RAIRO-Oper. Res. 52 (2018) 577–594
https://doi.org/10.1051/ro/2017090

RAIRO Operations Research
www.rairo-ro.org

A GENERALIZED MODEL AND A HEURISTIC ALGORITHM FOR THE LARGE-SCALE COVERING TOUR PROBLEM

Keisuke Murakami*

Abstract. The covering tour problem (CTP) is defined on a graph, where there exist two types of vertices. One is called *visited vertex*, which *can* be visited. The other is called *covered vertex*, which *must* be covered but cannot be visited. Each visited vertex covers a subset of covered vertices, and the costs of edges between visited vertices are given. The objective of the CTP is to obtain a minimum cost tour on a subset of visited vertices while covering all covered vertices. In this paper, we deal with the large-scale CTPs, which are composed of tens of thousands of vertices; in the previous studies, the scales of the instances in the experiments are at most a few hundred vertices. We propose a heuristic algorithm using local search techniques for the large-scale CTP. With computational experiments, we show that our algorithm outperforms the existing methods.

Mathematics Subject Classification. 90C06, 90C27, 90C59

Received May 1, 2017. Accepted December 17, 2017.

1. Introduction

The covering tour problem (CTP) is defined on a graph $G = (V \cup W, E \cup E')$, where V is a set of vertices that *can* be visited and W is a set of vertices that *must* be covered but cannot be visited. The edge set E is the set of undirected edges between two vertices in V (*i.e.*, $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$) and E' is the set of undirected edges between $v_k \in V$ and $v_l \in W$ (*i.e.*, $E' = \{(v_k, v_l) : v_k \in V, v_l \in W)\}$. Let $c_{ij} > 0$ be the cost of edges $(v_i, v_j) \in E$ and $d_{kl} > 0$ be the cost of edges $(v_k, v_l) \in E'$. Each vertex in V covers a subset of W. In general, $v_k \in V$ is likely to cover $v_l \in W$ when d_{kl} is small. The objective of the CTP is to obtain a minimum cost tour covering all $v_l \in W$; the tour consists of a subset of V. We refer to $v_i \in V$ as the *visited vertex* and $v_l \in W$ as the *covered vertex* (Fig. 1). The CTP is NP-hard as it reduces to the traveling salesman problem when each vertex of W is covered by just one vertex in V.

There are several applications of the CTP. For example, the construction of tours for visiting health care teams in developing countries is equivalent to solving the CTP. Traveling health care teams can only access a limited number of villages due to infrastructural restrictions, but the health care team must meet every person at least once [7, 14]. The health care team's goal is to minimize travel time to see as many patients as possible. Another example is to determine the locations of post boxes among a set of candidates [17]. The post boxes must be located within a reasonable distance from every household. Then, the aim of the post office is to minimize

Keywords and phrases: Covering tour problem, traveling salesman problem, set-covering problem, local search techniques, large-scale problem.

Faculty of Business and Commerce, Kansai University, 564-8680 Osaka Prefecture, Suita, Japan.

* Corresponding author: k_mura@kansai-u.ac.jp

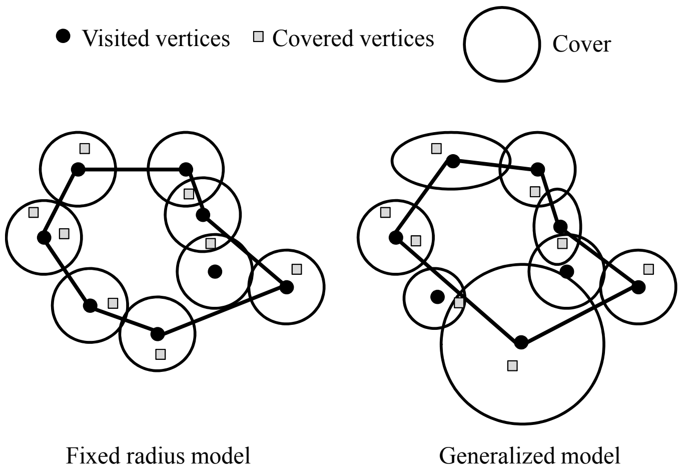

FIGURE 1. Examples of two models.

the travel time of a collection route through all post boxes. A similar example would be to locate a number of regional distribution centers among a set of candidate sites in such a way that all customers are within reasonable distance from at least one regional distribution center and that the cost of delivery and pick-up routes is minimized [6]. This example is also applied to a decision of the location of satellite distribution centers to cater humanitarian supplies [23].

1.1. Related studies and our contribution

Several studies on the CTP have been conducted. The problem was first introduced by Current and Schilling [6]. Although they named it "covering salesman problem (CSP)", it is a special case of CTP. Gendreau *et al.* presented a new formulation and proposed the first exact algorithm, as well as a heuristic [10]. Baldacci *et al.* presented a model and three scatter-search algorithms for the CTP. They also formulated the CTP as a mixed integer programming and proposed several valid inequalities [3]. Salari and Naji-Azimi proposed an integer linear programming based heuristic algorithm [27]. Golden *et al.* generalized the CTP, where each vertex is covered at least several times, and they proposed local search algorithms for the generalized CTP [11]. Salari *et al.* proposed a hybrid heuristic algorithm by combining the ant colony optimization algorithm with the dynamic programming technique [28]. Farahani *et al.* surveyed several covering problems in facility location, including the CTP [8]. Motta *et al.* proposed a GRASP meta-heuristic for a generalized CTP in which the vertices of W can also be visited [21]. Hachicha *et al.* introduced the multi-vehicle covering tour problem (m-CTP) [13] and Ha *et al.* proposed an exact algorithm and a metaheuristic for the m-CTP [12]. Jozefowiez *et al.* investigated the bi-objective CTP [16], where one objective function is the minimization of the tour length and the other is the minimization of radius of the cover. They have proposed a solution method combining a genetic algorithm with a branch-and-cut algorithm.

Moreover, there are several similar problems to the CTP, such as the (hub) location-routing problems. The location-routing problem (LRP) is not unique and well-defined. Thus, Nagy and Salhi defined the problem as "location planning with tour planning aspects taken into account" [22]; this is a combination of facility location and multi-depot vehicle routing problem. Some facility are located, and tours through customers are obtained, considering each facility as a depot. Each customer must be visited by one of the tours. In the problem, there exist two types of vertices, *i.e.*, the potential facility (depot) vertex and customer vertex; these vertices are equivalent to the visited and covered vertices in the CTP, respectively. The CTP does not find the tours through covered vertices (customers), but obtains a tour through visited vertices (facilities). On the other hand, the LRP does not find a tour through visited vertices (facilities), but constructs the tours through covered vertices (customer). Prodhon and Prins presented a survey of the related works on LRP [24]. The hub location-routing problem (HLRP) is closely similar to the LRP [26]. The only difference from the LRP is that we also find a tour thorough facility vertices in HLRP. One of the applications of HLRP is an organization participating in disaster relief operations in developing countries, where intermediate depots are set up and relief teams transport the goods from each intermediate depot to the disaster victims [19, 25]. Andrade *et al.* introduced the k-interconnected multi-depot multi-traveling salesman problem (k-IMDMTSP) [1], which is a special case of the HLRP. The k-IMDMTSP permits only a single (or empty) cycle for each facility vertex.

In this paper, we generalize the model of CTP. Although previous studies do not specify the definition of cover, the costs d_{kl} of edges $(v_k, v_l) \in E'$ are defined in their experiments by direct distances in Euclidean space and if d_{kl} is smaller than a certain fixed value d (*i.e.*, radius of the cover), v_k covers v_l. We call this the " fixed radius model" (Fig. 1). It seems that some of the existing algorithms provide good solutions in the fixed radius model. However, the ability to cover may differ according to the visited vertex and there are barriers such as mountains and river in the real world. Hence, even if the Euclidean distance between $v_k \in V$ and $v_l \in W$ is short, v_k cannot always cover v_l. We therefore determine a subset of W which is covered by each visited vertex in the generalized model, when we consider the realistic conditions (Fig. 1).

