The Multi-Vehicle Probabilistic Covering Tour Problem

İsmail Karaoğlan[a] Güneş Erdoğan[b,*] Çağrı Koç[c]

[a]*Department of Industrial Engineering, Selçuk University, Konya, Turkey,*
e-mail: ikaraoglan@selcuk.edu.tr

[*]*Corresponding author*
[b]*School of Management, University of Bath, Bath, United Kingdom*
e-mail: g.erdogan@bath.ac.uk

[c]*Department of Business Administration, Social Sciences University of Ankara, Ankara, Turkey*
e-mail: cagri.koc@asbu.edu.tr

Abstract

This paper introduces the Multi-Vehicle Probabilistic Covering Tour Problem (MVPCTP) which extends the Covering Tour Problem (CTP) by incorporating multiple vehicles and probabilistic coverage. As in the CTP, total demand of customers is attracted to the visited facility vertices within the coverage range. The objective function is to maximize the expected customer demand covered. The MVPCTP is first formulated as an integer non-linear programming problem, and then a linearization is proposed, which is strengthened by several sets of valid inequalities. An effective branch-and-cut algorithm is developed in addition to a local search heuristic based on Variable Neighborhood Search to obtain upper bounds. Extensive computational experiments are performed on new benchmark instances adapted from the literature.

Keywords: Vehicle routing; covering tour; linearization; integer programming; branch-and-cut.

1 Introduction

In this paper, we introduce the *Multi-Vehicle Probabilistic Covering Tour Problem* (MVPCTP), in which a fleet of vehicles with a limited range leave the depot and visit a subset of the *facility* vertices to service the set of *customer* vertices, and a vehicle visiting facility vertex i can successfully serve customer vertex j with probability p_{ij}. The MVPCTP is a generalization of the well known Covering Tour Problem (CTP) introduced by Gendreau et al. (1997), for which it is assumed that when a customer vertex is within pre-specified distance of a visited facility vertex, it is covered.

The MVPCTP has been motivated by the recent advances in the remote sensing and Unmanned Aerial Vehicle (UAV) technologies and the widespread applications of their joint use. Diverse

types of sensors can be fitted on UAVs, ranging from thermal cameras to infrared spectroscopy instruments and gas (or smoke) detectors. Beyond the obvious military and security applications, UAVs with remote sensing capabilities can be used in a large number of applications including but not limited to agriculture and forestry (crop health monitoring, fire detection), disaster management (monitoring ground displacement and floods), ecosystems management (mapping and monitoring species), and map making (digital elevation models and 3D mapping). We refer the interested reader to the comprehensive survey by Pajares (2015).

In addition, a wide variety of applications with a covering objective arise in healthcare and humanitarian logistics, such as the modeling of emergency response for earthquake, flood and tsunami (see Altay and Green, 2006; Caunhye et al., 2012; De La Torre et al., 2012). Most of the models that incorporate a covering objective with a step function for coverage may be upgraded into probabilistic coverage models using our approach. Before we proceed with our study, we briefly review the related problems and their variants.

1.1 A brief review of the literature

The Vehicle Routing Problem (VRP) and its variants have been intensively studied in the literature. Many algorithms, both exact and heuristics, have been developed for the VRP and its variations over the past fifty years. We refer the interested reader to the surveys by Cordeau et al. (2007) and Laporte (2009), and the books by Golden et al. (2008) and Toth and Vigo (2014). In recent years, there has been an increase in the study of stochastic variants of the VRP, and the amount of data enabling the understanding of various stochastic phenomena. For further details, we refer the reader to the recent survey of Gendreau et al. (2016).

Gendreau et al. (1997) introduced the CTP, proposed a branch-and-cut algorithm and a constructive heuristic algorithm. Jozefowiez et al. (2007) studied the bi-objective CTP where the objectives are to minimize the tour length and the cover. The authors developed a multi-objective evolutionary algorithm and a branch-and-cut algorithm. Vargas et al. (2015) proposed an adaptive large neighborhood search metaheuristic to solve the CTP, which is based on a selector operator that allows to convert a giant tour into an optimal CTP solution.

Hachicha et al. (2000) introduced the multi-vehicle covering tour problem (m-CTP) with a real-life application which considers the routing of mobile health care delivery teams. A mathematical formulation and three heuristic algorithms are proposed. For the same problem, a branch-and-price algorithm is developed by Lopes et al. (2013). A variant of the m-CTP is studied by Hà et al. (2013) where only the restriction on the number of vertices is considered, i.e., the constraint on the route length is relaxed. The authors developed a branch-and-cut algorithm as well as several metaheuristics. Jozefowiez (2014) presented a branch-and-price algorithm for the m-CTP and provided computational results for randomly generated instances with up to 100 facility nodes and 150 customer nodes. Another m-CTP variant, which considers multiple depots, is introduced by Allahyari et al. (2015). The authors developed two integer programming formulations, flow-based and node-based, and a hybrid metaheuristic which combines GRASP, iterated local search and simulated annealing. Flores-Garza et al. (2017) introduced the cumulative m-CTP, which aims to find a set of tours that must be followed by a fleet of vehicles in order to minimize the sum of arrival times at each visited location. The authors proposed a mixed integer linear programming formulation and a greedy randomized adaptive search procedure (GRASP). Recently, Pham et al. (2017) have introduced a variant named Multi-Vehicle Multi-Covering Tour Problem (mm-CTP), in which the customer vertices must be covered several times rather than once.

As an application of m-CTP in humanitarian logistics, Nolz et al. (2010) introduced the multi-objective m-CTP for water distribution in an area affected by a disaster. The goals are to minimize total traveled distance by the covered customers to reach their nearest visited vertex, the number of unreached customers within a prespecified maximum distance, the total tour length, and the latest arrival time at a population center point. A generalized version of the m-CTP is studied by Naji-Azimi et al. (2012) in which the humanitarian aid to the affected people is supplied by several satellite distribution centers located within a predefined distance from their domiciles.

As a final note, we would like to emphasize that deterministic and probabilistic coverage have been widely studied within the facility location literature, based on the seminal works of Church and ReVelle (1974) and Daskin (1983). We refer the interested reader to the recent survey by Ahmadi-Javid et al. (2017).

1.2 Scientific contributions and structure of the paper

To the best of our knowledge, the MVPCTP has not been studied in the literature. The main contributions of this paper are as follows. We first introduce the MVPCTP as a new CTP variant. We develop a mathematical formulation, devise a linearization scheme, and adapt a set of valid inequalities to strengthen the linear programming relaxation of the formulation. We then develop a branch-and-cut algorithm which is able to solve realistic size instances of the MVPCTP to optimality within limited computing times. We finally propose a variable neighborhood search (VNS) heuristic to improve the initial solution and lower bounds found during the search process of the solution approach.

