount of study of various types of heuristics and metaheuristics such as annealing heuristics, genetic algorithms (GAs), and tabu search which are based on procedures designed to cross boundaries of feasibility or local optimality, instead of treating them as barriers [Michael2017, Xie1996, Xie2015, Glover1993]. Reference []Tamura1994] proposed a hybrid approach of a GA and the Lagrange relaxation method (LR) where the solution space of the given problem is partitioned into several small subspaces and only promising subspaces are searched to obtain a suboptimal solution. Reference [Valejo2018] proposed a multilevel approach for combinatorial optimization in a bipartite network which consists of iteratively coarsening the original network into a hierarchy of smaller sized approximations. Recently, a slime mold algorithm has drawn much attention [Valejo2018].

Many biological systems are composed of unreliable components which selforganize effectively into systems that achieve a balance between efficiency and robustness. One such example is the true slime mold, which is an amoeba-like organism that seeks and connects food sources and efficiently distributes nutrients throughout its body [Li2011]. Experimental observations have shown that slime mold will connect multiple separate nutrient sources into a variety of final configurations while keeping the total tube length short. A method called an amoeba-based computer (ABC) was proposed to quantitatively evaluate the optimization capability of the amoeboid organism in searching for a solution to the traveling salesman problem under optical feedback control [Zhu2013]. ABC can find a high-quality solution for the 8-city TSP with a high probability. However, the ABC has some weaknesses, such as that the processing time increases significantly with an increase of the number of nodes.

In this paper, in order to improve the efficiency of searching in TSP, a new slime mold algorithm including partial solutions (pSMA) is proposed. The proposed method divides the TSP area into several partial areas and searches for partial solutions, which are then combined into an approximate solution. We perform experiments on a computer for the benchmark problems in TSPLIB [TSPLIB2018] to compare the proposed method with the conventional approach.

This paper is organized as follows. Section 2 describes the target problems and the simulation of slime mold. Section 3 introduces the pSMA the proposed method. In Section 4, based on the results of numerical experiments, the effectiveness of the proposed method is examined. The paper finishes with a short conclusion.

TSP and the simulation of slime mold

Traveling Salesman Problem

The traveling salesman problem (TSP) has many applications in operations research. It also has a nice structure and is one of the most investigated NP-complete problems of combinatorial optimization [Kovacs2018]. The travelling salesman problem (TSP) is to find a shortest possible route that visits each city and returns to the origin city, given a list of cities and the distances between each pair of cities.

In this paper, the symmetric TSP is set as the target problem. In the symmetric TSP, the distance between two cities is the same in both directions, corresponding to an undirected graph.

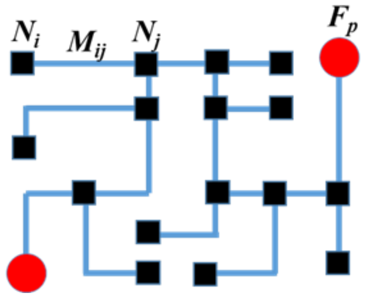
Simulation of slime mold

Slime molds are fungus-like single-celled organisms that can grow into a network of linked veins, spreading over a surface like a web. A tubular network of slime mold is generated to search for and connect food sources. A slime mold can find the shortest path through a maze or connect different arrays of food sources in an efficient manner with a low total length, shorter than the average minimum distance between pairs of food sources [Tero2010]. Slime mold has evolved to grow in the most efficient way possible in order to maximize its access to nutrients [Takamura2009].

Slime mold forms networks with comparable efficiency, fault tolerance, and cost to those of real-world infrastructure networks. The core mechanisms needed for adaptive network formation can be captured in a biologically inspired mathematical model that may be useful to guide network construction in other domains [Tero2010]. This mathematical model is called the slime mold algorithm (SMA) here.

The SMA is represented using an undirected weighted graph such that the tubes of the slime mold are the graph’s edges, the amounts of nutrients are the edges’ weights, and the connecting points are the graph's nodes (Figure 1). For example, N_i is the *i-th* node, M_{ij} is the edge between N_i and N_j, and F_p is the *p-th* food source node.

