Chapter 33
Tabu Search Algorithm Based on Strategic Oscillation for Nonlinear Minimum Spanning Tree Problems

Hideki Katagiri, Masatoshi Sakawa, Kosuke Kato, Ichiro Nishizaki, Takeshi Uno, and Tomohiro Hayashida

33.1 Introduction

The minimum spanning tree (MST) problem is to find the least cost spanning tree in an edge-weighted graph. In the real world, MST problems are usually seen in network optimization. For instance, when designing a layout for telecommunication systems, if a decision maker prefers to minimize the total cost for connection between cities or sections, it is formulated as an MST problem. In other examples, the objective is to minimize the total time for construction or to maximize the network reliability.

In classical MST problems, weights attached to edges are constant, and all the weights are independent of each other. In other words, the objective function is linear. Polynomial–time algorithms for solving a usual MST problem were first constructed by Kruskal [1] and Prim [2]. Gabow et al. [3] and Chazelle [4] developed more efficient algorithms.

However, there is a case where the objective function is nonlinear in real-world problems. For instance, when each edge weight is represented by a random variable or a fuzzy set, MST problems under randomness are often equivalently transformed into a deterministic nonlinear MST problem. Katagiri and others [5–7] considered some specific types of minimum spanning tree problems where each edge constant is a fuzzy random variable. They also constructed polynomial–time algorithms for solving nonlinear MST problems that are deterministic equivalent problems for the original ones.

Since a nonlinear MST problem is generally one of the NP-hard combinatorial optimization problems, it is important to construct approximate solution algorithms for solving large-scale nonlinear MST problems. As for previous studies on

Hideki Katagiri, Masatoshi Sakawa, Kosuke Kato, Ichiro Nishizaki, Takeshi Uno, and Tomohiro Hayashida
Graduate School of Engineering, Hiroshima University, Kagami-yama 1-4-1, Higashi-hiroshima, Hiroshima, 739-8527 Japan

nonlinear MST problems, Zhou and Gen [8] considered quadratic MST problems and proposed a solution algorithm through genetic algorithms (GAs).

For combinatorial optimization problems, there are some metaheuristics such as evolutionary computation (EC), simulated annealing (SA), ant colony optimization (ACO), and tabu search (TS). Blum and Blesa [9] investigated several metaheuristic approaches for edge-weighted k-cardinality tree problems and compared the performances of EC, SA, ACO, and TS. They demonstrated that TS [10] has advantages for high cardinality. Since MST problems corresponds to highest cardinality case in k-cardinality tree problems, it is expected that the performance of TS algorithm is good for nonlinear MST problems. Therefore, we motivate ourselves to develop a TS algorithm for solving nonlinear MST problems. In particular, we construct a TS algorithm based on strategic oscillation, which has good performance for multidimensional 0–1 knapsack problems [11].

This paper is organized as follows: Section 2 formulates a nonlinear MST problem. In Section 3, we propose a solution algorithm using TS. Section 4 provides numerical experiments and shows the advantage of our algorithm over the algorithm using a GA.

33.2 Problem Formulation

In this paper, we consider a connected undirected graph $\mathscr{G} = (V,E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a finite set of vertices and $E = \{e_1, e_2, \ldots, e_m\}$ is a finite set of edges. Let $x = (x_1, x_2, \ldots, x_m)^t$ be an m-dimensional column vector. We identify a spanning tree T with an x if

$$x_i = \begin{cases} 1 \text{ if edge } e_i \in T \text{ is selected,} \\ 0 \text{ otherwise.} \end{cases}$$

Then, an MST problem with a nonlinear objective function is formulated as follows:

$$\begin{aligned} &\text{Minimize } f(x) \text{ subject to} \\ &\qquad a_j x \le b_j,\ j = 1, \ldots, l,\ x \in X, \end{aligned} \tag{33.1}$$

where f is a real-valued nonlinear function and a_j is a m-dimensional row vector. X stands for the collection of x which corresponds to a spanning tree in the given graph $\mathscr{G}$.