Two heuristics for the CTP have been proposed in previous studies [6, 10]. One is called the "2-stage method", where the set-covering problem (SCP) and the traveling salesman problem (TSP) are considered separately within the CTP. In the first stage, we find a minimum subset of V which covers all $v_l \in W$; this corresponds to solving the SCP. In the second stage, we obtain a shortest tour that consists of the subset of V found in the first stage; this corresponds to solving the TSP. The solution (tour) to the TSP is finally a solution to the CTP. However, the 2-stage method is not appropriate for the generalized CTP model, because in the first stage, the solution to the SCP might not be good. For example, when several vertices are covered by "a" visited vertex or by "a few" visited vertices, the 2-stage method in the first stage (SCP) selects the former, because the first stage minimizes the number of visited vertices belonging to the solution (tour); this would lead to a good solution in the fixed radius model. However, in the generalized model this is not an appropriate selection because the cost might be lower to visit "a few" vertices than "a" vertex, as shown in Figure 2. Note that even in the fixed radius model, the number of visited vertices included in the optimal solution might not be lower than that included in the non-optimal solution.

The other heuristic is proposed by Gendreau *et al.* [10]. They combine the GENIUS heuristic for the TSP [9] with the PRIMAL1 heuristic for the SCP [2]. This method reconstructs the tour by using GENIUS, every time one vertex that is to be added to the tour is determined by using PRIMAL1. The Gendreau's heuristic seems to find a good (near-optimal) solution for both models. However, we are afraid that this heuristic cannot perform well for large-scale problems because the GENIUS heuristic is not likely to find a good feasible solution to the large-scale TSP. The performances of the GENIUS heuristic have been tested by our preliminary experiments, where we compared the results of the GENIUS heuristic with optimal solutions in several instances. The GENIUS heuristic could find good solutions for small-scale problems, but it could not perform well for large-scale problems.

We therefore propose an algorithm for the large-scale CTP, even though previous studies do not target the large-scale problems (the instance scale of the experiments in the previous studies is at most $|V| = 100$ and $|W| = 500$). Our algorithm is based on local search technique, called "ruin and recreate method [29]", where we solve a subproblem using a type of labeling algorithm and heuristic algorithm. This is iterated until the termination condition is satisfied. In computational experiments, we compare the results obtained using our local search algorithm with those using the two existing methods in the fixed radius model and the generalized model. We show that our algorithm outperforms the existing methods in the generalized model.

This paper is organized as follows. Section 2 presents the generalized model and formulation. In Section 3, we propose an algorithm, where we explain how to generate a subproblem and we present a solution method for the subproblem. In Section 4, we compare the results obtained using our algorithm with those using the existing methods. Section 5 concludes this study.

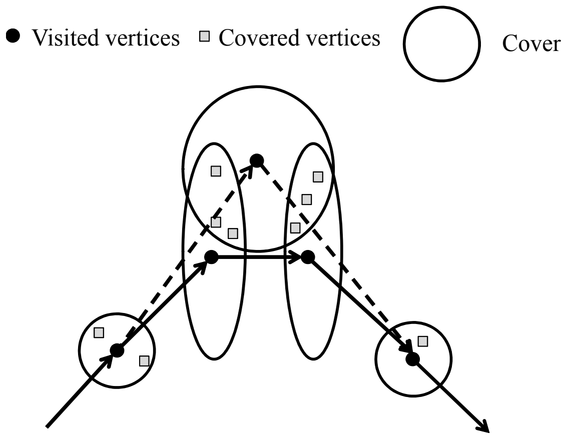

FIGURE 2. Paths in the generalized model.

2. Model

The generalized CTP model can be formulated as an integer linear program as follows [10]. Note that our formulation is not exactly the same as that of [10], because the model of [10] considers a set of vertices that *must* be visited ($i.e.$, $T \subset V$) though our model does not address it. Here, we define "tour" as a feasible solution which covers all vertices of W; in particular, an optimal solution is called "optimal tour". We let $V = \{v_1, v_2, ..., v_n\}$. For $v_k \in V$, let y_k be a binary variable equal to 1, if vertex v_k belongs to an optimal tour. Otherwise y_k equals 0. For $v_i, v_j \in V$ and $i < j$, let x_{ij} be a binary variable equal to 1, if edge $(v_i, v_j) \in E$ belongs to an optimal tour. Otherwise, x_{ij} equals 0. Let δ_{lk} denote a binary coefficient equal to 1 if $v_l \in W$ can be covered by $v_k \in V$. In the fixed radius model, a threshold value d is given, and if the Euclidean distance between vertices $v_l \in W$ and $v_k \in V$ is shorter than or equal to d, $\delta_{lk} = 1$; otherwise, $\delta_{lk} = 0$. On the other hand, in the generalized model, we assume that δ_{lk} is given on the basic of geographical conditions. We also define $S_l = \{v_k \in V | \delta_{lk} = 1\}$ for every $v_l \in W$. Then, $|S_l| \geq 1$ for all $v_l \in W$.

$$\min \sum_{i<j} c_{ij} x_{ij} \tag{2.1}$$

$$\text{s.t.} \sum_{v_k \in S_l} y_k \geq 1, \quad \forall v_l \in W \tag{2.2}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2y_k, \quad \forall v_k \in V \tag{2.3}$$

$$\sum_{\substack{v_i \in S, v_j \in V \setminus S \\ \text{or } v_j \in S, v_i \in V \setminus S}} x_{ij} \geq 2(y_t + y_u - 1),$$
$$S \subset V, 2 \leq |S| \leq n-2, v_t \in S, v_u \in V \setminus S \tag{2.4}$$

$$x_{ij} \in \{0, 1\}, \quad 1 \leq i < j \leq n \tag{2.5}$$

$$y_k \in \{0, 1\}, \quad v_k \in V \tag{2.6}$$

The objective function (2.1) represents the minimization of the sum of the costs. Equation (2.2) ensures covering constraints, whereas equation (2.3) ensures degree constraints. Equation (2.4) expresses subtour breaking constraints, and equations (2.5) and (2.6) ensure x_{ij} and y_k are binary variables.

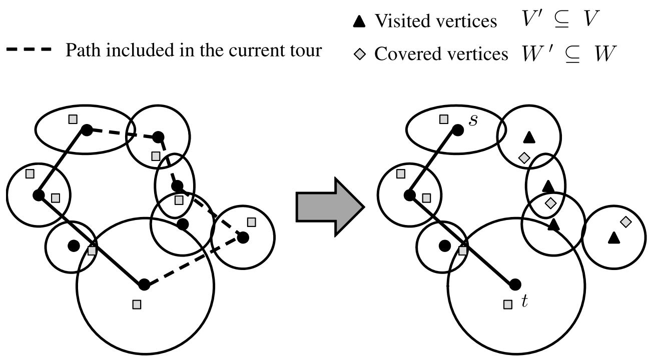
Path included in the current tour
▲ Visited vertices $V' \subseteq V$
◇ Covered vertices $W' \subseteq W$
s
t

Figure 3. Generation of subproblem.

3. Algorithm

Our solution method consists of the following three algorithms:

1. Ruin and recreate heuristic (R&R)
2. Labeling algorithm (LAB)
3. Heuristic algorithm (HEU)

The main algorithm is R&R, where LAB is used as a subroutine. Furthermore, HEU is used as a subroutine inside LAB. We present the R&R, LAB, and HEU in Sections 3.1, 3.2, and 3.3, respectively.

3.1. Ruin and recreate heuristic

The R&R is a local search algorithm [29]. It first finds an initial tour (solution). Next, the tour is ruined and a subproblem of the CTP is generated. Then, a new tour is recreated (the subproblem is solved). If the cost of the new tour is improved, we update the tour. This is repeated until the termination condition is satisfied. Our R&R for the CTP is described as follows.