The rest of this paper is organized as follows. Section 2 presents the problem definition, formulations, linearization scheme, and valid inequalities. Section 3 describes the branch-and-cut algorithm. Section 4 provides detailed results of our computational experiments. Finally, the conclusions are presented in Section 5.

2 Problem Definition, Formulation, Linearization and Valid Inequalities

We are given a directed graph $G = (V \cup W, A)$ where V is the set of facility vertices and W is the set of customer vertices. We define $q_j > 0$ as the demand associated with vertex $j \in W$. The arc set A consists of all possible connections between the facility vertices. Furthermore, associated with each arc $(i, j) \in A$, there is a distance d_{ij}. A fleet of vehicles, denoted K, will leave the depot (with index 0), visit a subset of V and return to the depot, each traveling a maximum distance of L. Each facility vertex in V can be visited at most once. If a vehicle visits $i \in V$, then it serves $j \in W$ with probability $0 \leq p_{ij} < 1$. Let us define y_i as a binary variable that assumes the value of 1 if facility vertex i is visited by a vehicle. Given the tours performed by the vehicles, the probability that customer vertex j has been served is then $1 - \prod_{i \in V} (1 - p_{ij})^{y_i}$. Let us also define variable x_{ij} that assumes the value of 1 if arc $(i, j) \in A$ is traversed, and 0 otherwise. Finally, let t_i be the cumulative distance traversed by a vehicle when it arrives at vertex $i \in V$.

Given a homogeneous fleet of vehicles, the MVPCTP aims to determine a set of distance-constrained

routes to visit a subset of facilities to maximize the expected demand covered, where a vehicle visiting facility vertex i covers the demand q_j of customer vertex j with probability $p_{ij} \in [0, 1)$. It is straightforward to show that MVPCTP is NP-Hard by a reduction from the TSP. Given an instance of the TSP, construct an instance of the MVPCTP with the same vertex set as the facility vertices, a single customer with a demand of 1, and a single vehicle. Set any facility vertex as the depot, and define the coverage probability $p_{ij} = p^* \in [0, 1)$ for all facility-customer pairs. The TSP instance has a feasible solution with the objective value z^* if and only if the MVPCTP instance has a feasible solution for the distance limit z^* for the vehicle, and an objective function value of $1-(1-p^*)^{|V|-1}$.

2.1 Formulation

The MVPCTP can be formulated as (MVPCTP1):

$$\text{Maximize} \sum_{j \in W} q_j \left(1 - \prod_{i \in V} (1 - p_{ij})^{y_i}\right) \tag{1}$$

Subject to

$$\sum_{i \in V} x_{ij} = y_j \qquad j \in V \backslash \{0\} \tag{2}$$

$$\sum_{j:(i,j) \in A} x_{ij} = \sum_{j:(j,i) \in A} x_{ji} \qquad i \in V \tag{3}$$

$$\sum_{j:(0,j) \in A} x_{0j} = |K| \tag{4}$$

$$t_i - t_j + (L - d_{i0} + d_{ij} - d_{0j}) x_{ij} + (L - d_{i0} + d_{ji} - d_{0j}) x_{ji} \leq (L - d_{i0}) y_i - d_{0j} y_j \qquad i, j \in V \backslash \{0\}, i \neq j \tag{5}$$

$$t_i \leq (L - d_{i0}) y_i - (L - d_{i0} - d_{0i}) x_{0i} \qquad i \in V \backslash \{0\} \tag{6}$$

$$t_i \geq d_{0i} y_i \qquad i \in V \backslash \{0\} \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j) \in A \tag{8}$$

$$y_i \in \{0, 1\} \qquad i \in V \backslash \{0\} \tag{9}$$

$$t_i \geq 0 \qquad i \in V \backslash \{0\}. \tag{10}$$

The objective function (1) maximizes the expected customer demand served. Constraints (2) and (3) are degree and assignment constraints, respectively. Constraints (4) ensure that exactly $|K|$ vehicles depart from the depot. Constraints (5)–(7) impose the subtour elimination and distance constraints, which are the adapted and strengthened version of Miller-Tucker-Zemlin constraints (Miller et al., 1960). Constraints (8)–(9) and (10) are the integrality and nonnegativity constraints, respectively.

2.2 Linearization

In this section, we provide a linearization scheme for the nonlinear model MVPCTP1. This is particularly important for a larger computational reach, since the state-of-the-art nonlinear solvers can handle a significantly smaller number of variables and constraints with respect to linear solvers, and do not allow dynamic addition of valid inequalities. To this end, we first analyze the objective function.

Proposition 1: Objective function (1) is concave.

Proof. It suffices to show that every member of the summation is a concave function. For a given $j \in W$, let us denote the probability that customer j is covered as $f_j(y) = 1 - \prod_{i \in V} (1 - p_{ij})^{y_i}$. Then the derivative of $f_j(y)$ with respect to y_k is:

$$\frac{\delta f}{\delta y_k} = -\ln(1 - p_{kj}) \prod_{i \in V} (1 - p_{ij})^{y_i}. \tag{11}$$

As a result, the Hessian H of $f_j(y)$ is given by the elements:

$$\frac{\delta f}{\delta y_k \delta y_l} = -\ln(1 - p_{kj}) \ln(1 - p_{lj}) \prod_{i \in V} (1 - p_{ij})^{y_i}. \tag{12}$$

Finally, we compute

$$a^T H a = -\prod_{i \in V} (1 - p_{ij})^{y_i} \sum_{k,l} \ln(1 - p_{kj}) \ln(1 - p_{lj}) a_k a_l = -\prod_{i \in V} (1 - p_{ij})^{y_i} \left(\sum_{k} \ln(1 - p_{kj}) a_k \right)^2 \leq 0. \tag{13}$$

The inequality follows from $-\prod_{i\in V}(1-p_{ij})^{y_i} < 0$ and $(\sum_k \ln(1-p_{kj})a_k)^2 \geq 0$, proves that the Hessian is negative semidefinite and $f_j(y)$ is concave. □

Based on this result, we can use the tangents of $f_j(y)$ to linearize the objective function (1). Let us define w_j as the auxiliary variable to linearize $f_j(y)$. Then, the linearized model is (MVPCTP2):

$$\text{Maximize} \sum_{j\in W} q_j w_j \tag{14}$$

Subject to

(2) – (10),

$$w_j \leq \sum_{k\in V} -\ln(1-p_{kj}) \prod_{i\in V}(1-p_{ij})^{y_i^*} y_k + 1 - \prod_{i\in V}(1-p_{ij})^{y_i^*} + \sum_{k\in V} \ln(1-p_{kj}) \prod_{i\in V}(1-p_{ij})^{y_i^*} y_k^* \qquad y^* \in [0,1]^{|V|-1} \tag{15}$$

$$0 \leq w_j \leq 1 \qquad j \in W. \tag{16}$$

We emphasize that there are an infinite number of elements in the constraint set (15), each of which is stated based on a fixed solution $y_i^* \in [0,1]\, \forall i \in V \setminus \{0\}$. The upper bound imposed on w_j is strictly equal to the value of $f_j(y^*)$ for this particular realization of y^*, and is valid for all other realizations of y^* due to the concavity of $f_j(y)$. Violated members of (15) for a given solution can be identified in $O(|V| \times |W|)$ time by simply plugging in the values of w, y, and $y^* = y$.