When there is no constraint and f is linear in the above problem, problem (33.1) becomes a usual minimum spanning tree problem. In this case, the problem is solved by polynomial–time algorithms such as the Prim method on the Kruskal method.

In general, however, problem (33.1) is an NP-hard problem, and the existing exact solution algorithms such as the branch and bound method cannot solve a

large-scale nonlinear MST problem in a practically feasible computational time. Therefore, we develop a TS algorithm for obtaining a good approximate optimal solution of the large-scale nonlinear MST problems.

33.3 Summary of Tabu Search

Local search generally improves the current solution because it moves from the current solution x^c to a solution $x' \in N(x^c)$, which is better than the current solution, where $N(\cdot)$ is a given neighborhood structure. For simplicity, suppose that x^c is a local minimum solution and that the next solution x' is selected as the best solution among $N(x^c)$. If the local search is applied for x', then the next solution moved from x' is back to x^c because x^c is the best solution among a neighborhood $N(x')$. In this way, cycling among solutions often occurs around local minima. In order to avoid such cycling, TS algorithms use a short-term memory. The short-term memory is implemented as a set of tabu lists that store solution attributes. Attributes usually refer to components of solutions, moves, or differences between two solutions. Tabu lists prevent the algorithm from returning to recently visited solutions.

Aspiration criteria permit a part of moves in the tabu list to cancel any tabu status. The typical aspiration criterion is to accept a tabu move if it leads to a new solution better than the current best solution. The outline of TS is as follows:

Step 1. Generate an initial solution x and initialize a tabu list TL.
Step 2. Find the best solution $x' \in N(x)$ such that $x' \notin TL$, and set $x := x'$.
Step 3. Stop if a termination condition is satisfied. If not, then update TL and return to Step 2.

In Step 2, a tabu list memorizes solution attributes. A tabu tenure, i.e., the length of the tabu list determines the behavior of the algorithm. A larger tabu tenure forces the search process to explore larger regions, because it forbids revisiting a higher number of solutions.

In Step 3, we check whether the algorithm satisfies a termination condition. The termination condition is usually related to the iteration number of the algorithm and/or the iteration number of not updating the current best solution.

33.4 Tabu Search Algorithm Based on Strategic Oscillation for Nonlinear MST Problems

Hanafi and Freville [11] considered a TS algorithm for the 0–1 multidimensional knapsack problems. Their algorithm is based on strategic oscillation. Strategic oscillation is useful to efficiently explore the region where there may be good

solutions, called *promising zone*. Since the promising zone for 0-1 multidimensional knapsack problems is the boundary between feasible region and infeasible region, strategic oscillation explores the region around the boundary, crossing over it.

We extend the Hanafi's algorithm to deal with nonlinear MST problems. The essential features of our TS algorithm for solving a nonlinear minimum spanning tree problem are characterized by several procedures, i.e., generating an initial solution, local search, strategic oscillation, and diversification procedure. The outline of our TS algorithm is as follows:

Step 0. (Initial Solution). Generate an initial solution x^0. If x^0 is feasible, then set $x^c := x^0$ and $x^b := x^0$. Otherwise, construct a feasible solution x_f^0 from x^0 using MCVQ, and set $x^c := x_f^0$ and $x^b := x_f^0$.

Step 1. (Local Search). If the termination condition is satisfied, then the algorithm is terminated. Otherwise, improve the current solution x_c by local search, and set $x^b := x^c$.

Step 2. (Strategic Oscillation). Explore around the boundary between the feasible and infeasible regions, alternating MOFV with MCVQ.

Step 3. (Diversification). Remove some of edges that have high residence frequency from T^c. Slap a long tabu tenure to the removed edges. Continue to add the edge that has low residence frequency so as not to make a cycle until a spanning tree is formed, using MOFV criterion. If the current solution is infeasible, then move it to a feasible solution using MCVQ criterion. Return to Step 1.

