Scientific Research Publishing

American Journal of Operations Research, 2017, 7, 249-262
http://www.scirp.org/journal/ajor
ISSN Online: 2160-8849
ISSN Print: 2160-8830

Covering Salesman Problem with Nodes and Segments

Takafumi Matsuura[1], Takayuki Kimura[2]

[1]Department of Computer & Information Engineering, Nippon Institute of Technology, Saitama, Japan
[2]Department of Electrical and Electronic Engineering, Nippon Institute of Technology, Saitama, Japan
Email: matsuura@nit.ac.jp, tkimura@nit.ac.jp

How to cite this paper: Matsuura, T. and Kimura, T. (2017) Covering Salesman Problem with Nodes and Segments. *American Journal of Operations Research*, **7**, 249-262.
https://doi.org/10.4236/ajor.2017.74017

Received: April 28, 2017
Accepted: July 17, 2017
Published: July 20, 2017

Abstract

In the Covering Salesman Problem (CSP), a distribution of nodes is provided, and the objective is to identify the shortest-length tour of a subset of all given nodes such that each node is not on the tour which is within a radius r of any node on the tour. In this paper, we define a new covering problem called the CSP with Nodes and Segments (CSPNS). The main difference between the CSP and the CSPNS is that in the CSPNS, not only the nodes on the tour but also the segments on the tour can cover the nodes not on the tour. We formulated the CSPNS via integer programming and found an optimal solution by using a general-purpose mixed-integer program solver. Benchmark instances of the CSPNS were generated by DIMACS, which is one of the benchmark problems of the Traveling Salesman Problem. Optimal solutions could not be obtained in a reasonable time frame for a large size of instances. Thus, in this study, we developed a simple heuristic method to find good near-optimal solutions to the CSPNS. The proposed heuristic method quickly finds good solutions.

Keywords

Covering Salesman Problem, Covering Salesman Problem with Nodes and Segments, Combinatorial Optimization Problem, Local Search Method

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most famous combinatorial optimization problems [1]. In the TSP, a set of nodes $V = \{1, 2, \cdots, n\}$ is provided. Let d_{ij} be the distance from node i to node j. A salesman starts from a node, visits each node exactly once, and returns to the starting node. The objective of the TSP is to find the shortest-length tour. The TSP has several practical applications, such as drilling, computer wiring, routing, very-large-scale integra-

tion design, and job sequencing. To reduce the cost of solving such problems, the development of an algorithm for finding an optimal solution or a near-optimal solution to the TSP has been actively pursued [2]-[7]. **Figure 1(a)** shows a graphical example of the TSP.

In recent years, several mobile-service cars, for example, mobile libraries, mobile shops, and waste-recovery vehicles, have traveled in our town. In the service systems, the vehicle does not visit all the users to provide the service. A service provider selects stopping places, and the vehicle visits these places to provide the service; the users visit the nearest stopping place to receive the service. If the service provider selects many stopping places, the distances between a user and a stopping place are short. However, the tour of the vehicle is long. Therefore, in the service system, it is important to determine optimal stopping places and an optimal tour. To solve this problem, the Covering Salesman Problem (CSP) has been formulated [8]. In the CSP, a set of nodes $V = \{1, 2, \cdots, n\}$ is provided, along with the distance d_{ij} between nodes i and j for all nodes. In addition, the covering distance r is given. The salesman visits a node. The node visited by the salesman can cover other nodes within a radius r. The objective of the CSP is to identify the shortest tour of a subset of all given nodes, such that each node that is not on the tour is within a radius r of a node on the tour [8]. **Figure 1(b)** shows a graphical example of the CSP.

In this study, we define a new covering problem called the CSP with Nodes and Segments (CSPNS). To illustrate the CSPNS, we consider advertising, which is one of inevitable activities in modern business. It requires a medium for sending promotional messages to targeted people, for example, an advertising truck (AD-truck) that drives on town streets while displaying and broadcasting information about a new product or a local event. The aim of the AD-truck is to promote new products or local events to people on the street. Even though the truck is moving, announcements from the truck can be heard by people on the street. Therefore, the AD-truck does not have to stop for people to see and hear the advertising. This is the most important difference between the CSP and the CSPNS. In the CSP, the nodes on the tour can only cover the nodes not on the tour. However, in the CSPNS, not only the nodes but also the segments on the

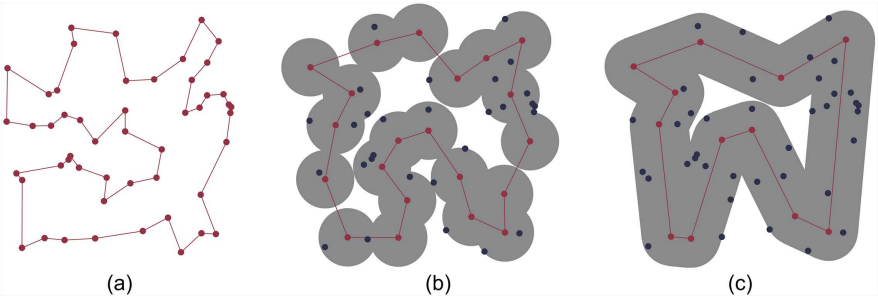

Figure 1. Graphical interpretation of (a) Traveling Salesman Problem; (b) Covering Salesman Problem; and (c) Covering Salesman Problem with Nodes and Segments. In these figures, red lines indicate the optimal tours. Red circles represent the visited nodes. The covering area is shown in gray.

tour can cover the nodes not on the tour. Figure 1(c) shows a graphical example of the CSPNS.