Algorithm 3.1. R&R for the CTP

```
Step 0: Find an initial tour and set it to current tour.
Step 1: Destroy a part of the current tour and generate a subproblem of the CTP.
Step 2: Recreate the partial tour (solve the subproblem).
Step 3: If the cost of the current tour is improved by Step 2, update the current tour.
Step 4: If the termination condition is satisfied, exit; otherwise, return to Step 1.
```

In Step 0, an initial tour is found; the detailed explanation is provided in Section 3.3. Step 1 ruins a part of the current tour and generates a subproblem of the CTP. The subproblem is generated as follows. We randomly choose a path included in the current tour (*i.e.*, a part of the current tour) as shown in the left schematic of Figure 3. The path is temporarily ruined, *i.e.*, the vertices on the path are not visited. As a result, some vertices in W are not covered. We find all vertices that are not covered and let W' denote the set of these uncovered vertices. Next, we find the visited vertices, each of which covers at least one vertex in W' and denote V' to be the set of these visited vertices. Let s and t refer to the two ends of the path (Fig. 3). Then, the shortest path problem with covering constraints is generated, where the source vertex is s, the target vertex is t, the set of the visited vertices is V' and the set of the covered vertices is W'. The shortest path problem is the subproblem of the CTP. We name the subproblem of the CTP "shortest path problem with covering constraints (SPCC)". Step 2 recreates the path that covers all vertices in W', *i.e.*, it is equivalent to solving the SPCC. We also propose a labeling algorithm for the SPCC in Section 3.2.

3.2. Labeling algorithm for the SPCC

A shortest path problem is often solved using the labeling algorithm, which is an improvement over the dynamic programming method. Examples of well known labeling methods are the Dijkstra's algorithm and the Bellman–Ford algorithm. However, the SPCC is not a standard shortest path problem due to covering constraints, and so we cannot use such well known algorithms without any change. We therefore propose a new labeling method for the SPCC.

The number of labels on a vertex is not always one for the SPCC, even though that is one for standard shortest path problems, because there are multiple criteria of labels in the SPCC. For example, two labels (Label 1 and Label 2) exist on a vertex; Label 1 covers a set $W_1 \subset W$ and the objective function is c_1, and Label 2 covers a set $W_2 \subset W$ and the objective function is c_2. When $W_2 \subset W_1$ and $c_2 < c_1$, we cannot judge which label is superior. Thus, the two labels must remain on the vertex. Consequently, a lot of labels might remain on each vertex in a Pareto-optimal fashion, and this causes the computation time to be long.

In our labeling algorithm for the SPCC, the number of labels on each vertex reduces to one to find a feasible solution within reasonable time, although the optimality is lost. In standard labeling algorithm such as the Dijkstra's algorithm, the label on a vertex is represented by the cost of the path from the source to that vertex. We call the path from the source to that vertex as the "previous path" of that vertex. However, in SPCC, we must evaluate a label by using not only the cost of the previous path but also the vertices covered by the previous path. To generate an objective problem (not multi-objective problem), we define the objective function as the sum of the cost and the number of covered vertices (*i.e.*, cost + $w\times$ (number of covered vertices); w implies a weight). When we use the objective function, the number of labels on each vertex is always one. However, this objective function has a serious drawback that the labeling algorithm is likely to terminate (*i.e.*, the updating of the labels terminates.) before a feasible path is found. We are afraid that we cannot appropriately evaluate the labels by using only the previous path. We therefore consider not only the previous path of that vertex but also the path from that vertex to the target vertex. We call the path from that vertex to the target vertex as the "forward path". That is, we find a feasible path including that vertex and the cost of the feasible path is used as a label on that vertex (we present a heuristic algorithm for finding a feasible path in Sect. 3.3). The feasible path covers all vertices $v_l \in W'$. We expect the cost of the feasible path to be one of the most appropriate labels.

We describe our labeling algorithm (LAB) for the SPCC below, where a label is expressed by the cost of a feasible path. Let $l(i)$ be the label on vertex $v_i \in V'$. Let R be a vertex set that cannot lead to update other vertices. Let $p(i)$ denote the cost of the previous path of the vertex $v_i \in V'$ and let $f(i)$ denote the cost of the forward path from vertex $v_i \in V'$.

Algorithm 3.2. LAB for the SPCC

```
Step 0: Set $l(s) \leftarrow 0$ and $p(s) \leftarrow 0$. Set $l(k) \leftarrow \infty$ and $p(k) \leftarrow \infty$ for all $v_k \in V' \setminus \{s\}$. Set $R \leftarrow \emptyset$.
Step 1: Find a vertex $v_j \in V' \setminus R$ such that $l(j) = \min_{v_k \in V' \setminus R} l(k)$.
Step 2: For all $v_i \in V' \setminus R$, if $l(i) > p(j) + c_{ji} + f(i)$, then set $l(i) \leftarrow p(j) + c_{ji} + f(i)$, $p(i) \leftarrow p(j) + c_{ji}$ and $R \leftarrow R \setminus \{v_i\}$.
Step 3: Set $R \leftarrow R \cup \{v_j\}$.
Step 4: If $V' = R$, then exit; otherwise, return to Step 1.
```

Step 0 initializes each parameter. In Step 1, the vertices with minimum label value are found. Step 2 updates the labels, where $p(j) + c_{ji} + f(i)$ represents the cost of a feasible path and we need to find the forward path from vertex v_i to compute $f(i)$. The method for finding the forward path is presented in Section 3.3. Note that we do not need the extra method for finding the previous path, because it is found in LAB ($p(i) \leftarrow p(j) + c_{ji}$). Step 3 updates the value of R. In Step 4, if the termination condition is satisfied, then exit; otherwise, return to Step 1.

3.3. Heuristic algorithm for finding the forward path

We propose a heuristic algorithm for the problem of finding the forward path from vertex v_i. The problem must quickly be solved in addition to finding high-quality solution because it is computed many times. We notice that the problem has a characteristic of the SCP. A fast and high-accuracy algorithm for the SCP has been proposed by Chvata [4]; this algorithm is based on a greedy heuristic, where we select a vertex covering more vertices for its cost. For each vertex, an evaluation value is computed, *i.e.*, (number of vertices covered by the vertex)/(cost of the vertex). Then, we find the vertex with the largest evaluation value and that identified vertex is added to the solution. This is repeated until all the vertices are covered. This heuristic is likely to find a good feasible solution quickly.

Here in, we improve the greedy heuristic to find the forward path from vertex v_i. This improved heuristic starts with vertex v_i and then repeats the determination of the next vertex to be visited using the evaluation value similar to the greedy heuristic for the SCP. That is, the improved heuristic combines the greedy heuristic for the SCP with the nearest neighbor method for the TSP. We explain this improved heuristic for finding the forward path from vertex v_i below.

Let $\tilde{W}' \subseteq W'$ be the set of the vertices that are *not* covered by the previous path of v_i. Let W'_k denote the set of the vertices covered by the visited vertex $v_k \in V'$. Let $\tilde{W}'_k$ denote the set of uncovered vertices and covered by vertex v_k, where we initialize $\tilde{W}'_k = \tilde{W}' \cap W'_k$. Then, the next vertex to be visited is determined in the SCP fashion; we use the ratio $|\tilde{W}'_k|/c_{ik}$. The next vertex is represented by the following equation:

$$v_{k_0} = \arg\max_{v_k \in V'} \left\{|\tilde{W}'_k|/c_{ik}\right\}. \tag{3.1}$$

When the next visited vertex v_{k_0} is found using equation (3.1), the uncovered set $\tilde{W}'_k$ is updated by

$$\tilde{W}'_k = \tilde{W}'_k \setminus W'_{k_0}, \quad \forall v_k \in V'. \tag{3.2}$$

Procedures (3.1) and (3.2) are iterated until all the vertices are covered ($\tilde{W}'_k = \emptyset$, $\forall v_k \in V'$). Finally, the target vertex, v_{target}, is visited. Furthermore, the solution (cost and path) is improved by using the 2-opt method and removing the redundant vertices. Let $\boldsymbol{\sigma}'_i$ be the forward path from vertex v_i. We describe the heuristic for finding the forward path from vertex v_i (HEU) as follows.