Figure 1 Example of simulating a slime mold network by a graph



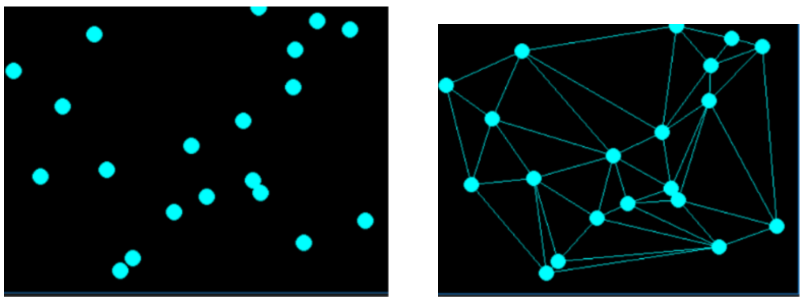
Slime Mold Algorithm Including Partial Solutions for TSP

As mentioned above, the conventional slime mold method has some weaknesses, such as that the processing time increases significantly with an increase in the number of nodes, which is why we propose our new method, the slime mold algorithm including partial solutions (pSMA). The proposed algorithm is composed of the following steps: (1) generate a mesh; (2) divide the search area; (3) search for a partial solution for each partial area by the conventional slime mold method; (4) integrate that partial solutions to obtain an approximate solution; and (5) repeat the process of division, search, and integration to obtain sub-optimal solutions.

Mesh generation

Mesh generation is performed by the Delaunay triangulation method (Figure 2). Delaunay's triangulation is to form a mesh comprising triangles whose vertices lie on a circumcircle that does not contain any other vertices. Because a Delaunay triangle has no vertices within its circumcircle, a Delaunay triangulation is an ideal search structure for finding vertices that are far from other vertices [Shewchuk2014]. In the method, given a set of points in a plane (Figure 2(a)), triangles are created by the Delaunay triangulation method (Figure 2(b)) for all points.

Figure 2 Meshes generated by Delaunay triangulation method (a) set of points (b) Delaunay triangle



Dividing search area

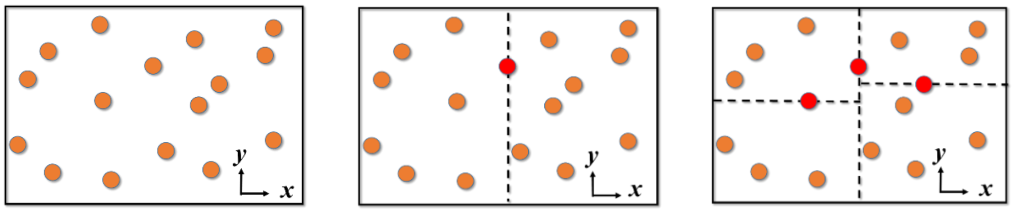
The procedure for dividing the search area is as follows.

(1) Draw a vertical border dividing an area into two as follows. The nodes indicating the cities of the TSP are rearranged in ascending order of x coordinate, and a vertical boundary line dividing the area into two is drawn based on the node located at the center, that is, where the numbers of nodes on both sides of the boundary are almost equal (Figure 3(b)).

(2) Draw a horizontal border dividing an area into two as follows. The nodes are sorted in ascending order of y coordinate within each divided region, and a horizontal boundary line dividing the area into two is drawn based on the node located at the center (Figure 3(c)).

(3) Repeat (1) and (2) until the number of nodes in each area is n or less. **Figure 3 Dividing search area**

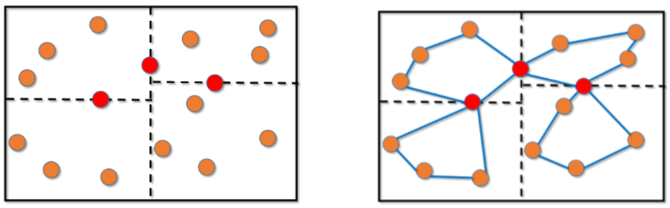
(a) set of nodes (b) dividing according to x (c) dividing according to y



Searching for partial solution by the slime mold algorithm for each divided area

The partial solutions of the TSP are searched for using SMA in each divided area. In the example shown in Figure 4(a), there are four areas to be searched. In each area, a new mesh is created by the Delaunay triangulation method, and a partial solution is searched for by the slime mold to generate a path as shown in Figure 4(b).