The rest of this paper is organized as follows. In Section 2, we formally define the Covering Salesman Problem with Nodes and Segments. We also present the simulation results by using a general purpose mixed integer program solver. Section 3 describes a local search method for solving the CSPNS. Section 4 discusses computational results of the proposed method. Section 5 provides concluding remarks and discusses possible extensions of the proposed method.

2. CSPNS

2.1. Problem Definition

In the CSPNS, a set of nodes $V = \{1, 2, \cdots, n\}$, the distance d_{ij} between nodes i and j, and the perpendicular distance p_{ijk} between node i and edge (j, k) are given, along with the covering distance of the node, $r_n \geq 0$, and that of the edge, $r_e \geq 0$. From the data provided, we have two constant values:

$$a_{ij} = \begin{cases} 1: \text{if node } i \text{ is covered by node } j \left(d_{ij} \leq r_n\right), \\ 0: \text{otherwise} \end{cases} \tag{1}$$

and

$$b_{ijk} = \begin{cases} 1: \text{if node } i \text{ is covered by edge } \left(j,k\right) \left(p_{ijk} \leq r_e\right), \\ 0: \text{otherwise.} \end{cases} \tag{2}$$

In the CSPNS, node 1 must be visited by the salesman, because node 1 is a depot. The salesman visits some nodes such that all nodes that are not on the tour are within the covering distance r_n of the visited nodes or the covering distance r_e of the edges on the tour. When r_n and r_e are set to zero, this is the TSP, because the salesman must visit all nodes. In the case of $r_n \geq 1$ and $r_e = 0$, the CSPNS becomes the CSP, because the edges on the tour cannot cover the unvisited nodes. We introduce two decision variables:

$$y_i = \begin{cases} 1: \text{if the salesman visits node } i, \\ 0: \text{otherwise} \end{cases} \tag{3}$$

and

$$x_{ij} = \begin{cases} 1: \text{if the salesman moves directly from node } i \text{ to node } j \left(\neq i\right), \\ 0: \text{otherwise.} \end{cases} \tag{4}$$

If the salesman moves directly from node i to node j ($x_{ij} = 1$), node i is the visited node ($y_i = 1$). In addition, we introduce a counting value f_{ij} to eliminate a sub-tour. If the salesman does not move from node i to node j, f_{ij} is set as 0. The value of f_{ij} is increased by 1 whenever the salesman visits a node. Using this notation, the CSPNS is formulated as follows:

$$\text{Minimize } \sum_{i \in V} \sum_{j \in V \setminus \{i\}} d_{ij} x_{ij} \tag{5}$$

$$\text{Subject to } y_1 = 1 \tag{6}$$

$$\sum_{h\in V\setminus\{i\}} x_{hi} = y_i \quad \forall i \in V \tag{7}$$

$$\sum_{j\in V\setminus\{i\}} x_{ij} = y_i \quad \forall i \in V \tag{8}$$

$$\sum_{j\in V} a_{ij} y_i + \sum_{j\in V}\sum_{k\in V\setminus\{j\}} b_{ijk} x_{jk} \geq 1 \quad \forall i \in V \tag{9}$$

$$\sum_{j\in V\setminus\{i\}} f_{ij} y_i - \sum_{h\in V\setminus\{i\}} f_{hi} y_i = y_i \quad \forall i \in V \tag{10}$$

$$f_{ij} \leq (n-1) x_{ij} \quad \forall i \in V,\ \forall j \in V\setminus\{i\} \tag{11}$$

$$\sum_{j\in V\setminus\{1\}} f_{1j} = 0 \tag{12}$$

$$\sum_{h\in V\setminus\{1\}} f_{h1} = \sum_{k\in V} y_k - 1 \tag{13}$$

$$f_{ij} \geq 0 \quad \forall i \in V, \forall j \in V\setminus\{i\} \tag{14}$$

$$y_i \in \{0,1\} \quad \forall i \in V \tag{15}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in V, \forall j \in V\setminus\{i\} \tag{16}$$

In this formulation, Equation (5) is the objective function that minimizes the length of the tour. Equations (6)-(16) are constraints of the CSPNS. Equation (6) specifies that the salesman must visit node 1, because node 1 is the depot. Equations (7) and (8) ensure that if the salesman directly moves from node i to node j, nodes i and j become visited nodes. In addition, each visited node should be visited exactly once. Equation (9) enforces the condition that every node is covered by the visited nodes or the edges on the tour. Equations (10)-(14) eliminate sub-tours by using flow formulation. Finally, constraints (15) and (16) define the variables as binary value.

2.2. Computational Simulations Using Mixed-Integer Programming Solver

In this section, we present optimal solutions of the CSPNS obtained using the formulation of the CSPNS and a mixed-integer programming solver. In this simulation, we used a Gurobi Optimizer 6.5.0 on a Mac Pro (3.0-GHz 8-core Intel Xeon E5) with 64 GB of memory running Mac OS X 10.11.5.Gurobi Optimizer is one of the powerful solvers and showed good performances for MIP benchmarks [9]. All simulations were performed using 16 threads, and the solver time limit was set to 12 h. The optimality tolerance of the solver was left as the Gurobi default of 0.0001.

For the benchmark instances, DIMACS [10], which is a benchmark problem of the TSP, was used. The number of nodes n was set to 50, 60, 70, 80, 90, and 100. In the simulation, 10 instances were created for each number of nodes (the values of the seed of DIMACS were 1 - 10). The covering distances r_n and r_e were set to the same value: 0, 10,000, 20,000, 30,000, 40,000, 50,000, 60,000, 70,000, and 80,000. In the case of $r_n = r_e = 0$, the problem was the TSP.