2.3 Valid Inequalities

We make use of several valid inequalities which are proposed and used in the literature. It can be shown that these inequalities are also valid for the MVPCTP.

The first inequality is used for the generalized subtour elimination constraints by Dang et al. (2013):

$$\sum_{u,v\in S} x_{uv} \leq \sum_{i\in S} y_i - y_j \qquad S \subset V\setminus\{0\}, j \in S. \tag{17}$$

which ensure that each customer is connected to the depot.

The second valid inequality, which was used by Letchford and Salazar-González (2006) for the distance-constrained VRP is as follows:

$$L \sum_{i \in S, j \in (V \setminus S)} x_{ij} \geq \sum_{i \in S, j \in (V \setminus S)} (d_{0j} + d_{ji})(x_{ij} + x_{ji}) + \sum_{i,j \in S} d_{ij} x_{ij} \qquad S \subset V \setminus \{0\}. \tag{18}$$

The next valid inequality is the strong 2-matching, also used for the Attractive TSP by Erdoğan et al. (2010). Note that these inequalities are originally proposed for undirected route variables, and the valid inequality stated below has been adapted to accommodate directed route variables.

$$\sum_{i,j \in S} x_{ij} + \sum_{(i,j) \in A'} x_{ij} \leq \sum_{i \in S} y_i + \frac{1}{2}(|A'| - 1) \qquad S \subset V \setminus \{0\}, A' \subset A \tag{19}$$

in which the following conditions must be hold

(i) $|\{i,j\} \cap S| = 1 \qquad (i,j) \in A'$

(ii) $(i,j) \cap (k,l) = \emptyset \qquad (i,j) \neq (k,l) \in A'$

(iii) $|A'| \geq 6$ and even.

The next valid inequality is based on the generalized large multistar inequalities originally proposed for the VRP by Gouveia (1995), used by Letchford and Salazar-González (2006), and recently used by Koç and Karaoglan (2016) as follows:

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq \frac{1}{L} \left(\sum_{i \in S} \sum_{j \in V \setminus S} \left(\overrightarrow{d_{ij}} x_{ij} + \overleftarrow{d_{ji}} x_{ji} \right) \right) \qquad S \subset V \setminus \{0\}, S \neq \emptyset \tag{20}$$

where

$$\overrightarrow{d_{ij}} = \begin{cases} d_{i0} & \text{if } j \text{ is depot} \\ d_{ij} + d_{j0} & \text{otherwise;} \end{cases} \qquad i,j \in V \setminus \{0\}$$

$$\overleftarrow{d_{ji}} = \begin{cases} d_{0i} & \text{if } j \text{ is depot} \\ d_{0j} + d_{ji} & \text{otherwise;} \end{cases} \qquad i, j \in V \setminus \{0\}.$$

For the separation algorithms of the four sets of valid inequalities we have stated above, we refer the interested reader to the papers cited for each set of valid inequality. The penultimate set of valid inequality we have utilized is

$$\sum_{i,j \in S} x_{ij} \leq |S| - r(S) \qquad S \subset V \backslash \{0\}, |S| > 2 \tag{21}$$

where $r(S)$ is a lower bound on number of vehicles required to visit the set of facilities S. Although $r(S)$ can easily be evaluated for routing problems having customer load and vehicle capacity (i.e., $r_{CVRP}(S) = \sum_{j \in W} \frac{q_j}{Q}$ where Q is the vehicle capacity), it is not as easy for routing problems without load restrictions. In this paper, we use the following property of any feasible route to obtain $r(S)$.

Let us consider a feasible route $P = \{v_0, v_1, v_2, \ldots, v_{s-1}, v_s, v_{s+1}\}$ with s arbitrary facilities (as illustrated in Figure 1) where v_0 and v_{s+1} represent the depot vertex. The total route length of P, denoted as d_P, is evaluated as

$$d_p = d_{v_0,v_1} + d_{v_1,v_2} + \ldots + d_{v_{s-1},v_s} + d_{v_s,v_{s+1}} \tag{22}$$

or equivalently

$$d_p = \frac{d_{v_0,v_1}}{2} + \frac{d_{v_0,v_1} + d_{v_1,v_2}}{2} + \frac{d_{v_1,v_2} + d_{v_2,v_3}}{2} + \ldots + \frac{d_{v_{s-1},v_s} + d_{v_s,v_{s+1}}}{2} + \frac{d_{v_s,v_{s+1}}}{2} \tag{23}$$

which can be rewritten as:

$$d_p = \frac{d_{v_0,v_1}}{2} + \sum_{i=1}^{s} \frac{d_{v_{i-1},v_i} + d_{v_i,v_{i+1}}}{2} + \frac{d_{v_s,v_{s+1}}}{2}. \tag{24}$$

In the best case, the previous v_{i-1} and the next vertex v_{i+1} of each facility v_i on any route are

the first and the second nearest vertex to v_i in P, respectively. Therefore, the total route length evaluation without sequence information in P using first nearest and second nearest vertices, called $\hat{d}_P$, gives a lower bound on d_P. Based on this property, a polynomial time algorithm to obtain $r(S)$ is given in Algorithm 1 where $\hat{d}_s$ is the total route length and Ω is the set of linked customer vertices.

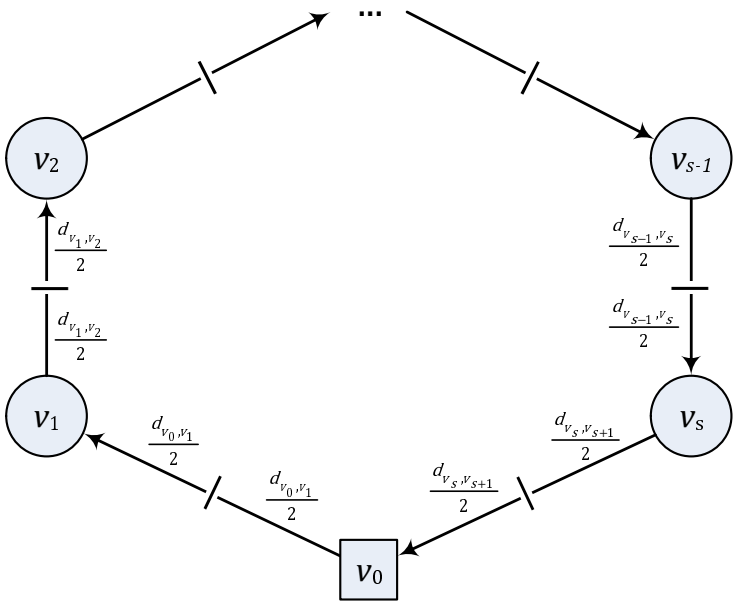

Figure 1: The total route length evaluation for any feasible solution.