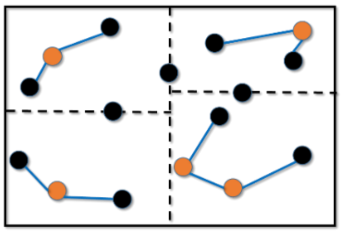
Figure 4 Searching for a partial solution by the SMA for each divided area (a) divided search areas (b) partial solutions



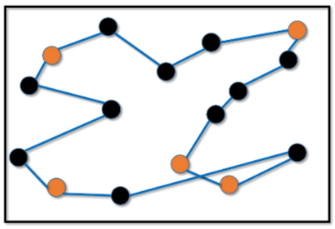
Integrating partial solutions to obtain an approximate solution

The partial solutions of the partial areas are integrated to generate one approximate solution of the whole area. First, all the edges connected to the nodes which are located on boundary lines are cut (Figure 5(a)). Second, we seek a set V of nodes (black dots) in which the number of connected edges is one or less (Figure 5(a)). Third, a traveling route is created by connecting all the nodes of set V avoiding nodes with shorter distances to create a circuit (Figure 5(b)).

Figure 5 Integrating partial solutions to obtain an approximate solution (a) cutting edges connected to the boundary nodes



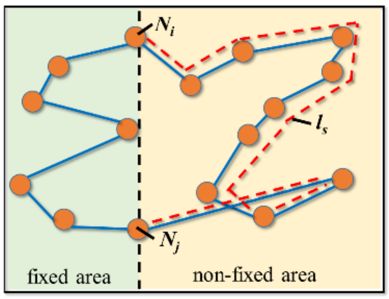
(b) connecting a new traveling route



Improvement of approximate solution by subdivision and search in partial areas It is necessary to search again for solutions in order to improve the accuracy of the created approximate solution. However, if all nodes were considered in the search, the computation time necessary to generate a solution would be quite large. In this study, we just subdivide the search area and search for a part of the approximate solution generated already in order to reduce the calculation time.

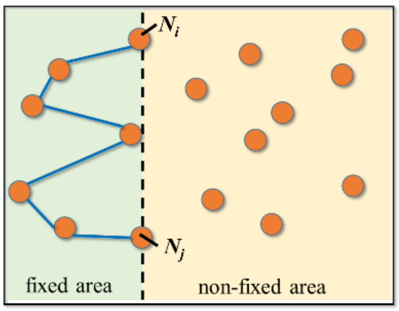
First, the approximate solution is divided into a fixed area and a non-fixed area. The path in the fixed area is used as part of the next solution. In the non-fixed area, a new solution route is searched for. The path length l_s of in the non-fixed area is obtained by using a probability density function [Tsutsui2007]. After l_s is determined, a route is set in the non-fixed area from city N_i randomly selected to another city N_j that is distance l_s from city N_i. This area division is shown in Figure 6.

Figure 6 Area division



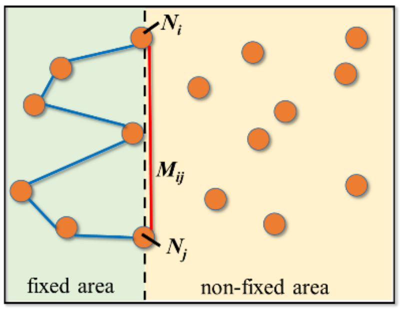
Second, as shown in Figure 7, all the edges in the non-fixed area are cut off in order to obtain a new partial solution.

Figure 7 Cut off all edges in non-fixed region

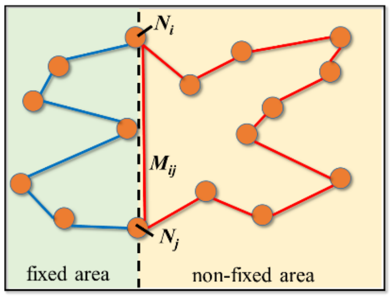


Third, a partial solution is obtained by using the SMA in the non-fixed area. After connecting the edge M_{ij} between city N_i and city N_j (Figure 8(a)), the partial solution including M_{ij} is searched for by using the SMA (Figure 8(b)).

Figure 8 Obtaining a partial solution by using the SMA in the non-fixed area (a) nodes after cutting off non-fixed area

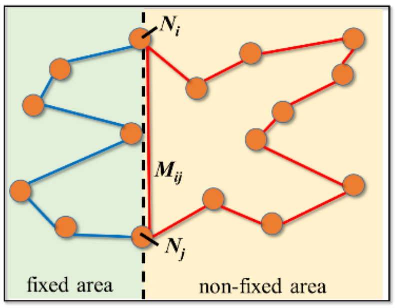


(b) connecting a new traveling route in non-fixed area



Finally, the paths between the fixed and non-fixed areas are combined. After cutting off the edge M_{ij}, a path newly generated in the non-fixed area is connected to the fixed path to obtain a new approximate solution. If this approximate solution is better than the previously obtained solution, then this approximate solution is taken as a new solution.