Table 1 presents the number of instances that could be solved within the 12 h

time limit. As indicated by the table, we obtained feasible solutions for a small size of instances. However, for a large size of instances (n = 90, 100), when the covering area was large, optimal solutions could not be found. In such cases, even though time limit was set to five days, the optimal solution could not be found by the Gurobi optimizer.

Figure 2 shows the average computational times required for an optimal solution to be found by the Gurobi optimizer. From Figure 2, the calculation times for the CSPNS were longer than those for the TSP ($r_n = r_e = 0$). Although the number of nodes increased slightly, the computational time increased exponentially. The results indicate that there is a need to develop a heuristic method for finding solutions in a reasonable time frame for cases of a large size of instances.

Table 1. The number of instances that could be solved within the 12 h time limit.

		Number of nodes (n)					
		50	60	70	80	90	100
$r_n = r_e = 0$	0	10	10	10	10	10	10
	10,000	10	10	10	10	10	10
	20,000	10	10	10	10	10	10
	30,000	10	10	10	10	10	9
	40,000	10	10	10	10	10	10
	50,000	10	10	10	10	10	8
	60,000	10	10	10	10	9	8
	70,000	10	10	10	10	9	8
	80,000	10	10	10	10	9	8

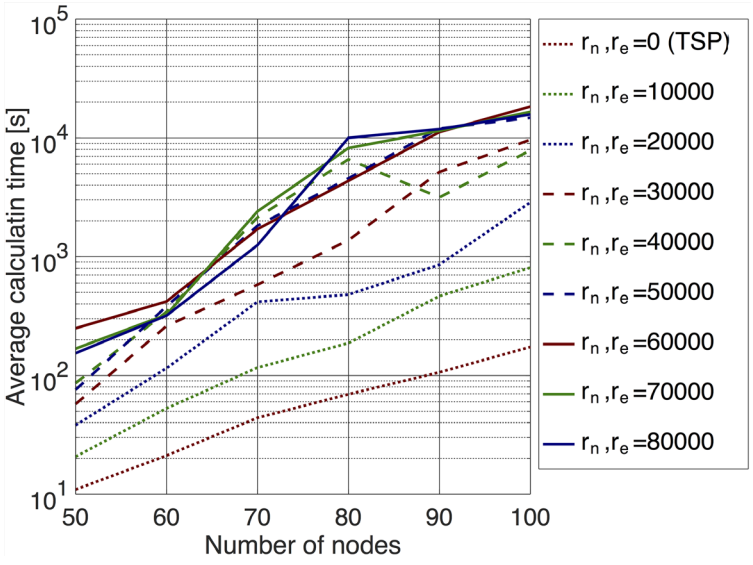

Figure 2. Average calculation times for finding an optimal solution by a Gurobi optimizer.

3. Local Search Method

In this section, we develop a heuristic method for determining the optimal or near-optimal tour in the CSPNS. Figure 3 shows an optimal tour of the TSP and the CSPNS. From these optimal tours, the shape of the optimal tour of the TSP (Figure 3(a)) and that of the CSPNS (Figure 3(b) and Figure 3(c)) are very similar. In the proposed method, this similarity is used to devise a short tour for the CSNPS.