33.4.1 Initial Solution

Let $SCC(i)$ denote a set of connected components that consists of exactly i edges. To construct a spanning tree, first, an edge $e \in E$ is chosen uniformly at random. This procedure constructs a subtree $SCC(1)$ that consists of only one edge. Then, a connected component $SCC(k+1)$ is constructed by adding an edge to the $SSC(k)$, following the *edge addition rule*:

Edge addition rule. $SCC(k+1)$ is constructed by adding an edge

$$e :\in \arg \min_{e' \in E_{NC}(SCC(k))} \{f(SCC(k)+e') - f(SCC(k))\}$$

to the current $SCC(k)$ under construction, where

$$E_{NC}(SCC(k)) := \{e \in E | SCC(k) \text{ and } e \text{ has no cycle}\}.$$

33.4.2 Neighborhood Structure and Local Search

Let T be a set of edges which forms a spanning tree, and let $\mathscr{T}$ be a class of all possible spanning trees in a given graph. The neighborhood $N(T)$ consists of all spanning trees that can be generated by removing an edge $e \in T$ and by adding an edge from the set $E_{NH}(T-e) \setminus \{e\}$, where $E_{NH}(T-e)$ is defined as follows:

$$E_{NH}(T-e) := \{e' \in E | T-e+e' \in \mathscr{T}\}.$$

In order to improve a current solution $x^c \in X$ corresponding to the current spanning tree T^c, it scans the neighborhood $N(x^c)$ and chooses a spanning tree x' such that

$$x' \in \arg \min_{x \in N(x^c) \cap X} f(x).$$

This procedure is local search or the exploration of local area of the current solution.

33.4.3 Tabu List and Aspiration Criterion

In general, local search updates a current best solution. However, such updating is stopped, when cycling occurs around a local optimum solution. Our TS algorithm uses only one tabu list denoted by *TabuList*, which stores indices of the edges that were recently added or removed. As described before, every move consists of two steps; the first step is to remove one edge $e \in T^c$ from the current spanning tree T^c, and the second step is to add an edge in $E_{NH}(T^c-e) \setminus \{e\}$ to T^c-e. The status of the forbidden moves are explained as follows: If an edge e_j is in *TabuList* and $x_j = 0$, then our algorithm forbids the addition of the edge e_j, namely, $x_j \not\leftarrow 1$. In addition, if an edge e_i is in *TabuList* and $x_i = 1$, then our algorithm forbids the deletion of the edge e_i, namely, $x_i \not\leftarrow 0$.

An aspiration criterion is activated to overcome the tabu status of a move whenever the solution then produced is better than the best historical solution achieved. This criterion will be effective only after a local optimum is reached.

33.4.4 Strategic Oscillation

The characteristic of the strategic oscillation is that the several move evaluation criteria are used for selecting moves. Our algorithm involves two move evaluation criteria. One is minimizing objective function value (MOFV), and the other is minimizing constraint violation quantity (MCVQ).

While MOFV is used, our algorithm continues to select moves for a specified depth beyond the boundary, without considering the constraints. To be more specific,

the algorithm alternates the edge addition rule described before and the following *edge deletion rule*:

Edge deletion rule. $SCC(k-1)$ is constructed by removing an edge

$$e \in \arg\max_{e' \in CY(k)} \{f(SCC(k) - e') - f(SCC(k))\}$$

from a cycle in the current $SCC(k)$ under construction, where $CY(k)$ denotes the set of edges in $SCC(k)$ which forms cycles.

The evaluation criterion is switched from MOFV to MCVQ at some turning point. While MCVQ is used to evaluate possible moves, the algorithm continues to select x' such that

$$x' \in \arg\min_{x \in N(x^c)} \delta(x),$$

where δ is the degree of violation of the constraints defined by

$$\delta(x) := \sum_{j=1}^{l} \max\{0, a_j x - b_j\}.$$

33.4.5 Diversification

Frequency-based memory is one of the long-term memories and consists of gathering pertinent information about the search process so far. In our algorithm, we use residence frequency memory, which keeps track of the number of iterations where edges have been selected as a part of the solution. The diversification procedure begins at the situation that some spanning tree is formed.

33.5 Numerical Experiment

In this section we apply our TS algorithm to solve examples of problem (33.1). The experiments are executed for complete undirected graphs where the number m of edges are 45, 105, 190, and 435. We use C as the programming language and compile all software with Microsoft Visual C++ 6.0. All the examples were tested on a PC with Celeron 2.4 GHz CPU under Microsoft Windows 2000.