Figure 9 A new approximate solution is obtained by integrating two partial solutions



Experimental results

For numerical experiments, the nonlinear feedback parameter of the slime mold algorithm is set as $\delta = 1.8$, the flow rate of node for the food source is set as $Q_0=1.0$, and the distance weight is set as $w=3.0$ in the edge flow calculation [Tero2010]. All experiments are performed 10 times, and the average value is taken as the result.

In order to verify the efficiency and effectiveness of the pSMA for TSP, as proposed in Section 3, and to investigate performance of the pSMA, we compare it to SMA, where SMA means the standard slime mold algorithm. For this comparison, the TSPLIB benchmark problems [TSPLIB2018] are used.

We chose eil51, lin105, ch150, d198, and lin318 from TSPLIB as the target problems. The area division number is set as 2 in the pSMA. The approximate solution is output every 2 seconds, and the execution time for each problem is set to 10 minutes. The summary of the obtained results is shown in Table 1. In the table, Best is the average of the path lengths of the best approximate solutions after 10 minutes, and the error is the relative error between the optimal solution and Best.

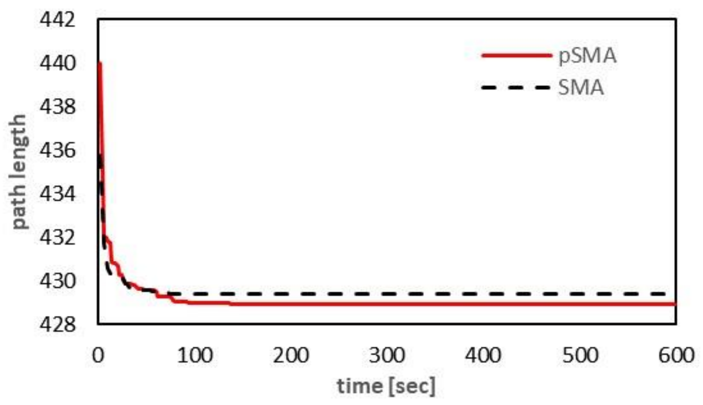
Table 1 Results of SMA and pSMA for the benchmark problems

problem	optimum	SMA		pSMA	
		Best	error [%]	Best	error [%]
eil51	426.00	429.41	0.80	428.92	0.68
lin105	14379.00	14441.09	0.43	14435.06	0.39
ch150	6528.00	6575.63	0.73	6571.88	0.67
d198	15780.00	18541.09	17.50	17235.77	9.23
lin318	42029.00	49029.97	16.66	46999.93	11.83

We can see that the path lengths and relative errors of the pSMA are better than those of the SMA for the benchmark problems. In particular, the difference in the total distance between the pSMA and SMA are large for d198 and lin318, allowing us to conclude that the pSMA is more effective than the SMA for optimizing the TSP when the number of nodes is approximately 200 or more.

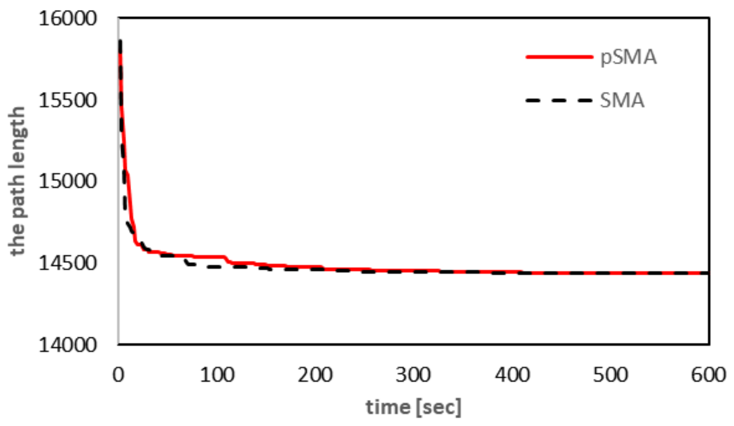
The changes over time in the path lengths of the approximate solutions to eil51 obtained by the pSMA and SMA are shown in Figure 10. The horizontal axis indicates search time and the vertical axis indicates the path length of the TSP. The solution converges at 74 seconds by the pSMA, but requires 100 seconds by the SMA. Finally, a slightly shorter path length is obtained by the pSMA than by the SMA.