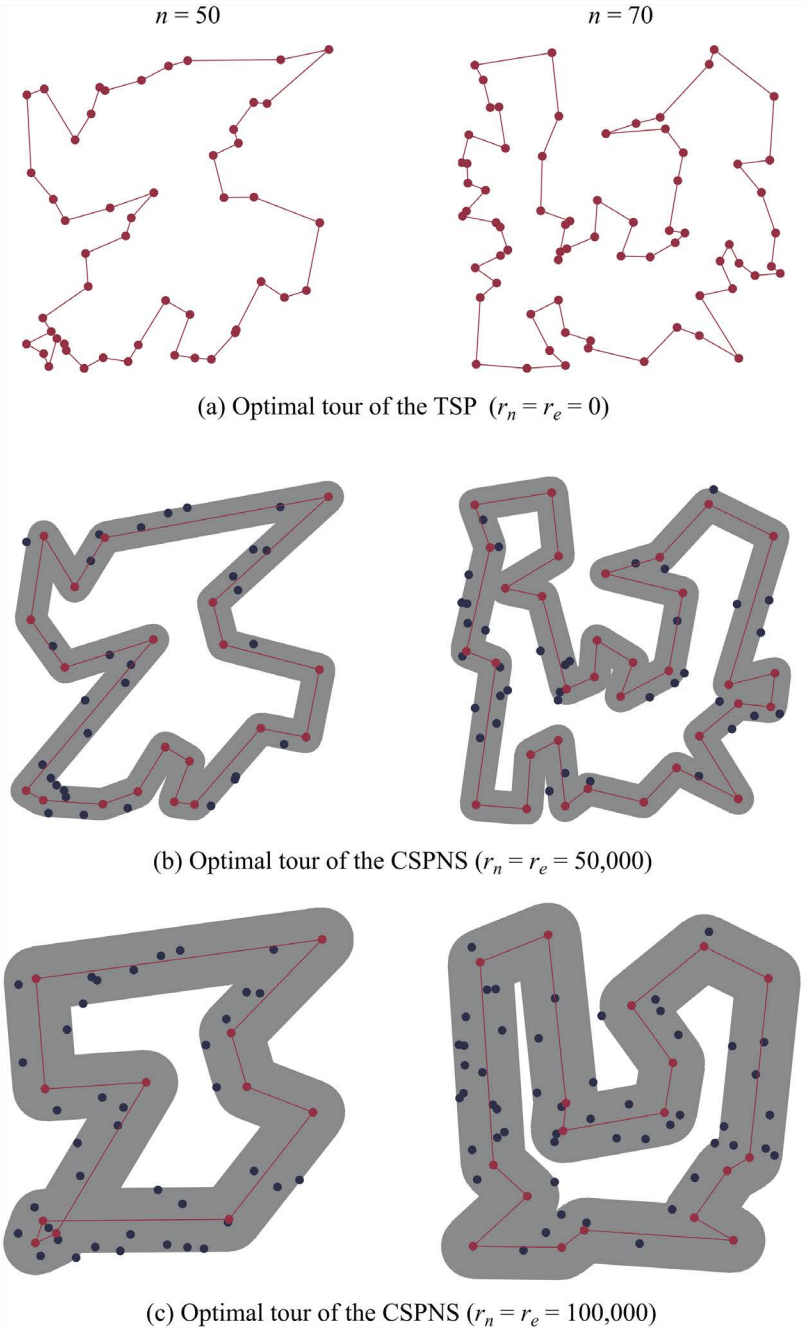

Figure 3. Optimal tours for the TSP and CSPNS. The red lines indicate the optimal tours. The red and blue circles represent the visited and unvisited nodes, respectively. The covering area is shown in gray.

In the proposed method, first, an initial tour passing through all the given nodes is constructed; that is, the TSP is solved. Second, the length of the initial tour is improved via local search methods for the TSP, such as the 2-opt, Or-opt [11], and Lin-Kernighan algorithms [12] [13] [14]. Finally, the tour is improved via local search methods, *i.e.*, eliminating a node, exchanging two nodes, exchanging two edges, and inserting a node. Figure 4 shows graphical explanation of the local search methods used.

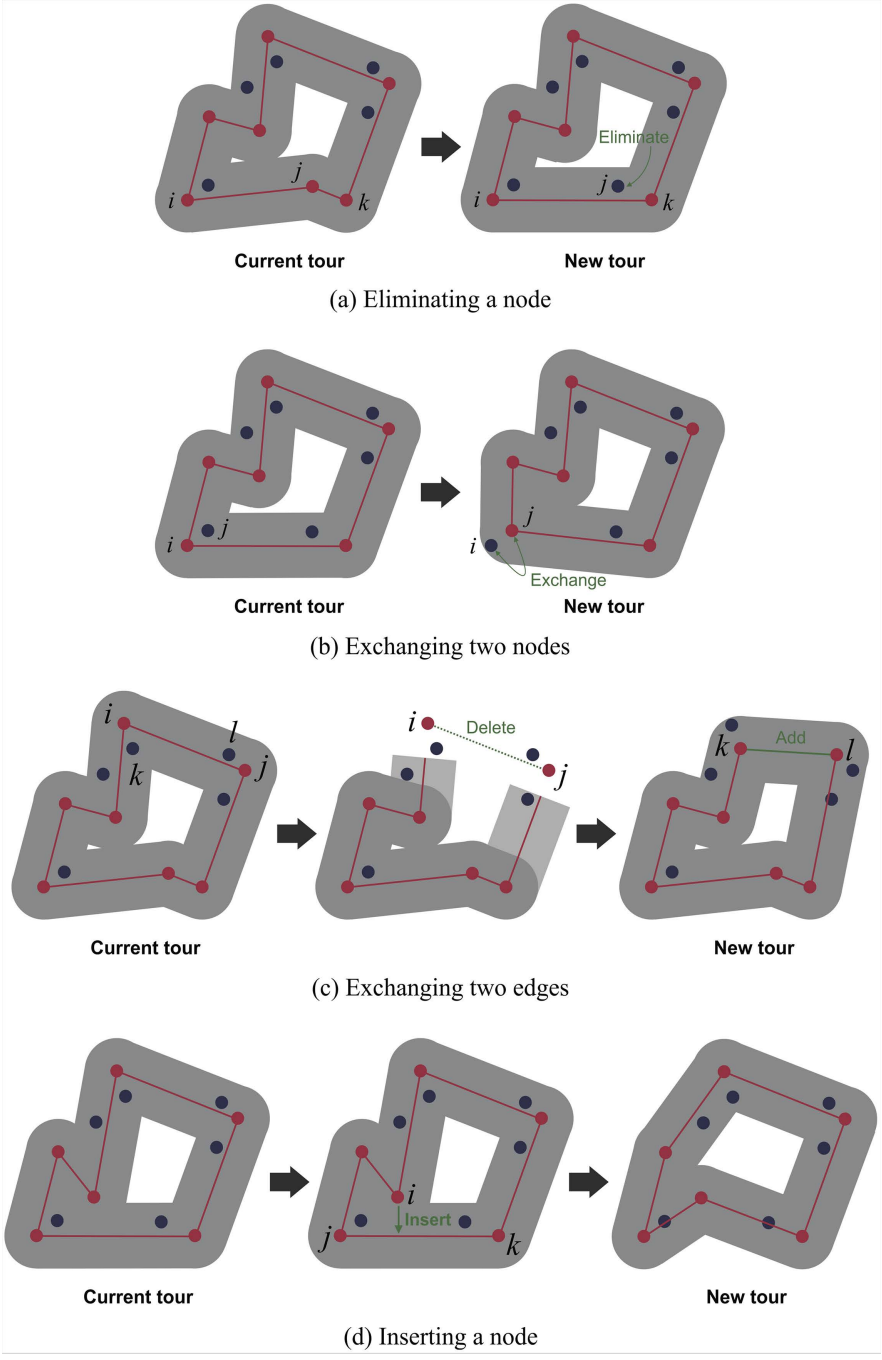

Figure 4. Graphical explanation of the local search methods used in the proposed method.