The final valid inequality which bounds above the number of facilities that can be served in a feasible solution is:

$$\sum_{j \in V \setminus \{0\}} y_j \leq F \tag{25}$$

where F is the maximum number of facilities that can be served. This value is obtained as follows: for each facility $i \in V \setminus \{0\}$ the nearest distance $\bar{d}_i = \dfrac{d_{ji} + d_{ik}}{2}$ is evaluated where $j, k \in V$ are the first and the second nearest vertex to i. Then facility vertices are sorted in increasing order of $\bar{d}_i$ and F is calculated as the largest number of facilities satisfying $\sum_{i=1}^{F} \bar{d}_i \leq |K| \times L$.

Algorithm 1 Procedure to obtain the minimum number of vehicles for a set of vertices

Procedure: Evaluation of $r(S)$
Input: Node set S not including the depot vertex
Output: $r(S)$
Step 1. Initialize parameters
    $\hat{d}_S \leftarrow 0, \Omega \leftarrow 0$
//Route length evaluation without depot vertex
Step 2. Perform the following steps to all vertices (v) in S
  Step 2.1. Find the first $(\bar{v})$ and second $(\bar{\bar{v}})$ nearest vertex to (v)
    $(\bar{v}) \leftarrow \arg\min_{i \in S}\{d_{i,v}\}$
    $(\bar{\bar{v}}) \leftarrow \arg\min_{i \in S: i \neq \bar{v}}\{d_{v,i}\}$
  Step 2.2. Evaluate the effect of v to $\bar{d}_S$
    $\hat{d}_S \leftarrow \hat{d}_S + \dfrac{d_{\bar{v},v} + d_{v,\bar{\bar{v}}}}{2}$
//Route length evaluation considering depot vertex
Step 3. Find the best unlinked customer vertex (i^*) in S causing least route length increment to return to the depot
    $i^* \leftarrow \arg\min_{i \in S \setminus \Omega}\{d_{0i} - \dfrac{d_{i,\bar{\bar{v}}_i}}{2}\}$
Step 4. Evaluate the effect of i to $\bar{d}_s$
    $\hat{d}_S \leftarrow \hat{d}_S + d_{0i^*} - \dfrac{d_{i^*,\bar{\bar{v}}_{i^*}}}{2}$
    $r(S) \leftarrow \lceil \hat{d}_s / L \rceil$
    $\Omega \leftarrow \Omega \cup \{i^*\}$
Step 5.
if $|\Omega| < 2 \times r(S)$ **then**
  go to Step 3
else
  return $r(S)$

3 A Branch-and-Cut Algorithm

We developed a branch-and-cut algorithm which implements a combination of valid inequalities and implicit enumeration to solve the MVPCTP. The algorithm identifies the violated members of the valid inequalities presented in Section 2.2 by solving the associated separation problems. At each node of the branch-and-cut tree, violated inequalities are added to the formulation and the corresponding linear program is re-optimized. Branch-and-cut algorithms have been used with impressive results on several variants of the routing and covering problems (see Erdoğan et al., 2010; Karaoglan et al., 2011; Erdoğan and Laporte, 2013; Battarra et al., 2014).

3.1 An overview of the branch-and-cut algorithm

The general structure of the branch-and-cut algorithm is provided in Algorithm 2. $LP(P)$ is the LP relaxation of model P. S_t^* is the optimal solution of $LP(P)$ at the particular node t. S_{init} and $f(S_{init})$ are the initial solution and its objective function value, respectively. Similarly, S_{best} and $f(S_{best})$ are the best solution and its objective function value, respectively. Finally, Φ is the set of unexplored nodes of branch-and-cut tree.

To improve the upper bound, we construct and solve the LP relaxation of the formulation MVPCTP2 (i.e., $LP(P)$) including constraints (2–10), (16) and (25). At each iteration, we check the violations of constraints (15) and (17–21). If any of these constraints are violated, we add it to the model, solve and run the previous steps until all constraints are satisfied. If the solution is integral, then it is declared the new incumbent, else we branch on one of the fractional decision variables. To improve the performance of the branch-and-cut algorithm, we obtain an initial solution using our VNS heuristic, and use it as the starting lower bound. We also obtain solutions based on fractional solutions encountered at the nodes of the branch-and-cut tree, by applying the VNS algorithm on the partial solutions. The new solution replaces the best known solution if it has a higher objective function value, otherwise the algorithm continues.

Violated members of the constraint set (15) can be identified in $O(|V||W|)$ time simply by plugging in the solution values for each $j \in W$. We use a greedy constructive heuristic to separate the valid inequalities (17–21). In each iteration of the heuristic, the node set (S) is initialized with a randomly selected node from the set of customer nodes, and then S is expanded by a new node

that minimizing the slack of the corresponding inequality. The violation is checked and if the inequality is violated, then it is stored into S and the inequality is added to the model. These steps are repeated until current set S not expanded by a new node.

Algorithm 2 The branch-and-cut algorithm

Step 1.
Initialization
Generate S_{init}
$S_{best} \leftarrow S_{init}$
Add the root node to Φ and set the root node as current node t
Step 2.
Solve $LP(P)$ including constraints (2–10), (16) and (25)
Step 3.
if S_t^* is infeasible **or** $f(S_t^*) \leq f(S_{best})$ **then**
    $\Phi \leftarrow \Phi \backslash \{t\}$
    Go to Step 8
Step 4.
if S_t^* is integer **and** $f(S_t^*) > f(S_{best})$ **then**
    $S_{best} \leftarrow S_t^*$, $\Phi \leftarrow \Phi \backslash \{t\}$
    Go to Step 8
Step 5.
if S_t^* is not integer **then**
    Generate a feasible solution (S_t^{feas}) for the particular node t.
    if $f(S_t^{feas}) > f(S_{best})$ **then**
        $S_{best} \leftarrow S_t^{feas}$
Step 6.
Identify violated constraints (15) and (17–21) sequentially
if there are any violated constraints **then**
    Add violated constraints to the formulation $LP(P)$, reoptimize $LP(P)$ and go to Step 3
Step 7.
Select a non-integer decision variable according to branching rule, generate two nodes and add them to Φ
Step 8.
if $\Phi \neq \emptyset$ **then**
    select a node t, reoptimize $LP(P)$ and go to Step 3
else
    Return S_{best}

3.2 Heuristic algorithm

In order to obtain an initial feasible solution (S_{init}), we select the nearest facility node ($i \in V \backslash \{0\}$) to the depot as the first node of a route and the partial route length is computed ($d' = d_{0i}$). The nearest feasible unvisited facility vertex (i.e., $j \in V \backslash \{0\} : d' + d_{ij} + d_{j0} \leq L$) to the node i is then

selected and inserted at the end of the current route, and the partial route length is updated as $d' = d' + d_{ij}$. If there are no feasible unvisited facility vertices, the current route is then closed. This procedure is iteratively applied for all vehicle routes.