Algorithm 3.3. Heuristic for finding the forward path (HEU)

```
Step 0: Initialize $\boldsymbol{\sigma}'_i \leftarrow v_i$, $v_{k_0} \leftarrow v_i$ and $\tilde{W}'_k$ for all $v_k \in V'$.
Step 1: Set $v_{k_0} \leftarrow \arg\max_{v_k \in V'} \{|\tilde{W}'_k|/c_{k_0,k}\}$.
Step 2: Set $\boldsymbol{\sigma}'_i \leftarrow \boldsymbol{\sigma}'_i + v_{k_0}$.
Step 3: Update $\tilde{W}'_k \leftarrow \tilde{W}'_k \setminus W'_{k_0}$ for all $v_k \in V'$.
Step 4: If $\tilde{W}'_k = \emptyset$ for all $v_k \in V'$, set $\boldsymbol{\sigma}'_i \leftarrow \boldsymbol{\sigma}'_i + v_{target}$ and go to Step 5; otherwise, return to Step 1.
Step 5: Apply 2-opt method to $\boldsymbol{\sigma}'_i$ and update $\boldsymbol{\sigma}'_i$.
Step 6: Remove redundant vertices from $\boldsymbol{\sigma}'_i$.
```

Step 0 initializes each parameter. Step 1 selects the vertex with maximum evaluation value as the next vertex to be visited (Eq. (3.1)). In Step 2, the vertex v_{k_0} is allocated to the tail of the path $\boldsymbol{\sigma}'_i$. In Step 3, the uncovered set $\tilde{W}'_k$ is updated (Eq. (3.2)). Step 4 checks whether the covering constraints are satisfied or not. In Step 5, the path $\boldsymbol{\sigma}'_i$ is improved. It is known that the solutions to the nearest neighbor algorithms for the TSP are improved by using the 2-opt method [18]. Thus, we use the 2-opt method here to improve the solution. The input to the 2-opt method is a path $\boldsymbol{\sigma}'_i$ obtained previously from executing Steps 0–4. Note that we ignore everything but the vertices that are included in the path $\boldsymbol{\sigma}'_i$, *i.e.*, this 2-opt method is the same as that often used to solve the TSP.

In Step 6, we remove the redundant vertices from path $\boldsymbol{\sigma}'_i$ because this heuristic might generate redundant vertices. Even if the redundant vertices are removed, all $v_l \in W'$ remain to be covered. The method for removing the redundant vertices is next explained. We first enumerate the critical vertices from path $\boldsymbol{\sigma}'_i$. If $v_l \in W'$ is covered by only one visited vertex, that visited vertex is critical. Then, let $Cr(\boldsymbol{\sigma}'_i) \subseteq V'$ denote the critical vertex set in path $\boldsymbol{\sigma}'_i$ and let $W'_{Cr(\boldsymbol{\sigma}'_i)} \subseteq W'$ be the vertex set covered by $Cr(\boldsymbol{\sigma}'_i)$. In addition, let $\bar{C}r(\boldsymbol{\sigma}'_i)$ denote the set of vertices that are on path $\boldsymbol{\sigma}'_i$ but are not included in $Cr(\boldsymbol{\sigma}'_i)$. The critical vertices $Cr(\boldsymbol{\sigma}'_i)$ cannot be removed because if one of them is removed, all $v \in W'$ are not covered. Therefore, we next select some removal vertices from $\bar{C}r(\boldsymbol{\sigma}'_i)$. Note that we can preliminarily remove the visited vertex covering only the subset of $W'_{Cr(\boldsymbol{\sigma}'_i)}$ from $\bar{C}r(\boldsymbol{\sigma}'_i)$, if such vertices exist, because the visited vertex is determinately redundant. Because it might take a long time to select the optimal removal vertices, we use the following method to quickly find removal vertices.

We compute the saving cost of each vertex in $\bar{C}r(\boldsymbol{\sigma}'_i)$ to decide which vertex to remove. The saving cost of vertex $v_r \in \bar{C}r(\boldsymbol{\sigma}'_i)$ is computed by

$$\Delta c_r = c_{kr} + c_{rh} - c_{kh} \quad \forall v_r \in \bar{C}r(\boldsymbol{\sigma}'_i), \tag{3.3}$$

where v_k and v_h represent the predecessor and successor vertices, respectively, of vertex v_r on path $\boldsymbol{\sigma}'_i$. The vertex that has the largest saving cost is then removed. After we remove a redundant vertex, we connect the predecessor vertex to the successor one of the removed vertex. This is how path $\boldsymbol{\sigma}'_i$ is updated. Then, $Cr(\boldsymbol{\sigma}'_i)$ and $\bar{C}r(\boldsymbol{\sigma}'_i)$ are also updated. This process is iterated until the redundant vertices set, $\bar{C}r(\boldsymbol{\sigma}'_i)$, is empty. The method for removing redundant vertices (RRV) is described as follows.

Algorithm 3.4. Removal of Redundant Vertices (RRV)

```
**Step 0:** Initialize $Cr(\boldsymbol{\sigma}'_i)$ and $\bar{C}r(\boldsymbol{\sigma}'_i)$.
**Step 1:** Compute $\Delta c_r \leftarrow c_{kr} + c_{rh} - c_{kh}$ for all $v_r \in \bar{C}r(\boldsymbol{\sigma}'_i)$, where $v_k$ and $v_h$ are the predecessor and successor vertices of $v_r$, respectively.
**Step 2:** Set $v_{r'} \leftarrow \arg \max_{v_r \in \bar{C}r(\boldsymbol{\sigma}'_i)} \{\Delta c_r\}$.
**Step 3:** Remove $v_{r'}$ from $\boldsymbol{\sigma}'_i$ and update $\boldsymbol{\sigma}'_i$.
**Step 4:** Update $Cr(\boldsymbol{\sigma}'_i)$ and $\bar{C}r(\boldsymbol{\sigma}'_i)$.
**Step 5:** If $\bar{C}r(\boldsymbol{\sigma}'_i) = \emptyset$, exit; otherwise, return to Step 1.
```

Step 1, computes the saving cost for all $v_k \in \bar{C}r(\boldsymbol{\sigma}'_i)$. Step 2 finds the vertex with maximum saving cost and this vertex is removed from $\boldsymbol{\sigma}'_i$ in Step 3. In Steps 4 and 5, $Cr(\boldsymbol{\sigma}'_i)$ and $\bar{C}r(\boldsymbol{\sigma}'_i)$ are updated and if the termination condition is satisfied, we exit; otherwise, we return to Step 1.

This is how HEU (Algorithm 3) finds the forward path from vertex v_i by using RRV (Algorithm 4). Furthermore, we can find an initial tour (solution) of R&R (Algorithm 1) (*i.e.*, Step 0 in R&R) by using HEU, where $V' = V$ and $\tilde{W}' = W$ are set, and vertex v_i and the target vertex are set to the same vertex.

Our R&R (Algorithm 1) finally finds a solution to the CTP by using LAB (Algorithm 2) and HEU (Algorithm 3). We now discuss the size of SPCC (Step 1 in R&R). The performance of the R&R depends heavily on the size of the SPCC. When it is too small, R&R might not perform well. This is because the path (solution) is not likely to improve, even though only a few vertices on the path are switched with others. However, when the size of the SPCC is too large, R&R might also not perform well, because it is too time-consuming to solve the SPCC. Thus, we empirically suggest an appropriate size for the SPCC. We generate an SPCC where the number of vertices is about 100 (*i.e.* $|V'| = 100$). These subproblems are difficult to generate without prior information, because the conditions are different with respect to each problem (instance). So, we conduct few preliminary experiments and determine the appropriate size of the part of the current tour that is temporarily ruined.

4. Computational experiments

We show the comparative results of the computational experiments obtained using our R&R, the 2-stage method [6] and Gendreau's heuristic (GH) [10]. The 2-stage method and the GH are presented in Section 1.1. Our R&R and the GH are implemented in C++ using STL vector. The 2-stage method solves the SCP and the TSP using the MIP solver called CPLEX 12.6 for the SCP [15] and the Chained-Lin–Kernighan heuristic for the TSP [5], respectively. All tests were performed on a 3.7 GHz Core i7-4820k with a Linux operating system (CentOS release 6.6) with 64 GB of RAM memory. Note that none of the implementations used multi-cores.