In the CSPNS, the distance from node i to node j is the same as that from node j to node i, and the distances are Euclidean distances in a two-dimensional space, rounded to the nearest integer. Therefore, if a visited node is eliminated from the current tour, the length of a new tour is inevitably shorter than that of the current tour. Eliminating a node involves removing one node from the current tour if a new tour is the feasible: all given nodes can be covered by the visited nodes or the segments on the new tour. Now, nodes i, j, and k are the $(i-1)$th, ith, and $(i+1)$th visited nodes in the current tour, respectively. Node j is eliminated from the current tour, and nodes i and k are connected, if the new tour is feasible (**Figure 4(a)**). Exchanging two nodes involves switching a visited node $i \in V'$ and an unvisited node $i \in V \setminus V'$ (**Figure 4(b)**). Here, V' is the set of visited nodes. Exchanging two edges involves switching an edge (i, j) $(i, j \in V')$ and another edge (k, l) $(k, l \in V \setminus V')$ (**Figure 4(c)**). Finally, inserting a node involves adding a visited node $i \in V'$ to an edge (j, k) $(j, k \in V')$ (**Figure 4(d)**). **Figure 5** shows a flowchart of the proposed method.

4. Results

The proposed method was coded in C and run on a Mac Pro with a 3.0-GHz 8-Core Intel Xeon E5 processor and 64 GB of RAM using the Mac OS X 10.11.5 operating system. To investigate the performance of the method, we solved the benchmark instances used in Section 2.2. The number of nodes n was set as 50, 60, 70, 80, 90, and 100. The cities were uniformly distributed in the $10^6 \times 10^6$ square, and the seed for generating the instances was set as 1 - 3. The covering distances (r_n and r_c) were set as 20,000, 40,000, 60,000, and 80,000.

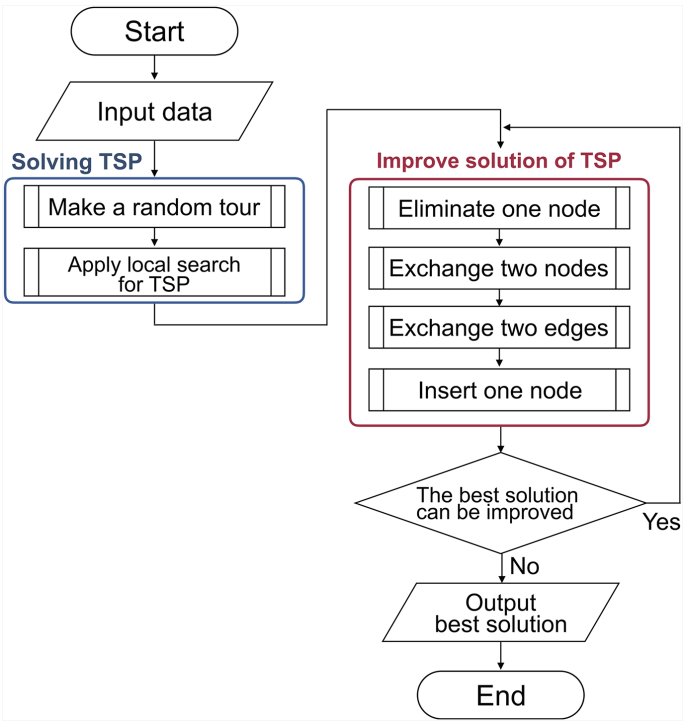

Figure 5. Flowchart of the proposed method.

In the proposed method, each instance was tested 50 times, and each run stopped when the local optimal solution was found. Initial random tours were constructed 50 times for each instance, and the initial tours were improved via the local search method for the TSP. In these experiments, we used three algorithms: the Lin-Kernighan, 2-opt, and Or-opt algorithms. Among these, the Lin-Kernighan algorithm exhibited the best performance for the TSP. The performance of the proposed method was indicated by the percentage of the gaps between the obtained solutions and the optimal solution obtained using the Gurobi optimizer.

Figure 6 shows calculation costs of the proposed method. According to the results, the proposed method can quickly obtain solutions to the CSPNS. We observe that as the covering area increased, the calculation time increased. This is because eliminating the visited nodes from the tour required considerable time.

Tables 2-7 report on the results of the proposed method for the different number of nodes. The first column is seed of the instance; the second column shows the covering distances r_n and r_e. The third to the 11th columns show the results of the proposed method. Avg, Best, and Worst denote the average, best, and worst gaps, respectively.