Figure 10 Path lengths of the approximate solutions to eil51 obtained by the pSMA and SMA



The path length changes of the approximate solutions to lin105 and ch150 obtained by the pSMA and SMA are shown in Figure 11. When the number of nodes is about 100 to 150, the accuracies of the solutions of SMA and pSMA are almost the same, making the merit of using pSMA unclear. For small-scale problems, the reduction of processing time due to the proposed method was not so much.

Figure 11 Path lengths of the approximate solutions obtained by the pSMA and SMA for (a) 105 and (b) 150 nodes (a) lin105



(b) ch150

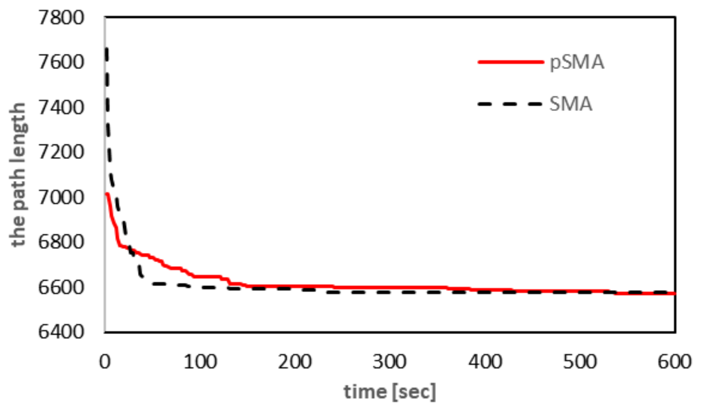


Figure 12 shows the evolution of path length of the approximate solutions to d198 obtained by the pSMA and SMA. It can be seen that the pSMA obtains shorter paths than the SMA. As the number of nodes increases, the SMA takes much more time to search for a single solution, whereas the amount of calculation is suppressed by the pSMA because of the divided search area becoming smaller. In addition, the approximate solutions of the pSMA are superior to those of the SMA at the initial stage. From this result, we also see that the solution generated by dividing and joining partial solutions obtained a better approximate solution even at an early stage.

Figure 12 Path lengths of the approximate solutions to d198 obtained by the pSMA and SMA

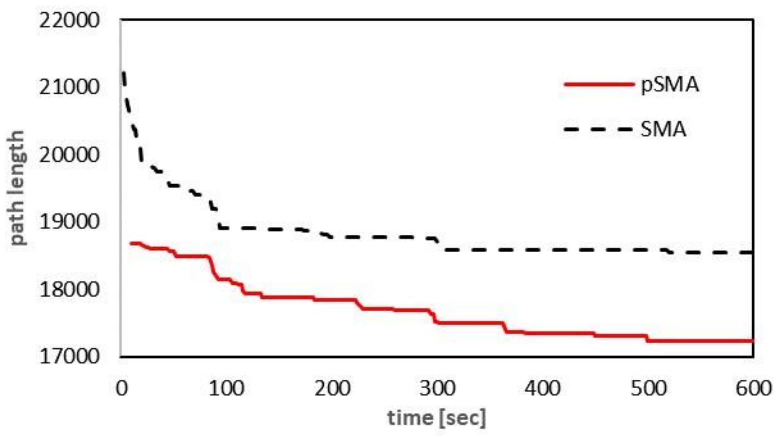
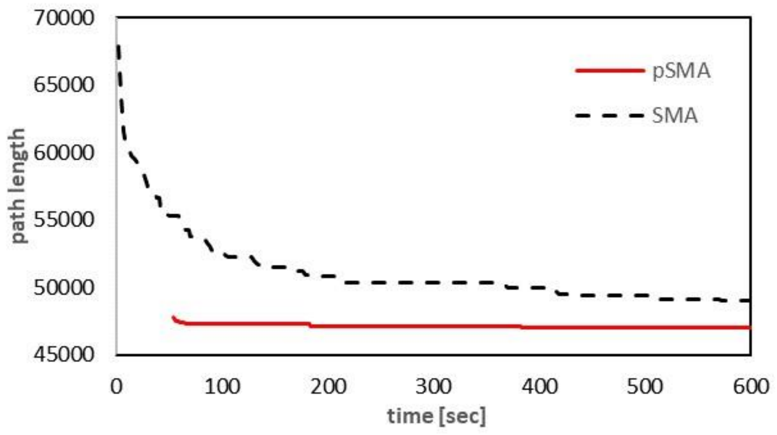


Figure 13 shows the evolution of path length of the approximate solutions to lin318 obtained by the pSMA and SMA. As mentioned above, it takes more time to generate the first approximate solution as the number of nodes increases; in this case, the first solution was generated in 54 seconds by pSMA. In our results, the approximate solution by the pSMA at 100 seconds is better than that by the SMA at 600 seconds. However, since the final approximate solution of the pSMA still has a large error relative to the optimal solution, it is considered that the algorithm became trapped at a local solution.