The compared GA is a modified version of the algorithm proposed by Zhou and Gen [8], in which the solution structure and genetic operations are based on the EC algorithm proposed by Blum and Blesa [9]. We have run each experiment 30 times. In each table, Best, Mean and Worst denote the best, mean and worst objective function values, respectively. Time in this table represents the average CPU running time in seconds.

Table 33.1 Comparison result for Example 1

		TS	GA
$m = 45$	Best	2.65×10^{-4}	2.65×10^{-4}
	Mean	2.65×10^{-4}	2.65×10^{-4}
	Worst	2.65×10^{-4}	2.65×10^{-4}
	Time(s)	**0.094**	3.964
$m = 105$	Best	5.98×10^{-7}	5.98×10^{-7}
	Mean	$\mathbf{5.98 \times 10^{-7}}$	2.03×10^{-6}
	Worst	$\mathbf{5.98 \times 10^{-7}}$	4.98×10^{-6}
	Time(s)	**0.162**	19.003
$m = 190$	Best	$\mathbf{5.47 \times 10^{-12}}$	3.05×10^{-8}
	Mean	$\mathbf{1.15 \times 10^{-10}}$	2.80×10^{-9}
	Worst	$\mathbf{3.50 \times 10^{-10}}$	9.25×10^{-9}
	Time(s)	**0.631**	101.479
$m = 435$	Best	$\mathbf{2.98 \times 10^{-17}}$	1.89×10^{-11}
	Mean	$\mathbf{7.01 \times 10^{-12}}$	5.11×10^{-10}
	Worst	$\mathbf{4.44 \times 10^{-11}}$	1.96×10^{-9}
	Time(s)	**4.11**	716.58

Example 1. The first example is a nonlinear fractional MST problem, which is derived from a deterministic equivalent problem of an MST problem under fuzzy stochastic environments.

$$\text{Minimize} \quad \frac{x^t V x}{(\beta x + \gamma)^2} \quad \text{Subject to} \quad ax \le b,\ x \in X,$$

where V is an $m \times m$ positive definite matrix. The coefficient vector β is defined as $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$, where $\beta_j, j = 1, \ldots, m$, are constant and γ is also constant.

Table 33.1 shows the comparison result for Example 1. In Table 33.1, better values are indicated by boldface. Our TS algorithm is better than GA from the viewpoints of both accuracy and computational time.

Example 2. The second example is derived from the reliability optimization problem.

$$\text{Maximize} \quad \prod_{i=1}^{m} (r_i + c_i x)^{x_i} \quad \text{subject to} \quad a_j x \le b_j,\ j = 1, \ldots, l,\ x \in X$$

where r_i and $c_i = (c_{i1}, c_{i2}, \ldots, c_{im})$ are a constant and a constant value vector, respectively.

Table 33.2 shows the comparison result for Example 2. Our TS algorithm is better than GA from the viewpoints of both accuracy and computational time.

Table 33.2 Comparison result for Example 2

		TS	GA
$m = 45$	Best	$\mathbf{6.36 \times 10^{-1}}$	6.33×10^{-1}
	Mean	6.33×10^{-1}	6.33×10^{-1}
	Worst	6.33×10^{-1}	6.33×10^{-1}
	Time(s)	**0.152**	2.802
$m = 105$	Best	$\mathbf{6.21 \times 10^{-1}}$	6.15×10^{-1}
	Mean	$\mathbf{6.17 \times 10^{-1}}$	5.95×10^{-1}
	Worst	$\mathbf{6.15 \times 10^{-1}}$	5.81×10^{-1}
	Time(s)	**0.364**	8.3438
$m = 190$	Best	$\mathbf{6.09 \times 10^{-1}}$	5.72×10^{-1}
	Mean	$\mathbf{6.07 \times 10^{-1}}$	5.34×10^{-1}
	Worst	$\mathbf{6.07 \times 10^{-1}}$	4.83×10^{-1}
	Time(s)	**0.948**	24.592
$m = 435$	Best	$\mathbf{5.61 \times 10^{-1}}$	4.55×10^{-1}
	Mean	$\mathbf{5.59 \times 10^{-1}}$	3.95×10^{-1}
	Worst	$\mathbf{5.59 \times 10^{-1}}$	3.20×10^{-1}
	Time(s)	**6.31**	96.93