From Tables 2-7, for the TSP ($r_n = r_e = 0$), the Lin-Kernighan algorithm obtained better solutions than the other methods because it is the most powerful local search method. For the CSPNS, when the initial random tours were improved by the Lin-Kernighan algorithm, we obtained the best tour. The method employing the Lin-Kernighan algorithm found an optimal solution for some in

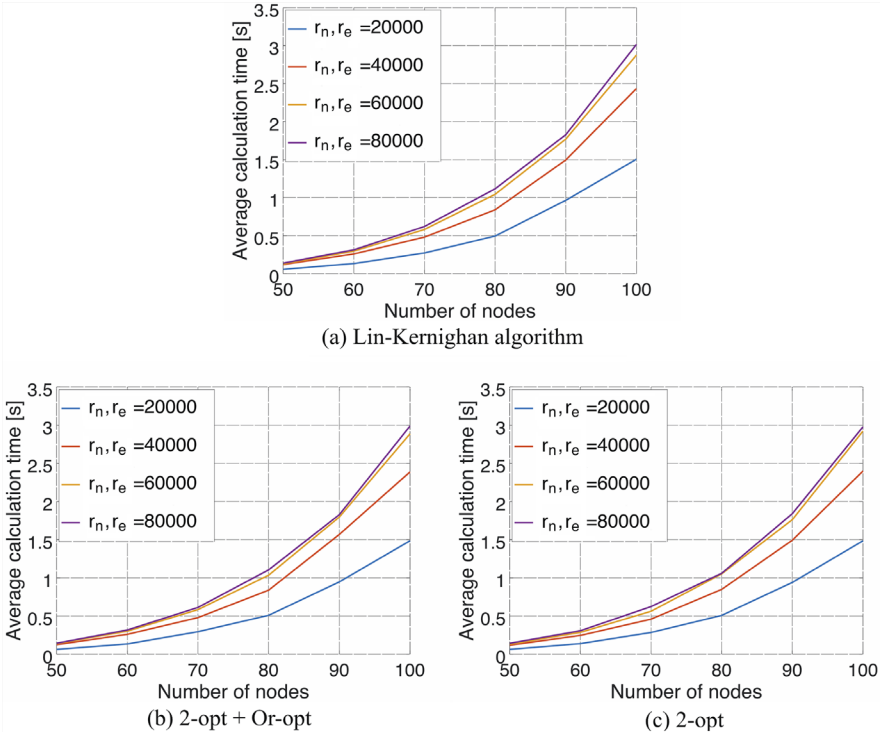

Figure 6. Average calculation times of the proposed method.

Table 2. Performances of the proposed methods for $n = 50$.

		Lin-Kernighan			2-opt + Or-opt			2-opt		
seed	r_n, r_e	Avg.	Best	Worst	Avg.	Best	Worst	Avg.	Best	Worst
1	0	1.28	0.00	3.65	2.93	0.00	8.83	4.50	0.88	9.82
	20,000	1.32	0.00	3.92	3.01	0.00	8.83	4.96	0.89	9.83
	40,000	1.69	0.00	5.11	3.39	0.00	10.35	4.79	0.00	11.17
	60,000	1.72	0.50	3.89	2.81	0.50	8.32	4.22	0.78	9.54
	80,000	5.65	2.71	9.72	7.30	2.71	13.31	7.65	2.11	16.86
2	0	0.20	0.00	1.75	3.95	0.00	7.75	3.59	0.00	12.76
	20,000	0.35	0.00	1.71	3.99	0.00	7.73	3.58	0.00	12.97
	40,000	1.46	1.06	2.92	5.13	1.30	9.10	4.78	0.72	13.01
	60,000	3.27	2.49	3.97	7.47	1.99	12.62	6.75	1.99	18.57
	80,000	3.27	1.79	3.48	10.37	0.00	17.38	6.86	0.00	23.55
3	0	0.04	0.00	0.52	4.35	0.00	11.33	5.15	0.00	14.68
	20,000	0.03	0.00	0.47	4.27	0.00	11.25	5.29	0.00	14.67
	40,000	0.13	0.00	1.36	5.46	0.00	12.81	5.86	0.00	15.77
	60,000	3.33	3.24	4.84	8.35	3.24	19.25	8.96	3.24	20.13
	80,000	0.00	0.00	0.00	5.79	0.00	16.25	6.00	0.00	15.96

Table 3. Performances of the proposed methods for $n = 60$.

		Lin-Kernighan			2-opt + Or-opt			2-opt		
seed	r_n, r_e	Avg.	Best	Worst	Avg.	Best	Worst	Avg.	Best	Worst
1	0	1.69	0.00	4.16	3.58	0.37	6.52	4.25	0.59	9.31
	20,000	1.77	0.00	4.19	3.74	0.37	6.91	4.62	0.63	10.36
	40,000	2.54	0.78	6.21	4.68	0.88	9.54	5.39	0.94	11.08
	60,000	4.52	2.03	8.74	5.35	2.64	10.25	6.39	2.22	11.58
	80,000	7.08	2.96	11.94	8.26	1.58	15.14	8.56	2.41	16.14
2	0	1.29	0.00	4.41	3.11	0.00	9.60	4.46	0.00	9.03
	20,000	1.30	0.01	4.29	3.12	0.01	9.47	4.23	0.01	8.41
	40,000	3.18	0.88	7.64	4.77	0.88	11.13	5.76	1.38	10.89
	60,000	5.22	1.99	10.23	7.04	1.99	14.47	6.73	1.56	12.60
	80,000	3.81	0.89	9.50	6.11	0.45	18.40	5.71	0.40	12.31
3	0	0.26	0.00	1.42	3.84	0.00	9.23	5.02	0.35	12.52
	20,000	0.44	0.21	1.52	3.96	0.21	9.34	5.58	0.58	12.66
	40,000	0.50	0.00	2.65	5.15	0.00	11.77	5.45	0.00	15.27
	60,000	4.12	3.33	5.89	8.76	3.33	16.61	8.77	3.33	19.12
	80,000	2.40	1.18	6.44	8.00	1.99	14.90	7.37	1.18	15.16

Table 4. Performances of the proposed methods for $n = 70$.