We develop a VNS algorithm to improve this solution. The VNS, introduced by (Mladenović and Hansen, 1997), is based on systematically changing the neighborhood structure in order to improve the current solution and aims to further explore the solution space. The VNS algorithm have been implemented for many routing problems (see Hemmelmayr et al., 2009; Stenger et al., 2013; Polat et al., 2015). The algorithmic steps of the VNS are presented in Algorithm 3.

Algorithm 3 The variable neighborhood search algorithm

Step 1.
Set $S_{imp} \leftarrow S$, $k = 1$.
Step 2.
if $k = 1$ **then**
  Apply ADD procedure to S_{imp} for randomly selected $|V|$ pairs of unvisited facility and route pairs and select the best one (S').
else if $k = 2$ **then**
  Apply MOVE procedure to S_{imp} for randomly selected $|V|$ pairs of visited facility and route pairs and select the best one (S').
else if $k = 3$ **then**
  Apply FLIP procedure to S_{imp} for randomly selected $|V|$ pairs of unvisited and visited facility pairs and select the best one (S').
else
  return S_{imp}
Step 3.
if S' better than S_{imp} **then**
  $S_{imp} \leftarrow S'$, $k = 1$ and go to Step 2.
else
  $k = k + 1$ and go to Step 2.

A feasible MVPCTP solution consists of a set of routes (R) and each route $r \in R$ is represented by an ordered list of NF_r facilities and two copies of the depot. The i^{th} facility in route r is denoted as r_i, and the position of facility j is denoted as t_j. The total length of each route r is calculated as the sum of the distances between successive facilities on a route, i.e. $D_r = d_{0,r_1} + \sum_{i=1}^{NF_r-1}(d_{r_i,r_{i+1}}) + d_{r_{NF_r},0}$. The objective function $Z = \sum_{j\in W} q_j(1-\Pi_{i\in V}(1-p_{ij})^{y_i})$ can be rewritten as $Z = Z_1 + Z_2$ with $Z_1 = \sum_{j\in W} q_j$ and $Z_2 = \sum_{j\in W}(q_j\Pi_{i\in V}(1-p_{ij})^{y_i})$. For a given solution vector $y \in [0,1]^{|V|-1}$, we define the objective function contribution of each customer vertex $j \in W$ as $k_j = q_j\Pi_{i\in V}(1-p_{ij})^{y_i}$ and therefore $Z_2 = \sum_{j\in W} k_j$.

The procedure to obtain the initial feasible solution (S_{init}) is called once at the beginning of the branch-and-cut algorithm. Additionally, in order to improve the best feasible solution throughout the algorithm, the following procedure is invoked to obtain a feasible solution from a fractional solution at any node of enumeration tree. Let x^* be optimal fractional solution of any node on the branch-and-cut tree. An empty route (r) is opened and a customer i with maximum x^*_{0i} is added to r as the first facility and the partial route length is evaluated ($d' = d_{0i}$). Next, a new feasible unvisited facility j (i. e. $j \in V\backslash\{0\} : d' + d_{ij} + d_{j0} \leq L$) having maximum x^*_{ij} is selected and added to the end of r, where i is the last facility on r. If there is no feasible facility vertex to add, then r is closed and these steps are reiterated for a new route. Finally, VNS procedure is applied in order to improve this solution. If it provides a better solution than the current lower bound it is updated, otherwise the algorithm continues.

We use three neighborhood search procedures, ADD, MOVE, and FLIP, in the VNS. The feasibility of a move within a neighborhood can easily be evaluated by considering the statistics of the current solution, such as the length of routes. However, we have utilized an additional data structure and a special evaluation scheme for the MVPCTP to speed up the objective function evaluation of the new solution. Therefore, we used following notations and definitions in the remainder. The neighborhood search procedures are as follows.

3.2.1 ADD **procedure:**

One unvisited facility (i) is selected randomly and inserted into the best position (b) of a randomly selected route (r). This procedure is feasible under the following condition.

$$D_r + \Delta^b \leq L \tag{26}$$

where $\Delta^b = d_{r_{b-1},i} + d_{i,r_b} - d_{r_{b-1},r_b}$ is the deviation on tour length when the facility i is added to the position b.

The difference (Δ_{obj}) between the objective function value of the new (Z^{new}) and the current (Z^{cur}) solution can be calculated as

$$\Delta_{obj} = Z^{new} - Z^{cur} = (Z_1^{new} - Z_2^{new}) - (Z_1^{cur} - Z_2^{cur}). \tag{27}$$

Since the first part of the objective function is constant (i.e., $Z_1^{new} = Z_1^{cur}$), the difference can be rewritten as

$$\Delta_{obj} = Z_2^{cur} - Z_2^{new} = \sum_{j \in W} k_j - \sum_{j \in W} k_j(1 - p_{ij}) = \sum_{j \in W} k_j p_{ij}. \tag{28}$$

As it can be observed from (28), Δ_{obj} is independent of the position where the new facility is inserted. Therefore, the best position for locating the new facility is selected as $b = \arg\min_{k=1,\ldots,NF_r}\{\Delta^k\}$.

3.2.2 MOVE **procedure:**

One visited facility (i) is selected randomly, removed from its current route (r) and moved to the best position (b) of a randomly selected route (s). This procedure is feasible under the following conditions.

- If $r = s$, which means that the selected customer is moved to another position in the same route, then the feasibility condition is as

  $$D_r + \Delta_b^- - \Delta^+ \leq L \tag{29}$$

  where $\Delta_b^- = d_{r_{b-1},i} + d_{i,r_b} - d_{r_{b-1},r_b}$ is the increment on the tour length by inserting the facility i to the position b and $\Delta^+ = d_{r_{t_i-1},i} + d_{i,r_{t_i+1}} - d_{r_{t_i-1},r_{t_i+1}}$ is the saving on the tour length obtained by removing facility i from its current position.

- if $r \neq s$, which means that the selected customer is moved to a position in a different route. In this situation, it is sufficient to check the feasibility for the new route since the total route length of r will always decrease and the maximum tour length restriction is always satisfied. Therefore the feasibility condition is

  $$D_s + \Delta_b^- \leq L \tag{30}$$

  where $\Delta_b^- = d_{s_{b-1},i} + d_{i,s_b} - d_{s_{b-1},s_b}$.