Figure 13 Path lengths of the approximate solutions to lin318 obtained by the pSMA and SMA



Conclusion

In this study, a new method referred to as the slime mold algorithm including partial solutions (pSMA) was proposed for solving the TSP. In order to reduce the amount of calculation, the proposed method divides the TSP area into several partial areas and searches for partial solutions, which it combines into an approximate solution.

We performed experiments on a computer for the TSPLIB benchmark problems to compare the proposed method with the conventional approach, examining how the approximate solutions obtained by the different algorithms change over time. We found that the solution accuracy of the proposed method is significantly higher than that of the conventional method for the TSP with 200 or more nodes. On the other hand, when the number of nodes was small, the conventional method sometimes showed better convergence. In the case of the proposed method, it is considered that excellent convergence can be achieved for a larger number of nodes by improving the method of determining the proportion of non-fixed areas and the method of integrating solutions.

Our future work will include adding additional features to our model and improving the method of combining the partial solutions into an approximate solution.

References

Cook, William J., Cunningham, William H., Pulleyblank, William R., Schrijver, Alexander (1997). Combinatorial Optimization. Wiley. ISBN 0-471-55894-X

Cook, William. (2016). Optimal TSP Tours. University of Waterloo

Roberto Cordone, Guglielmo Lulli. (2016). Multimode extension of combinatorial optimization problem, Electronic Notes in Discrete Mathematics 55, 17-20

M. Yannakais. (1994). On the approximation of maximum satisfiability, J. Algorithms, 17 (1994), 475–502

Michael Brusco, Hannah J. Stolze, Michaela Hoffman, and Douglas Steinley. (2017). A simulated annealing heuristic for maximum correlation core/periphery partitioning of binary networks, PLoS One, 12(5): e0170448, doi: 10.1371/journal.pone.0170448

M.C. Xie, H. Ogura, T. Odaka, and J. Nishino. (1996). Application of genetic algorithm to inter-correlated nonlinear Knapsack problem, 1996 International Symposium on Nonlinear Theory and its Application, 145-148

M. C. Xie, M. Murata, Y. Murakami, K. Yamagiwa. (2015). A Search Method of Particle Swarm Optimization Including Pheromone Information for Function Optimization, Proceeding of the 15th International Conference on Evolutionary Computing,11-16

Glover, F. and M. Laguna. (1993). "Tabu Search", Modern Heuristic Techniques for Combinatorial Problems, C. Reeves (ed.), Blackwell Scientific Publishing, Oxford, (1993)70-150

H. Tamura, A. Hirahara, I. Hatono and M. Umano. (1994). An approximate solution method for combinatorial optimization, The Society of Instrument and Control Engineers, Vol.30. No.3, 329-336

A. Valejo, M. Cristina, G. Filho, A. Lopes. (2018). Multilevel approach for combinatorial optimization in bipartite network, Knowledge-Based Systems 151, 45-61

K. Li, C.E. Torres, K. Thomas, L. F. Rossi, and C. Shen. (2011). Slime mode inspired routing protocols for wireless sensor networks, Swarm Intell (2011)5:183-223

L. Zhu, M. Aono, S. Kim, M. Hara. (2013). Amoeba-based computing for traveling salesman problem: long-term correlations between spatially separated individual cells of Physarum polycephalum, Biosystems, Apr. 112(1):1-10

TSPLIB. (2018). http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/

G. Kovacs, Z. Tuza, B. Vizvari, and H. K. Jabbari. (2018). A note on the polytope of bipartite TSP, Discrete Applied Mathematics 235, 92-100

A. Tero, S. Takagi, T. Saigusa, k. Ito et al. (2010). Rules for biologically inspires adaptive network design, Science, Vol.327, No.5964, 439-442

A. Takamura, E. TAkaba, G. Takizawa. (2009). Environment-dependent morphology in plasmodium of true slime mold Physarum polycephalum and a network growth model, Journal of Theoretical Biology 256, 29-44

J. R. Shewchuk. (2014). Reprint of : Delaunay refinement algorithms for triangular mesh generation, Computational Geometry 47, 741-778

S. Tsutsui. (2007). Ant colony optimization with cunning ants, The Japanese Society for Artificial Intelligence, Vol. 22 (1), 29-36

Acknowledgments. This research was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number JP16K01303.