We randomly generate several types of problems from $|V| = 100$, 1000, 5000, 10000 and $|W| = |V|, 2|V|, ..., 5|V|$ in two models (fixed radius and generalized models). In other words, we generate 4×5 types of problems in each model. In the instances of the fixed radius model, we set a fixed radius d and if the Euclidean distance between a covered vertex $v_l \in W$ and a visited vertex $v_k \in V$ is shorter than d, v_k covers v_l. This fixed radius d is given by

$$d = \max(\max_{v_k \in V} \min_{v_l \in W} \{d_{kl}\}, \max_{v_l \in W} \{d_{k(l),l}\}), \tag{4.1}$$

where $k(l)$ is the index of the vertex of V that is the second closest to v_l [10].

In the instances of the generalized model, we determine some vertices that are covered by each visited vertex as follows. We first set the value $d' = 2 \times d$, where d is given by equation (4.1). Then, v_k covers v_l if the Euclidean distance between the covered vertex $v_l \in W$ and the visited vertex $v_k \in V$ is shorter than d' and $rnd \leq 0.5$, where rnd is a uniform random number $(0 \leq rnd \leq 1)$.

We compare the results obtained using our R&R algorithm with the 2-stage method and the GH with respect to the cost (objective function), tour size (number of vertices) and computation time (seconds). Each computational experiment is performed up to 3600 s. Note that in the 2-stage method, the solvers of the SCP and the TSP both perform up to 3600 s. Also, the termination condition of our R&R is set to be the scenario when the solution does not show any improvement for the 50th time in a row (*i.e.*, R&R is iterated 50 times without improvement in the solution). We show the experimental results with respect to the cost, tour size, and computation time in Tables 1–4, 5–8, and 9–12, respectively of the fixed radius model and Tables 13–16, 17–20, and 21–24, respectively of the generalized model. Note that the costs in the tables are rounded to the nearest integer, even though the costs c_{ij} and d_{kl} are positive real numbers.

Because our R&R and the GH somewhat uses random numbers, we perform 30 independent runs of them for each instance, where the random numbers are generated by using Mersenne Twister [20]. Then, we test the normality of the these distributions using Shapiro–Wilk. In each instance, if normalities of both distributions obtained using R&R and GH are satisfied, t-test is applied to estimate the difference in means of the two distributions; otherwise, the Wilcoxon–Mann–Whitney test is applied to estimate the difference in medians of them. In the t-test, if the null hypothesis of equality of variance is not rejected, the Student's t-test is performed; otherwise, the Welch's t-test is performed. In addition, to find the confidence interval for the mean or median of each distribution obtained using R&R, we use t-test or bootstrapping method with resample size of 10000. For all tests, we assume a confidence interval of 95%.

In all Tables, we show the results of normality tests, where if the normality of the distribution is satisfied, it is denoted by "A"; otherwise, it is denoted by "R".

To compare our R&R with the GH, we indicate the results of the tests of differences in means or medians, where we find the t-value or difference in location (diff. loc.) and p-value. If the t-value (or diff. loc.) is positive and the p-value is less than 0.05, we can conclude that the value obtained using our R&R is smaller than that obtained using the GH. If the t-value (or diff. loc.) is negative and the p-value is less than 0.05, we can conclude that the value obtained using our R&R is larger than that obtained using the GH. Furthermore, in each instance, we show the medians or means; if the normality is satisfied, it is mean; otherwise, it is median. Note that the median of a normal distribution theoretically equals to the mean value. We also compute a gap between the values (mean or median) obtained using R&R and that obtained using the GH if the p-value is less

Table 1. Cost (fixed radius model, $|V| = 100$).

$\lvert W\rvert$	Normality test R&R	Normality test GH	Diff. loc. t-value	p-value	Median (mean) R&R	Median (mean) GH	Gap1	R&R CI	2-stage	Gap2
100	R	R	32.4	5.9e−07	385	419	8.9	[377,389]	388	–
200	R	R	1.6	0.67	347	347	–	[333,362]	329	−5.3
300	R	R	2.9	0.33	308	317	–	[306,311]	302	−2.0
400	R	R	9.5	1.4e−04	353	363	2.9	[350,356]	343	−2.9
500	R	R	2.4	0.84	424	414	–	[412,451]	409	−3.6

Table 2. Cost (fixed radius model, $|V| = 1000$).

$\lvert W\rvert$	Normality test R&R	Normality test GH	Diff. loc. t-value	p-value	Median (mean) R&R	Median (mean) GH	Gap1	R&R CI	2-stage	Gap2
1000	R	R	653.9	3.0e−14	8924	9605	7.6	[8887,8958]	8787	−1.5
2000	R	A	435.1	9.9e−13	9330	9771	4.7	[9253,9380]	9029	−3.2
3000	R	A	320.9	5.4e−12	10385	10683	2.9	[10346,10439]	9867	−5.0
4000	R	A	103.0	1.5e−04	9641	9695	0.6	[9602,9703]	9008	−6.6
5000	A	A	17.9	2.2e−16	9726	10289	5.8	[9706,9746]	9518	−2.1

Table 3. Cost (fixed radius model, $|V| = 5000$).

$\lvert W\rvert$	Normality test R&R	Normality test GH	Diff. loc. t-value	p-value	Median (mean) R&R	Median (mean) GH	Gap1	R&R CI	2-stage	Gap2
5000	A	A	73.1	2.2e−16	17035	18917	11.0	[17017,17054]	16542	−2.9
10000	A	R	1744.6	2.0e−12	21227	22916	8.0	[21214,21240]	20375	−4.0
15000	A	A	79.0	2.2e−16	23292	25332	8.8	[23278,23305]	22708	−2.5
20000	R	A	1671.7	7.4e−15	19815	21495	8.5	[19807,19823]	19527	−1.5
25000	A	R	1434.0	7.3e−13	18992	20480	7.8	[18972,19012]	18572	−2.2

than 0.05. We denote the gap by Gap1, which is computed as follows:

$$\mathrm{Gap1}[\%] = \frac{\mathrm{GH} - \mathrm{R\&R}}{\mathrm{R\&R}} \times 100, \tag{4.2}$$

where GH and R&R represent the values (means or medians) obtained using the GH and R&R, respectively. In the tables, the symbol "–" implies that the p-value is more than 0.05 (*i.e.*, we assume that there is no significant difference between the two values).

Next, we show the confidence intervals (CI) for the value (mean or median) obtained using R&R and the results obtained using the 2-stage method. Note that the result obtained using the 2-stage method is not a distribution but a unique value. We also compute a gap the value obtained using R&R and that obtained using the 2-stage method. We denote the gap by Gap2, which is computed as equation (4.2), where GH is replaced with the value obtained using the 2-stage method. Note that, in each instance, if the value obtained using the 2-stage method is in the CI, the gap is not computed, because we can consider that there is no difference between the two values.

In the Tables of the results with respect to computation time (Tabs. 9–12 and 21–24), the terms "SCP" and "TSP" represent the computation times of the first and second stages, respectively, and the term "#sol." expresses the number of optimal solutions found in the first stage.