In this operator, no new facility is added to the solution. Therefore, the objective function value

does not change. However, moving a customer to another position may decrease the route length and give a new chance to add a new facility to the solution by using ADD operator. This situation also effects the selection of the best position (b): since there is no change in objective function we use the route length deviations to choose the best position as given below

$$\begin{aligned} &\text{if } r = s && b = \arg \min_{m=1,\dots,NF_r} \{\Delta_m^- - \Delta^+\} \\ &\text{if } r \neq s && b = \arg \min_{m=1,\dots,NF_s} \{\Delta_m^-\} \end{aligned} \tag{31}$$

3.2.3 FLIP procedure:

One unvisited facility (i) is selected randomly and set in place of a visited facility (f) of a randomly selected route (r). This move is feasible subject to the following condition

$$D_r + \Delta_f^- - \Delta_f^+ \leq L \tag{32}$$

where $\Delta_f^- = d_{r_{t_f-1},i} + d_{i,r_{t_f+1}}$ is the increment on the tour length by inserting the facility i to the position of facility f and $\Delta_f^+ = d_{r_{t_f-1},f} + d_{f,r_{t_f+1}}$ is the saving on the tour length obtained by removing facility f from its current position.

Similar to the evaluation scheme in (28), the difference Δ_{obj} between the objective function values can be calculated as

$$\Delta_{obj}^f = \sum_{j \in W} k_j - \sum_{j \in W} k_j \frac{1 - p_{ij}}{1 - p_{fj}} = \sum_{j \in W} k_j \left(1 - \frac{1 - p_{ij}}{1 - p_{fj}}\right) = \sum_{j \in W} k_j \frac{p_{ij} - p_{fj}}{1 - p_{fj}} \tag{33}$$

where Δ_{obj}^f is dependent on the removed facility. Therefore, the removed facility is selected as $f^* = \arg\max_{f=1,\dots,NF_r} \{\Delta_{obj}^f\}$ which maximizes the objective function value of the new solution.

In the VNS, we sequentially use these three neighborhood search procedures. At each iteration, new solutions are obtained randomly and the best one is selected as the new solution for the corresponding neighborhood search procedure. If a new solution is worse than the current one, the next neighborhood search procedure is then applied, and algorithm stopped after unsuccessful operation of the FLIP procedure. Otherwise, the new solution is accepted and the algorithm restarts

with the ADD procedure. Acceptance criterion is another important factor for our algorithm. For ADD and FLIP procedures, the difference between objective function values are used for both determining the best solution amongst $|V|$ neighbors and deciding if the best neighbor solution is better or not than the current one. However, for MOVE procedure the objective function value does not change. Therefore, we use the difference between total route lengths (i.e., $\sum_{r \in R} D_r$) for acceptance of the new solution.

4 Computational Experiments

In this section, we present the results of computational experiments performed in order to assess the performance of the branch-and-cut algorithm. All experiments were conducted on a server with Intel Xeon 3.16 GHz processor. The branch-and-cut algorithm was implemented in C++ and with CPLEX 12.6. A CPU time limit of one hour was imposed on the solution time for all instances.

We first provide a description about the benchmark instances and their generation procedure in Section 4.1. We then evaluate the performance of the branch-and-cut algorithm in Section 4.2.

4.1 Benchmark instances

Since there are no benchmark instances in the literature for the MVPCTP, we have generated the data sets by adapting the instances for the Capacitated VRP (Augerat et al., 1995; Christofides and Eilon, 1969; Fisher, 1994) and for the Symmetric Capacitated VRP (VRPLIB, 1998). There are six main sets in our benchmark instances: Set A (107 instances) (Augerat et al., 1995), Set B (92 instances) (Augerat et al., 1995), Set E (264 instances) (VRPLIB, 1998), Set E-n (44 instances) (Christofides and Eilon, 1969), Set F (12 instances) (Fisher, 1994), and Set P (68 instances) (Augerat et al., 1995).

We have used the first one-third of vertices as facilities, and the remaining two-third as customers. The number of available vehicles is fixed as two and three. Although the fleet size is smaller and the distance limit is shorter than the traditional values used in the VRP literature, this reflects the ability of the vehicles to serve customers remotely, requiring a smaller fleet and shorter routes as a result. The maximum distance is equal to two or three times of the average distance between facilities and depot. In total, we have generated 587 benchmark instances, ranging between 5 to

161 facilities, and 10 to 322 customers.

We have generated the coverage probabilities as follows. We first compute the probability $\hat{p}_{ij} = \min(0.95, \frac{1}{d_{ij}^2})$ for all $i, j : i \in V, j \in W$. The quadratic service decay based on the distance is characteristic to optic and magnetic sensing devices. We then determine the minimum probability $p_{min} = \min_{i \in V, j \in W}(\hat{p}_{ij})$ and the maximum probability $p_{max} = \max_{i \in V, j \in W}(\hat{p}_{ij})$. Finally, we normalize the probability values between p_{min} and 0.95 for all $(i, j : i \in V, j \in W)$ as $p_{ij} = p_{min} + (0.95 - p_{min})\frac{\hat{p}_{ij} - p_{min}}{p_{max} - p_{min}}$. The normalization process results in a full range of coverage probability values, and ensures a robust computational test.

4.2 Results for the branch-and-cut algorithm

This section presents the results of our experiments on benchmark instances. Table 1 shows the average results of all instance sets. For detailed results of all instances, the reader is referred to the supplementary material of this article. In the tables, the columns show: instance set, range of total number of customer and facility vertices ($|V| + |W|$), range of total number of facility vertices ($|V|$), percentage deviation ($Dev(\%)$), total computation time in seconds (CPU), and finally total number of optimal solutions ($\#Opt$) over total number of instances of each set indicated in parenthesis. The percentage deviation between the linear relaxation bound (UB) produced by a particular formulation and lower bound (LB) obtained by the branch-and-cut algorithm is calculated as $((UB - LB)/UB) \times 100$.

Table 1 shows that our branch-and-cut algorithm obtains optimal solutions for all 107 in Set A, 91 out of 92 instances in Set B, 217 out of 264 instances in Set E, all 44 instances in Set E-n, 11 optimal solutions out of 12 instances for Set F, and all 68 instances in Set P. In total, our branch-and-cut algorithm finds 538 optimal solutions out 587 instances. Average deviations are 0.00%, 0.17%, 2.59%, 0.00%, 0.07%, 0.00% and 0.47% for Sets A, B, E, E-n, F and P, respectively. The average time required by the branch-and-cut algorithm to solve Sets A, B, E, E-n, F and P are 1.17, 45.46, 743.55, 6.60, 370.10, 2.63 and 194.92 seconds, respectively.