		Lin-Kernighan			2-opt + Or-opt			2-opt		
seed	r_n, r_e	Avg.	Best	Worst	Avg.	Best	Worst	Avg.	Best	Worst
1	0	0.69	0.00	3.49	3.20	0.00	8.24	5.59	0.09	11.33
	20,000	0.80	0.00	3.72	3.37	0.00	7.60	5.76	0.15	11.43
	40,000	2.16	1.27	6.44	5.08	1.06	10.27	7.68	1.27	13.75
	60,000	2.36	0.58	7.18	4.09	1.16	9.62	6.76	0.58	18.59
	80,000	5.23	1.69	10.11	6.73	2.97	12.00	7.64	1.26	14.02
2	0	1.18	0.00	5.94	4.80	0.75	9.18	5.21	0.13	11.30
	20,000	1.17	0.09	5.94	4.74	0.87	9.09	5.08	0.29	10.80
	40,000	3.21	2.29	7.46	7.12	2.86	12.33	6.68	2.36	14.63
	60,000	4.66	3.06	11.41	8.95	1.54	15.89	7.73	1.54	19.40
	80,000	2.74	0.55	8.67	7.03	0.39	14.12	5.61	0.60	14.45
3	0	0.72	0.00	4.85	3.77	0.01	7.46	5.23	1.47	10.34
	20,000	0.94	0.19	5.18	4.09	0.19	7.61	5.56	1.51	9.69
	40,000	1.17	0.00	6.86	5.32	0.00	9.90	6.27	1.34	12.18
	60,000	2.61	1.48	8.71	6.21	1.48	12.51	6.30	1.48	11.27
	80,000	3.99	1.96	9.74	8.39	1.96	16.97	9.59	4.57	17.42

Table 5. Performances of the proposed methods for $n = 80$.

		Lin-Kernighan			2-opt + Or-opt			2-opt		
seed	r_n, r_e	Avg.	Best	Worst	Avg.	Best	Worst	Avg.	Best	Worst
1	0	0.97	0.00	2.72	2.88	0.32	6.53	5.58	1.52	10.50
	20,000	1.46	0.44	3.43	3.39	0.74	7.09	6.04	1.96	10.94
	40,000	2.91	1.17	5.22	4.61	1.17	9.64	7.04	0.20	12.60
	60,000	3.56	1.05	6.13	4.98	0.97	10.88	6.62	2.20	13.47
	80,000	4.40	0.63	9.24	5.74	0.99	12.66	6.92	1.28	12.63
2	0	1.28	0.13	3.31	3.95	0.82	8.28	5.47	1.14	9.24
	20,000	1.43	0.23	3.52	4.06	1.02	8.33	5.42	0.87	8.68
	40,000	2.60	0.94	6.22	6.04	1.54	11.39	6.52	1.54	12.09
	60,000	3.82	1.54	8.21	7.27	1.54	13.21	7.38	3.11	12.60
	80,000	4.45	1.29	11.01	8.40	1.14	18.27	7.89	1.29	14.57
3	0	0.63	0.00	2.39	3.70	0.09	8.81	6.14	0.64	11.40
	20,000	0.99	0.24	3.09	4.14	0.34	9.69	6.77	1.03	12.44
	40,000	0.72	0.00	3.00	4.08	0.12	10.16	6.15	0.00	13.28
	60,000	4.91	3.55	7.48	8.23	4.38	14.41	9.48	3.89	17.12
	80,000	3.87	1.54	6.25	7.91	0.98	16.98	9.04	1.74	15.49

Table 6. Performances of the proposed methods for $n = 90$.

		Lin-Kernighan			2-opt + Or-opt			2-opt		
seed	r_n, r_e	Avg.	Best	Worst	Avg.	Best	Worst	Avg.	Best	Worst
1	0	0.92	0.21	5.69	3.36	0.50	8.04	6.67	0.63	14.39
	20,000	1.24	0.42	6.31	3.73	0.87	8.47	6.93	1.43	15.06
	40,000	3.36	1.77	7.03	5.08	1.66	11.45	7.19	2.46	17.32
	60,000	4.81	1.96	9.36	6.11	1.53	11.66	7.78	1.96	16.62
	80,000	3.81	0.80	8.64	6.36	2.34	12.55	6.79	2.38	12.52
2	0	1.38	0.14	4.16	4.01	0.14	10.61	5.53	1.07	9.93
	20,000	1.26	0.05	3.78	3.92	0.08	10.52	5.07	0.32	9.72
	40,000	3.23	1.17	5.91	6.21	1.70	15.06	7.38	1.53	13.03
	60,000	4.22	0.78	7.39	6.48	1.00	15.14	6.91	0.78	14.25
	80,000	6.10	0.37	10.59	8.97	0.00	19.63	9.26	0.37	20.18
3	0	0.80	0.00	4.01	4.34	0.38	9.45	5.72	1.39	11.72
	20,000	1.10	0.33	4.27	4.86	0.68	9.84	6.26	1.52	12.36
	40,000	1.76	0.28	6.39	5.50	0.56	10.07	6.23	0.91	13.42
	60,000	5.81	3.70	9.78	8.34	3.70	15.15	8.85	3.70	14.69
	80,000	3.99	0.54	11.08	8.16	0.54	13.93	8.48	0.10	21.81

Table 7. Performances of the proposed methods for $n = 100$.