Table 4. Cost (fixed radius model, $|V| = 10000$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
10000	A	R	3501.6	4.2e−14	27136	30611	12.8	[27121,27151]	26535	−2.2
20000	A	R	2707.7	2.7e−13	27263	29988	10.0	[27250,27275]	26463	−2.9
30000	A	A	114.6	2.2e−16	29258	31954	9.2	[29246,29269]	28676	−2.0
40000	A	A	88.0	2.2e−16	25691	27890	8.6	[25679,25703]	25598	−0.4
50000	A	A	72.3	2.2e−16	27094	29134	7.5	[27079,27110]	26149	−3.5

Table 5. Size (fixed radius model, $|V| = 100$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
100	R	R	−1.0	0.38	20	19	–	[19,21]	17	−15.0
200	R	R	0.5	0.72	14	15	–	[14,14]	13	−7.1
300	R	R	−1.0	0.45	14	13	–	[13,15]	11	−21.4
400	R	R	−1.0	0.45	17	16	–	[16,18]	13	−23.5
500	R	R	−1.0	0.45	22	21	–	[21,23]	17	−22.7

Table 6. Size (fixed radius model, $|V| = 1000$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
1000	R	A	0.5	0.77	106	108	–	[105,106]	81	−23.2
2000	R	R	3.3	0.03	110	114	3.6	[109,113]	85	−22.7
3000	R	A	1.4	0.35	136	140	–	[136,136]	105	−22.8
4000	R	A	0.0	1.00	116	117	–	[115,117]	84	−27.6
5000	R	A	2.0	0.46	127	129	–	[127,128]	94	−26.0

Table 7. Size (fixed radius model, $|V| = 5000$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
5000	R	A	21.5	2.7e−05	412	430	4.5	[409,414]	311	−24.4
10000	R	R	2.0	0.69	625	628	–	[624,625]	475	−23.9
15000	R	A	21.0	5.8e−06	732	753	2.9	[731,733]	582	−20.5
20000	R	A	2.7	0.55	547	549	–	[545,549]	440	−19.6
25000	R	R	−9.7	0.01	512	502	−2.0	[511,512]	406	−20.6

In Tables 1–12, we first observe the results of the fixed radius model with respect to cost, size, and time. We mainly look at Gap1 and Gap2 because these data summarize the information of performances of the three algorithms.

Cost: There are no significant differences between the results obtained using R&R and GH in three of small-scale five instances (Tab. 1). In the other instances, our R&R outperforms the GH because Gap1 are between 0.6% and 12.8% (*i.e.*, positive values). We believe R&R finds near-optimal solutions in small-scale instances because the GH can also find near-optimal solutions in the instances generated by $|V| = 100$;

TABLE 8. Size (fixed radius model, $|V| = 10000$).

$\lvert W\rvert$	Normality test		Diff. loc.		Median (mean)			R&R		
	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
10000	R	R	43.5	3.7e−05	1054	1100	4.4	[1053,1055]	813	−22.9
20000	R	A	15.0	1.7e−03	1045	1065	1.9	[1044,1046]	847	−18.9
30000	R	R	22.0	1.4e−03	1190	1213	1.9	[1188,1192]	982	−17.5
40000	R	A	9.5	0.05	924	933	1.0	[922,925]	800	−13.4
50000	R	R	−5.0	0.41	1019	1007	–	[1017,1021]	838	−17.8

TABLE 9. Computation time (fixed radius model, $|V| = 100$).

$\lvert W\rvert$	Normality test		Diff. loc.		Median (mean)			R&R	2-stage			
	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	SCP	TPS	#sol.	Gap2
100	A	R	−8.3	3.7e−04	8.58	0.01	−99.9	[8,10]	5.30	50.44	4920	549.4
200	R	R	−37.1	4.3e−05	37.13	0.03	−99.9	[27,45]	6.07	56.05	5692	67.3
300	A	R	−56.0	3.7e−04	61.11	0.03	−100.0	[54,68]	9.57	87.65	6664	59.1
400	R	R	−61.4	3.7e−04	61.44	0.04	−99.9	[53,70]	25.49	289.59	21252	412.9
500	R	R	−42.0	3.7e−04	42.02	0.05	−99.9	[35,49]	31.26	0.59	44	−24.2

TABLE 10. Computation time (fixed radius model, $|V| = 1000$).

$\lvert W\rvert$	Normality test		Diff. loc.		Median (mean)			R&R	2-stage			
	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	SCP	TPS	#sol.	Gap2
1000	R	A	−246.2	2.0e−12	247.03	0.88	−99.6	[194,315]	3600	0.04	2	1357.3
2000	R	A	−226.6	2.7e−13	228.02	1.45	−99.4	[170,283]	3600	0.01	1	1478.8
3000	A	A	−10.9	9.0e−12	294.03	2.22	−99.2	[239,349]	3600	0.01	1	1124.3
4000	A	A	−11.7	1.6e−12	619.61	2.00	−99.7	[512,727]	3600	0.01	1	481.0
5000	R	A	−397.2	1.1e−13	399.57	2.36	−99.4	[280,500]	3600	0.03	1	801.0

TABLE 11. Computation time (fixed radius model, $|V| = 5000$).

$\lvert W\rvert$	Normality test		Diff. loc.		Median (mean)			R&R	2-stage			
	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	SCP	TPS	#sol.	Gap2
5000	R	A	−349.3	2.2e−16	385.76	35.90	−90.7	[290,473]	3600	0.23	2	833.3
10000	R	A	−87.5	3.1e−13	173.14	85.87	−50.4	[122,215]	3600	0.23	1	1979.4
15000	R	A	−28.4	0.29	152.96	124.85	–	[98,203]	3600	0.61	2	2254.0
20000	R	A	−287.9	2.2e−16	380.66	91.15	−76.1	[265,488]	3600	0.20	1	845.8
25000	R	A	−562.7	2.2e−16	651.55	92.88	−85.7	[378,898]	3600	0.17	1	452.6

their report shows that the gaps between the costs obtained using the GH and optimal solutions are less than 4% [10]. Also, the 2-stage method is superior to our R&R in almost all the instances. However, the gaps (Gap2) are larger than −6.6%; thus we perceive that the differences between the performances of the R&R and the 2-stage method are small.

Size: There is not much of a difference between the sizes of tours obtained using R&R and the GH. The sizes of tours obtained using the 2-stage method are the smallest of the three in all instances. In the fixed radius model, the smaller sizes of tours are likely to lead to better solutions. Thus, the 2-stage method finds good solutions. In larger-scale instances, the GH is inferior to our R&R with respect to cost, even though the

Table 12. Computation time (fixed radius model, $|V| = 10000$).

	Normality test		Diff. loc.		Median (mean)			R&R	2-stage			
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	SCP	TPS	#sol.	Gap2
10000	R	A	156.3	2.8e−05	143.49	307.22	114.1	[64,183]	3600	0.38	1	2409.2
20000	R	A	29.5	0.43	377.71	409.33	–	[265,479]	3600	0.38	1	853.2
30000	R	A	226.2	8.7e−06	346.78	575.73	66.0	[268,409]	3600	0.48	1	938.3
40000	R	A	−316.4	7.6e−16	774.60	455.61	−41.2	[550,967]	3600	0.41	1	364.8
50000	R	A	−400.9	9.4e−06	914.72	510.51	−44.2	[724,1101]	3600	0.44	1	293.6

Table 13. Cost (generalized model, $|V| = 100$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
100	R	A	32.5	4.3e−07	281	315	11.9	[278,285]	299	6.3
200	R	R	−10.6	1.8e−03	208	186	−10.6	[203,216]	297	42.6
300	R	A	22.5	1.0e−04	237	260	9.6	[237,247]	270	13.7
400	R	R	28.9	3.0e−09	233	259	10.7	[226,241]	278	19.1
500	R	R	57.4	2.5e−13	292	352	20.5	[286,297]	334	14.5

Table 14. Cost (generalized model, $|V| = 1000$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
1000	A	R	898.3	1.1e−13	5268	6188	17.5	[5215,5322]	6138	16.5
2000	A	A	21.6	2.2e−16	5501	6483	17.9	[5443,5559]	6886	25.2
3000	A	R	1055.6	2.0e−12	6476	7479	15.5	[6422,6530]	7664	18.3
4000	R	A	938.9	1.7e−13	5863	6879	17.3	[5659,6045]	6981	19.1
5000	A	A	22.8	2.2e−16	6555	7475	14.0	[6488,6621]	7639	16.5

Table 15. Cost (generalized model, $|V| = 5000$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
5000	A	R	2728.4	1.7e−14	10283	13011	26.5	[10181,10385]	12511	21.7
10000	A	R	3511.1	2.0e−12	12980	16573	27.7	[12874,13086]	15496	19.4
15000	R	A	3658.0	5.7e−15	14622	18254	24.8	[14577,14696]	16987	16.2
20000	A	R	3416.5	6.5e−16	12329	15740	27.7	[12260,12397]	14898	20.8
25000	A	A	50.4	2.2e−16	12309	15076	22.5	[12230,12388]	14533	18.1

sizes of tours obtained using the two heuristics are not very different. We believe that this is due to the substandard performance of GENIUS (in the GH) for large-scale problems.