Table 2 shows the average results of instance sets in an incremental order of number of customer and facilities for Sets A, B, E, E-n, F, and P. The number in parentheses represents the total number of instances. Our results clearly show that for Sets A, E-n and P instances, the branch-and-cut

Table 1: Average results of all instance sets.

Instance set	Range of $\|V\| + \|W\|$	Range of $\|V\|$	Dev (%)	CPU	$\#Opt$
Set A	[31-79]	[10-26]	0.00	1.17	107 (107)
Set B	[30-77]	[10-26]	0.17	45.46	91 (92)
Set E	[15-483]	[5-161]	2.59	743.55	217 (264)
Set E-n	[29-100]	[10-33]	0.00	4.04	44 (44)
Set F	[44-134]	[15-45]	0.00	64.98	11 (12)
Set P	[39-100]	[13-33]	0.00	1.76	68 (68)
Average			0.46	143.49	
Total					538 (587)

algorithm is able to obtain optimal solutions for all number of facility ranges. For both Sets B and F, the algorithm is not able to find optimal solution for only one instance. For Set E, until 33 facilities the algorithm obtains optimal solutions. As it can be seen clearly, computation time generally increases when the number of customer and facility vertices increases.

In Table 3, we report the average numbers of generated valid inequalities for each instance set. We interpret these values as a measure of comparative effectiveness of the valid inequalities. It can be observed that the inequality (21) is added more frequently with respect to the rest. The second and third most added inequalities are (18) and (20), respectively. The inequalities (17) and (19) are the least frequently added inequalities.

To show the impact of the VNS heuristic, we present an analysis in Table 4, where we report the average number of times VNS is applied throughout the branch-and-cut tree (Times applied), the average number of times it resulted in an improvement of solution quality (Times improved), and the frequency of improvement which is calculated as (Times improved / Times applied)$\times$100. The average frequency of improvement is calculated as 0.67 %, with 9.28%, 0.25%, 0.62%, 2.60%, 0.11%, and 5.47% for Sets A, B, E, E-n, F, and P, respectively. Note that a low frequency implies that the optimal (or best known) solution was found early in the search, and further applications of the VNS did not provide improvements.

As a final comment, we state that the average percentage of facility vertices visited by the vehicles are 55.11%, 60.93%, 43.96%, 44.22%, 49.84%, and 41.59% for Sets A, B, E, F, and P, respectively. We therefore conclude that any direct enumeration scheme is very unlikely to achieve a satisfactory performance.

Table 2: Average results on Sets A, B, E, E-n, F, and P.

Instance set	Range of $\|V\| + \|W\|$	Range of $\|V\|$	Dev (%)	CPU
Set A (18)	[31-35]	[10-12]	0.00	0.06
Set A (18)	[35-38]	[12-13]	0.00	0.05
Set A (18)	[38-45]	[13-15]	0.00	0.20
Set A (18)	[45-54]	[15-18]	0.00	0.43
Set A (18)	[59-62]	[20-21]	0.00	0.43
Set A (17)	[62-79]	[21-26]	0.00	6.14
Set B (16)	[30-37]	[10-12]	0.00	0.06
Set B (16)	[38-43]	[13-14]	0.00	0.18
Set B (16)	[44-49]	[15-16]	0.00	2.24
Set B (15)	[50-56]	[17-19]	0.00	0.41
Set B (15)	[56-65]	[19-22]	0.00	25.00
Set B (14)	[65-77]	[22-26]	1.11	268.71
Set E (44)	[15-29]	[5-10]	0.00	0.03
Set E (44)	[30-75]	[10-25]	0.00	11.86
Set E (44)	[75-100]	[25-33]	0.00	5.01
Set E (44)	[100-150]	[33-50]	0.02	129.68
Set E (44)	[150-255]	[50-85]	0.71	1259.76
Set E (44)	[261-483]	[87-161]	14.84	3054.98
Set E-n (8)	[29-29]	[10-10]	0.00	0.04
Set E-n (8)	[32-50]	[11-17]	0.00	0.36
Set E-n (8)	[75-75]	[25-25]	0.00	7.08
Set E-n (7)	[75-75]	[25-25]	0.00	5.41
Set E-n (7)	[75-100]	[25-33]	0.00	9.00
Set E-n (6)	[100-100]	[33-33]	0.00	21.59
Set F (2)	[44-44]	[15-15]	0.00	0.10
Set F (2)	[44-44]	[15-15]	0.00	0.05
Set F (2)	[71-71]	[24-24]	0.00	15.36
Set F (2)	[71-71]	[24-24]	0.00	244.42
Set F (2)	[134-134]	[45-45]	0.00	44.10
Set F (2)	[134-134]	[45-45]	0.41	1916.56
Set P (12)	[39-49]	[13-16]	0.00	0.37
Set P (12)	[49-50]	[16-17]	0.00	0.44
Set P (11)	[54-54]	[18-18]	0.00	0.48
Set P (11)	[54-59]	[18-20]	0.00	0.98
Set P (11)	[59-75]	[20-25]	0.00	2.99
Set P (11)	[75-100]	[25-33]	0.00	10.95

Table 3: Average number of valid inequalities added.

Instance set	Valid inequality set				
	(17)	(18)	(19)	(20)	(21)
Set A	5.61	23.09	1.26	7.54	13.68
Set B	11.92	16.86	14.66	17.86	21.50
Set E	65.50	284.41	38.52	93.16	1112.70
Set E-n	38.45	54.91	4.52	12.00	53.55
Set F	81.17	94.17	107.08	25.25	137.50
Set P	22.09	41.81	1.81	9.84	35.26
Average	39.45	145.65	22.59	48.63	517.20

Table 4: The effect of the VNS algorithm.

Instance set	Times applied	Times improved	Improvement (%)
Set A	17.93	1.66	9.28
Set B	611.14	1.55	0.25
Set E	543.07	3.39	0.62
Set E-n	115.57	3.00	2.60
Set F	2225.92	2.50	0.11
Set P	57.81	3.16	5.47
Average	404.16	2.71	0.67

5 Conclusions

We have introduced the MVPCTP which is a multi-vehicle and probabilistic coverage generalization of the CTP. We have provided a nonlinear integer programming model and have devised a linearization scheme for the objective function that we have proved to be concave. We have adapted a set of valid inequalities from the VRP literature, to strengthen the linear programming relaxation of the formulation. We have developed a branch-and-cut algorithm, and have proposed a VNS algorithm to improve the initial solution and lower bounds found during the search process of the solution approach.

We have performed extensive computational experiments on 587 benchmark instances adapted from the literature, with the instance size ranging from 5 to 161 facilities, and from 10 to 322 customers. Results show that our algorithm is highly effective on the MVPCTP by yielding 538 optimal solutions out 587 instances within one hour of CPU time.