		Lin-Kernighan			2-opt + Or-opt			2-opt		
seed	r_n, r_e	Avg.	Best	Worst	Avg.	Best	Worst	Avg.	Best	Worst
1	0	1.53	0.00	4.25	3.89	0.98	8.29	6.41	1.63	12.26
	20,000	1.84	0.23	4.85	4.31	1.31	8.77	6.48	2.09	12.99
	40,000	4.35	1.79	6.99	6.87	2.99	12.49	8.73	3.11	13.23
	60,000	4.08	2.10	8.91	6.32	2.25	14.44	7.04	3.10	14.03
	80,000	7.73	3.20	13.03	9.29	0.66	16.04	9.03	3.91	16.35
2	0	1.35	0.15	3.28	4.11	0.42	9.83	4.95	0.42	9.39
	20,000	1.66	0.21	3.51	4.45	0.63	10.86	5.15	0.49	9.78
	40,000	2.77	0.92	5.49	5.98	1.04	14.09	5.79	1.54	13.38
	60,000	3.18	0.30	7.48	5.51	0.47	12.37	5.59	0.21	12.98
	80,000	6.40	0.96	12.90	8.87	1.85	20.30	8.84	1.64	19.80
3	0	1.03	0.00	5.87	4.92	0.50	9.72	6.10	1.22	11.70
	20,000	1.12	0.13	5.66	5.13	0.83	10.14	6.22	1.41	12.10
	40,000	2.39	0.55	10.02	6.87	1.46	12.28	6.89	0.92	11.28
	60,000	6.27	3.59	9.69	10.28	0.29	17.13	10.35	4.53	17.35
	80,000	4.24	1.87	8.95	8.19	1.87	17.77	7.43	2.31	14.54

stances. In the proposed method, first, the tour of the TSP was constructed; then, the nodes on the tour were deleted. These results indicate that if a good tour of the TSP can be constructed, a good tour of the CSPNS can also be constructed. However, as the covering area increases, the quality of the solution for the CSPNS becomes worse. The proposed method is quickly trapped by local minima because it employs the local search algorithm. Several methods for avoiding local minima have been proposed, such as Tabu Search [5] [6], simulated annealing [2], and genetic algorithms [3]. We expect that applying such algorithms to the proposed method can yield better solutions and an improved performance.

5. Conclusions

This paper defines and formulates a new CSP called the CSPNS. We presented mixed-integer linear programming formulations of the CSPNS, and by using the formulations and a mixed-integer programming solver, optimal solutions of the CSPNS were obtained. However, if the number of nodes was large, a long time was needed to find the optimal solution.

To find near-optimal solutions quickly, we proposed a heuristic method employing simple local search methods. Experimental results for a set of benchmark instances show that the proposed method quickly obtained good solutions. For some instances, the optimal solution was found in a few seconds. However, the tour obtained by the proposed method is not a global optimum but a local optimum. In future work, powerful meta-strategies, such as genetic algorithms, Tabu Search, and simulated annealing can be employed to improve the search capability of the proposed method.

References

[1] Danzig, G., Fulkerson, R. and Johnson, S. (1954) Solution of a Large Scale Traveling Salesman Problem. *Operations Research*, **2**, 393-410.

[2] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) Optimization by Simulated Annealing. *Science*, **220**, 671-680. https://doi.org/10.1126/science.220.4598.671

[3] Grefenstette, J.J., Gopal, R., Rosmaita, B.J. and Gucht, D.V. (1985) Genetic Algorithms for the Traveling Salesman Problem. *Proceedings of the 1st International Conference on Genetic Algorithms*, Erlbaum Associates Inc., Hillsdale, 160-168.

[4] Dorigo, M. and Gambardellab, L.M. (1997) Ant Colonies for the Traveling Salesman Problem. *Biosystems*, **43**, 73-81. https://doi.org/10.1016/S0303-2647(97)01708-5

[5] Glover, F. (1989) Tabu Search I. *ORSA Journal on Computing*, **1**, 190-206. https://doi.org/10.1287/ijoc.1.3.190

[6] Glover, F. (1990) Tabu Search II. *ORSA Journal on Computing*, **2**, 4-32. https://doi.org/10.1287/ijoc.2.1.4

[7] Hasegawa, M., Ikeguchi, T. and Aihara, K. (1997) Combination of Chaotic Neurodynamics with the 2-Opt Algorithm to Solve Traveling Salesman Problems. *Physical Review Letters*, **79**, 2344-2347. https://doi.org/10.1103/PhysRevLett.79.2344

[8] Current, J.R. and Schilling, D.A. (1989) The Covering Salesman Problem. *Trans-*

portation Science, **23**, 208-213. https://doi.org/10.1287/trsc.23.3.208

[9] Gurobi Optimizer. http://www.gurobi.com/index

[10] Johnson, D.S., McGeoch, L.A., Glover, F. and Rego, C. (2000) 8th DIMACS Implementation Challenge: The Traveling Salesman Problem. http://dimacs.rutgers.edu/Challenges/TSP/index.html

[11] Or., I. (1967) Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking. Ph.D. Thesis, Northwestern University, Illinois.

[12] Lin, S. and Kernighan, B.W. (1973) An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, **21**, 498-516. https://doi.org/10.1287/opre.21.2.498

[13] Held, M. and Karp, R.M. (1970) The Traveling-Salesman Problem and Minimum Spanning Trees. *Operations Research*, **18**, 1138-1162. https://doi.org/10.1287/opre.18.6.1138

[14] Held, M. and Karp, R.M. (1971) The Traveling-Salesman Problem and Minimum Spanning Trees: Part II. *Mathematical Programming*, **1**, 6-25. https://doi.org/10.1007/BF01584070