Example 3. The third example is an original problem in which the objective function includes a trigonometric function.

$$\text{Minimize} \sum_{i=1}^{m} \left[y_i^2 - 10\cos(2\pi y_i) \right] \quad \text{subject to} \quad y_i = 5.14 \times \frac{\sum_{j=1}^{i} (-1)^j x_j}{n_2 - 1},$$
$$i = 1, \ldots, m \quad a_j x \leq b_j,\ j = 1, \ldots, l\ x \in X$$

Table 33.3 Comparison result for Example 3

		TS	GA
$m = 45$	Best	97.01	97.01
	Mean	**97.01**	98.95
	Worst	**97.01**	116.41
	Time(s)	**0.174**	2.802
$m = 105$	Best	117.46	117.46
	Mean	**117.46**	171.16
	Worst	**117.46**	251.71
	Time(s)	**2.787**	11.8323
$m = 190$	Best	**112.93**	225.86
	Mean	**129.07**	306.04
	Worst	393.40	**361.38**
	Time(s)	**12.377**	32.688
$m = 435$	Best	**83.68**	412.80
	Mean	**97.43**	528.14
	Worst	**145.04**	601.89
	Time(s)	**50.62**	110.53

Table 33.3 shows the comparison result for Example 3. Our TS algorithm is better than GA except for the worst objective function value for $m = 190$.

As a whole, Tables 1 to 3 show the advantage of TS over the GA in terms of both computational time and accuracy.

33.6 Conclusion

In this paper, we have considered a TS algorithm for solving nonlinear minimum spanning tree problems. The proposed algorithm has been developed based on strategic oscillation and diversification by residence frequency. The results of numerical experiments show that our algorithm has advantages of high speed and high accuracy.

In the future, we will investigate effects on the performance of TS algorithm if the parameters involved in TS algorithm are changed. Furthermore, we will examine the performance of our TS algorithm for various types of graphs, such as grid graphs, regular graphs, and scale-free network.

References

1. J.B. Kruskal (1956) On the shortest spanning subtree and the traveling salesman problem, Proceedings of the American Mathematical Society, 7: 48–50.
2. R.C. Prim (1957) Shortest connection networks and some generalisations. Bell System Technical Journal, 36: 1389–1401.
3. H.N. Gabow, Z. Galil, T. Spencer, and R.E. Tarjan (1986) Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica, 6: 109–122.
4. B. Chazelle (2000) A minimum spanning tree algorithm with inverse-Ackermann type complexity. Journal of the ACM, 47: 1028–1047.
5. H. Katagiri and H. Ishii (2000) Chance constrained bottleneck spanning tree problem with fuzzy random edge costs, Journal of the Operations Research Society of Japan, 43: 128–137.
6. H. Katagiri, M. Sakawa, and H. Ishii (2004) Fuzzy random bottleneck spanning tree problems. European Journal of Operational Research, 152: 88–95.
7. H. Katagiri, E.B. Mermri, M. Sakawa, K. Kato, and I. Nishizaki (2005) A possibilistic and stochastic programming approach to fuzzy random MST problems. IEICE Transaction on Information and Systems, E88-D: 1912–1919.
8. G. Zhou and M. Gen (1998) An efficient genetic algorithm approach to the quadratic minimum spanning tree problem. Computers & Operations Research, 25: 229–237.
9. C. Blum and M.J. Blesa (2005) New metaheuristic approaches for the edge-weighted k-cardinality tree problem. Computers & Operations Research, 32: 1355–1377.
10. F. Glover and M. Laguna (1997) Tabu search. Kluwer Academic Publishers, Norwell, MA.
11. S. Hanafi and A. Freville (1998) An efficient tabu search approach for the 0–1 multidimensional knapsack problem. European Journal of Operational Research, 106: 659–675.