Acknowledgments: We thank the two anonymous reviewers for their positive and constructive comments, which helped us to improve the paper.

References

Ahmadi-Javid, A., Seyedi, P., Syam, S. S., 2017. A survey of healthcare facility location. Computers & Operations Research 79, 223–263.

Allahyari, S., Salari, M., Vigo, D., 2015. A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. European Journal of Operational Research 242, 756–768.

Altay, N., Green III, W. G., 2006. OR/MS research in disaster operations management. European Journal of Operational Research 175, 475–493.

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., Rinaldi, G., 1995. Computational results with a branch and cut code for the capacitated vehicle routing problem. Rapport de recherche- IMAG.

Battarra, M., Erdoğan, G., Vigo, D., 2014. Exact algorithms for the clustered vehicle routing problem. Operations Research 62, 58–71.

Caunhye, A. M., Nie, X., Pokharel, S., 2012. Optimization models in emergency logistics: A literature review. Socio-Economic Planning Sciences 46, 4–13.

Christofides, N., Eilon, S., 1969. An algorithm for the vehicle-dispatching problem. Journal of the Operational Research Society 20, 309–318.

Church, R., ReVelle, C., 1974. The maximal covering location problem. In Papers of the Regional Science Association 32 (pp. 101–118). Springer-Verlag.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W. P., Vigo, D. 2007. Vehicle routing. In: Barnhart, C. and Laporte, G. (Eds.), Transportation, Handbooks in Operations Research and Management Science (pp 367–428). Elsevier, Amsterdam.

Current, J. R., Schilling, D. A., 1989. The covering salesman problem. Transportation Science 23, 208–213.

Dang, D. C., El-Hajj, R., Moukrim, A., 2013. A branch-and-cut algorithm for solving the team orienteering problem. In International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems (pp. 332–339). Springer, Berlin Heidelberg.

Daskin, M. S., 1983. A maximum expected covering location model: formulation, properties and heuristic solution. Transportation Science 17, 48–70.

De La Torre, L. E., Dolinskaya, I. S., Smilowitz, K. R., 2012. Disaster relief routing: Integrating research and practice. Socio-Economic Planning Sciences 46, 88–97.

Erdoğan, G., Cordeau, J. F., Laporte, G., 2010. The attractive traveling salesman problem. European Journal of Operational Research 203, 59–69.

Erdoğan, G., Laporte, G., 2013. The orienteering problem with variable profits. Networks 61, 104–116.

Fisher, M. L., 1994. Optimal solution of vehicle routing problems using minimum K-tree. Operations Research 42, 626–642.

Flores-Garza, D. A., Salazar-Aguilar, M. A., Ngueveu, S. U., Laporte, G., 2017. The multi-vehicle cumulative covering tour problem. Annals of Operations Research 258, 761-780.

Gendreau, M., Laporte, G., Semet, F., 1997. The covering tour problem. Operations Research 45, 568–576.

Gendreau, M., Jabali, O., Rei, W., 2016. Future research directions in stochastic vehicle routing. Transportation Science 50, 1163–1173.

Golden, B. L., Raghavan S., Wasil E. A. 2008. The vehicle routing problem: latest advances and new challenges. Springer, New York.

Gouveia, L., 1995. A result on projection for the vehicle routing problem. European Journal of Operational Research 85, 610–624.

Hà, M. H., Bostel, N., Langevin, A., Rousseau, L. M., 2013. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. European Journal of Operational Research 226, 211–220.

Hachicha, M., Hodgson, M. J., Laporte, G., Semet, F., 2000. Heuristics for the multi-vehicle covering tour problem. Computers & Operations Research 27, 29–42.

Hansen, P., Mladenović, N., 2014. Variable neighborhood search. In Search methodologies (pp. 313–337), Springer, New York.

Hemmelmayr, V. C., Doerner, K. F., Hartl, R. F., 2009. A variable neighborhood search heuristic for periodic routing problems. European Journal of Operational Research 195, 791–802.

Jozefowiez, N., Semet, F., Talbi, E., 2007. The bi-objective covering tour problem. Computers & Operations Research 34, 1929–1942.

Jozefowiez, N, 2014. A branch-and-price algorithm for the multi-vehicle covering tour problem. Networks 64, 160–168.

Karaoglan, I., Altiparmak, F., Kara, I., Dengiz, B., 2011. A branch and cut algorithm for the location-routing problem with simultaneous pickup and delivery. European Journal of Operational Research 211, 318–332.

Koç, Ç, Karaoglan, I, 2016. The green vehicle routing problem: A heuristic based exact solution approach. Applied Soft Computing 39, 154–164.

Laporte, G. 2009. Fifty years of vehicle routing. Transportation Science 43, 408–416.

Letchford, A. N., Salazar-González, J. J., 2006. Projection results for vehicle routing. Mathematical Programming 105, 251–274.

Lopes, R., Souza, V. A. A., da Cunha, A. S., 2013. A branch-and-price algorithm for the multi-vehicle covering tour problem. Electronic Notes in Discrete Mathematics 44, 61–66.

Miller, C., Tucker, A., Zemlin, R., 1960. Integer programming formulations and traveling salesman problems. Journal of the ACM 7, 326–329.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & Operations Research 24, 1097–1100.

Naji-Azimi, Z., Renaud, J., Ruiz, A., Salari, M., 2012. A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. European Journal of Operational Research 222, 596–605.

Nolz, P. C., Doerner, K. F., Gutjahr, W. J., Hartl, R. F., 2010. A bi-objective metaheuristic for disaster relief operation planning. In: Advances in multi-objective nature inspired computing (pp.167–187). Springer, Berlin Heidelberg.

Pajares, G., 2015. Overview and Current Status of Remote Sensing Applications Based on Unmanned Aerial Vehicles (UAVs). Photogrammetric Engineering & Remote Sensing 81, 281–329.

Pham, T.A., Há, M.H., Nguyen, X.H., 2017. Solving the multi-vehicle multi-covering tour problem. Computers & Operations Research 88, 258–278.

Polat, O., Kalayci, C. B., Kulak, O., Günther, H. O., 2015. A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit. European Journal of Operational Research 242, 369–382.

Stenger, A., Vigo, D., Enz, S., Schwind, M., 2013. An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. Transportation Science 47, 64–80.

Toth, P. Vigo, D., eds. 2014. Vehicle Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization, Philadelphia.

Vargas, L., Jozefowiez, N., Ngueveu, S. U., 2015. A selector operator-based adaptive large neighborhood search for the covering tour problem. In Learning and Intelligent Optimization (pp. 170–185). Springer, Switzerland.

VRPLIB, 1998. A vehicle routing problem library-Symmetric CVRP instances. <http://or.dei.unibo.it/library/vrplib-vehicle-routing-problem-library>.