Time: In small-scale instances (Tab. 9), the GH is faster than the R&R and 2-stage method. In the 2-stage method, the computation times of TSPs become larger when the numbers of optimal solutions to SCP (#sol) increase. In larger-scale instances (Tabs. 10–12), the 2-stage method is extremely slow and R&R and GH gradually slow with expanding scales of instances. The GH is faster than the other two algorithms.

TABLE 16. Cost (generalized model, $|V| = 10000$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
10000	A	R	5032.7	1.1e−13	16667	21745	30.5	[16547,16786]	19076	14.5
20000	A	R	4859.5	4.2e−14	16615	21414	28.9	[16507,16722]	20058	20.7
30000	R	R	4778.2	4.2e−14	18722	23468	25.3	[18625,18831]	21819	16.5
40000	A	A	75.8	2.2e−16	16277	20513	26.0	[16228,16326]	19274	18.4
50000	R	A	4855.6	4.2e−14	17106	21927	28.2	[17062,17144]	21182	23.8

TABLE 17. Size (generalized model, $|V| = 100$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
100	R	A	−1.5	0.30	18	17	–	[17,19]	12	−33.3
200	R	R	−3.0	0.09	19	17	–	[19,19]	12	−36.8
300	R	R	−0.5	0.71	18	17	–	[16,19]	12	−31.4
400	R	R	0.0	1	19	19	–	[18,19]	14	−24.3
500	R	R	0.0	1	24	24	–	[23,25]	17	−27.7

TABLE 18. Size (generalized model, $|V| = 1000$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
1000	A	A	2.9	0.01	87	89	2.1	[86,88]	58	−33.6
2000	A	A	−10.2	1.4e−14	107	102	−5.1	[107,108]	69	−35.8
3000	A	R	−7.0	2.8e−03	129	125	−3.2	[127,130]	92	−28.5
4000	A	R	0.0	0.86	114	115	–	[113,115]	80	−29.7
5000	A	R	−4.0	0.04	132	126	−4.8	[131,133]	94	−29.0

TABLE 19. Size (generalized model, $|V| = 5000$).

	Normality test		Diff. loc.		Median (mean)			R&R		
$\|W\|$	R&R	GH	t-value	p-value	R&R	GH	Gap1	CI	2-stage	Gap2
5000	A	A	−5.6	1.2e−06	363	352	−2.8	[359,366]	245	−32.4
10000	A	R	−13.0	4.0e−04	540	529	−2.1	[537,542]	389	−27.9
15000	A	A	9.8	6.1e−14	634	651	2.7	[632,637]	474	−25.3
20000	A	A	−4.1	1.4e−04	526	519	−1.3	[523,528]	380	−27.7
25000	A	A	−2.1	0.04	501	498	−0.7	[499,504]	366	−27.0

Next, we observe the results of the generalized radius model with respect to cost, size, and time in Tables 13–24.

Cost: In small-scale instances (Tab. 13), the costs obtained using R&R are lower than that obtained using the GH except for the second instance, and the 2-stage method is inferior to R&R. In larger-scale instances (Tabs. 14–16), the costs obtained using R&R are lowest. Gap1 increases with scales of instances, whereas Gap2 depends little on scales of instances. The gaps (Gap1 and Gap2) are large, up to 30.5%; thus, our R&R outperforms the other two algorithms.

Table 20. Size (generalized model, $|V| = 10000$).

$\|W\|$	Normality test R&R	Normality test GH	Diff. loc. t-value	p-value	Median (mean) R&R	Median (mean) GH	Gap1	R&R CI	2-stage	Gap2
10000	A	A	1.3	0.20	864	868	–	[858,870]	614	−28.9
20000	A	A	−7.7	2.3e−10	927	907	−2.2	[923,931]	652	−29.7
30000	R	R	9.7	0.09	1054	1071	–	[1048,1059]	797	−24.3
40000	A	R	−8.0	0.02	902	897	−0.6	[900,905]	641	−29.0
50000	R	R	−2.0	0.53	995	997	–	[993,996]	748	−24.8

Table 21. Computation time (generalized model, $|V| = 100$).

$\|W\|$	Normality test R&R	Normality test GH	Diff. loc. t-value	p-value	Median (mean) R&R	Median (mean) GH	Gap1	R&R CI	2-stage SCP	2-stage TPS	2-stage #sol.	Gap2
100	R	R	−15.6	4.0e−03	15.67	0.04	−99.8	[12,19]	3.56	0.06	2	−76.9
200	A	R	−6.6	3.7e−04	6.62	0.04	−99.4	[5,8]	7.84	0.06	2	19.4
300	A	R	−18.1	3.7e−04	19.89	0.05	−99.7	[16,23]	4.40	0.15	9	−77.1
400	A	R	−11.4	3.7e−04	13.25	0.07	−99.5	[11,16]	17.30	0.12	2	31.5
500	R	R	−32.5	4.3e−05	32.60	0.11	−99.7	[29,37]	2.59	0.27	16	−91.2

Table 22. Computation time (generalized model, $|V| = 1000$).

$\|W\|$	Normality test R&R	Normality test GH	Diff. loc. t-value	p-value	Median (mean) R&R	Median (mean) GH	Gap1	R&R CI	2-stage SCP	2-stage TPS	2-stage #sol.	Gap2
1000	A	R	−1072.5	2.0e−12	1025.03	1.16	−99.9	[917,1133]	3600	0.01	1	251.2
2000	A	A	−14.4	9.7e−15	367.83	2.22	−99.4	[316,420]	3600	0.01	1	878.7
3000	A	A	−11.3	4.2e−12	389.87	3.01	−99.2	[320,460]	3600	0.03	1	823.4
4000	A	A	−9.5	2.1e−10	313.66	3.38	−98.9	[247,381]	3600	0.03	1	1047.8
5000	A	A	−14.3	1.1e−14	313.71	4.22	−98.7	[270,358]	3600	0.01	1	1047.6

Table 23. Computation time (generalized model, $|V| = 5000$).

$\|W\|$	Normality test R&R	Normality test GH	Diff. loc. t-value	p-value	Median (mean) R&R	Median (mean) GH	Gap1	R&R CI	2-stage SCP	2-stage TPS	2-stage #sol.	Gap2
5000	A	A	−12.8	1.8e−13	2359.20	48.71	−97.9	[1990,2728]	3600	0.36	3	52.6
10000	A	A	−7.4	3.3e−08	702.75	107.38	−84.7	[539,866]	3600	0.20	1	412.3
15000	R	A	−166.4	9.7e−04	325.83	158.78	−51.3	[173,485]	3600	0.23	1	1005.0
20000	R	A	−454.4	1.7e−12	613.87	159.43	−74.0	[342,928]	3600	0.14	1	486.5
25000	R	R	−734.4	3.9e−13	891.95	154.45	−82.7	[657,1263]	3600	0.17	1	303.6

Size: There is not much of a difference between the sizes of tours obtained using R&R and the GH. The sizes of tours obtained using the 2-stage method is the smallest of the three in all instances. In the generalized model, because the small size of tour does not always lead to a good solution (see Fig. 2), the 2-stage method cannot always find good solutions.

Time: In small-scale instances (Tab. 21), the GH is faster than the other heuristics. The 2-stage method in the generalize model is faster than in the fixed radius model because of the smaller numbers of optimal solutions to SCP (#sol.) in the generalize model. In larger-scale instances (Tabs. 22–24), the performance

TABLE 24. Computation time (generalized model, $|V| = 10000$).

$\|W\|$	Normality test R&R	Normality test GH	Diff. loc. t-value	p-value	Median (mean) R&R	Median (mean) GH	Gap1	R&R CI	2-stage SCP	2-stage TPS	2-stage #sol.	Gap2
10000	A	A	−10.7	1.3e−11	2326.16	376.45	−83.8	[1954,2698]	3600	1.12	3	54.8
20000	A	A	−8.8	1.0e−09	1872.76	575.73	−69.3	[1572,2173]	3600	0.38	1	92.2
30000	R	A	127.6	0.33	661.65	794.56	–	[252,1022]	3600	0.47	1	444.2
40000	A	A	−4.8	3.7e−05	1377.90	759.26	−44.9	[1118,1638]	3600	0.36	1	161.3
50000	R	A	112.0	0.30	753.17	866.64	–	[504,986]	3600	0.38	1	378.0

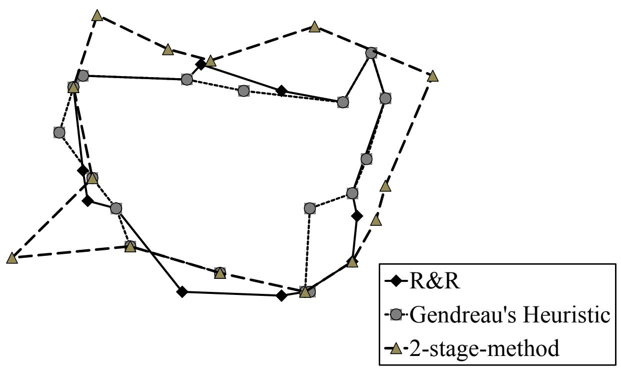

FIGURE 4. Three tours.

of the three heuristics gradually slows down with expanding scales of problems. In particular, the 2-stage method is extremely slow. The GH is faster than the other heuristics in almost all instances.

In the paragraphs below, we qualitatively summarize the performances of the three algorithms with respect to cost and (computation) time.

R&R: The performance is good in most instances. In particular, they are much better than the other heuristics in the generalized model.

2-Stage method: In the fixed radius model, the performance with respect to cost is slightly better than that of R&R. However, in the generalized model, it is much inferior to R&R. The computation times for the SCP explode when the scale of the problem expands, whereas those of the TSP are very fast in spite of the increase in scale of the problems

Gendreau's heuristic (GH): In small-scale instances, the performance is very good. However, it deteriorates quite a lot for larger-scale instances. We believe that this is because the GENIUS heuristic is unlikely to find a good feasible solution to large-scale TSP.

We show an example of the tours in Figure 4. We describe three tours (solutions) obtained using our R&R, the 2-stage method and the GH in the generalized model generated by $|V| = 100$ and $|W| = 400$.

The 2-stage method traces a roundabout route, and our R&R and the GH trace smaller tours than the 2-stage method, even when the sizes of tours are larger. (The sizes of tours obtained using the R&R, GH, and 2-stage method, are 18, 19, and 14, respectively. Note that the sizes are not equal to those listed in Table 17 because Figure 4 shows the result from only one in 30 runs of $|V| = 100$ and $|W| = 400$). This occurrence is shown in Figure 2.

5. Concluding remarks

In this study, we proposed an algorithm for the large-scale CTP. Our algorithm is based on the ruin and recreate heuristic (R&R). This ruin and recreate heuristic first generates a subproblem and solves the subproblem by using a labeling algorithm. In the labeling algorithm, we use the cost of a feasible path as a label on a vertex instead of the cost from the source to that vertex. We also proposed a heuristic for finding the feasible path. Computational experiments revealed that our R&R performs as well as the existing methods for the fixed radius model scenario. On the other hand, for the case of the generalized model, our R&R algorithm outperforms the existing methods and performs well even for large-scale instances.

References

[1] C.E. Andrade, F.K. Miyazawa and M.G. Resende, Evolutionary algorithm for the k-interconnected multi-depot multi-traveling salesmen problem, in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. ACM (2013) 463–470.

[2] E. Balas and A. Ho, Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study. *Math. Prog.* **12** (1980) 37–60.

[3] R. Baldacci, M.A. Boschetti, V. Maniezzo and M. Zamboni, Scatter Search Methods for the Covering Tour Problem. Vol. 30 of *Metaheuristic Optimization via Memory and Evolution* (2005) 59–91.

[4] V. Chvata, A greedy heuristic for the set-covering problem. *Math. Oper. Res.* **4** (1979) 233–235.

[5] Concorde TSP solver. Available at: http://www.math.uwaterloo.ca/tsp/concorde.html (2016).

[6] J.R. Current and D.A. Schilling, The covering salesman problem. *Transp. Sci.* **23** (1989) 208–213.

[7] K.F. Doerner and R.F. Hartl, Health care logistics, emergency preparedness, and disaster relief: new challenges for routing problems with a focus on the Austrian situation, in The Vehicle Routing Problem: Latest Advances and New Challenges (2008) 527–550.

[8] R.Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia and M. Goh, Covering problems in facility location: a review. *Comput. Ind. Eng.* **62** (2012) 368–407.

[9] M. Gendreau, A. Hertz and G. Laperte, New insertion and postoptimization procedures for the traveling salesman problem. *Oper. Res.* **40** (1992) 1086–1094.

[10] M. Gendreau, G. Laporte and F. Semet, The covering tour problem. *Oper. Res.* **45** (1997) 568–576.

[11] B. Golden, Z. Naji-Azimi, S. Raghavan, M. Salari and P. Toth, The generalized covering salesman problem. *INFORMS J. Comput.* **24** (2012) 534–553.

[12] M.H.N. Ha, N. Bostel, A. Langevin and L.M. Rosseau, An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *Eur. J. Oper. Res.* **226** (2013) 211–220.

[13] M. Hachicha, M.J. Hodgson, G. Laporte and F. Semet, Heuristics for the multi-vehicle covering tour problem. *Comput. Oper. Res.* **27** (2000) 29–42.

[14] M.J. Hodgson, G. Laporte and F. Semet, A covering tour model for planning mobile health care facilities in Suhum district, Ghana. *J. Reg. Sci.* **38** (1998) 621–628.

[15] ILOG: CPLEX. Available at: http://www.ilog.com/ (2018).

[16] N. Jozefowiez, F. Semet and E. Talbi, The bi-objective covering tour problem. *Comput. Oper. Res.* **34** (2007) 1929–1942.

[17] M. Labbe and G. Laporte, Maximizing user convenience and postal service efficiency in post box location. *Belgian J. Oper. Res. Statt. Comput. Sci.* **26** (1986) 21–35.

[18] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.* **21** (1973) 498–516.

[19] E. Luis, I.S. Dolinskaya and K.R. Smilowitz, Disaster relief routing: integrating research and practice. *Socio-econ. Plan. Sci.* **46** (2012) 88–97.

[20] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.* **8** (1998) 3–30.

[21] L. Motta, L.S. Ochi and C. Martinhon, Grasp metaheuristics for the generalized covering tour problem, Vol. 1 of *Proceedings of IV Metaheuristic International Conference, Porto, Portugal* (2001) 387–93.

[22] G. Nagy and S. Salhi, Location-routing: issues, models and methods. *Eur. J. Oper. Res.* **177** (2007) 649–72.

[23] Z. Naji-Azimi, J. Renaud, A. Ruiz and M. Salari, A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. *Eur. J. Oper. Res.* **222** (2012) 596–605.

[24] C. Prodhon and C. Prins, A survey of recent research on location-routing problems. *Eur. J. Oper. Res.* **238** (2014) 1–17.

[25] S. Rath and W.J. Gutjahr, A math-heuristic for the warehouse location-routing problem in disaster relief. *Comput. Oper. Res.* **42** (2014) 25–39.

[26] I. Rodriguez-Martin, J.J. Salazar-Gonzalez and H. Yaman, A branch-and-cut algorithm for the hub location and routing problem. *Comput. Oper. Res.* **50** (2014) 161–174.

[27] M. Salari and Z. Naji-Azimi, An integer programming-based local search for the covering salesman problem. *Comput. Oper. Res.* **39** (2012) 2594–2602.

[28] M. Salari, M. Reihaneh and M.S. Sabbagh, Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. *Comput. Ind. Eng.* **83** (2015) 244–251.

[29] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt and G. Dueck, Record breaking optimization results using the ruin and recreate principle. *J. Comput. Phys.* **159** (2000) 139–171.