


## EXACT ALGORITHMS AND HEURISTICS FOR THE CAPACITATED COVERING SALESMAN PROBLEM

LUCAS PORTO MAZIERO, FABIO LUIZ USBERTI\*  AND CELSO CAVELLUCCI

**Abstract.** We introduce the Capacitated Covering Salesman Problem (CCSP), a novel extension of classical vehicle routing that integrates the concept of service by coverage into capacity-constrained routing. In the CCSP, a fleet of vehicles based at a single depot is tasked with satisfying customer demands at minimum travel cost, where customers can be serviced indirectly by being located within the coverage range of a visited vertex. This extension addresses practical challenges in routing scenarios where direct access to every customer is either difficult or inefficient. We develop a comprehensive solution framework that combines an exact integer linear programming (ILP) formulation with a biased random-key genetic algorithm (BRKGA). The proposed ILP model captures both capacity and coverage constraints, while the BRKGA efficiently explores the solution space, especially for larger instances. Furthermore, we enhance our approach through a hybrid strategy that integrates the best solutions from the BRKGA with an exact optimization process, thereby refining the results even further. Extensive computational experiments on a benchmark set derived from established vehicle routing instances demonstrate the effectiveness and robustness of our methods. The ILP formulation is able to find optimal solutions for smaller instances, and the BRKGA consistently delivers high-quality solutions for more complex problems, with the hybrid approach yielding additional improvements. Our work not only highlights the practicality of incorporating coverage in routing but also lays a solid foundation for future research on advanced routing models that balance direct visitation with flexible service strategies.

**Mathematics Subject Classification.** 68R05, 90B06, 90C10, 90C57.

Received May 1, 2025. Accepted September 23, 2025.

### 1. INTRODUCTION

The Capacitated Vehicle Routing Problem (CVRP) [7] is a foundational problem in combinatorial optimization, widely studied due to its practical relevance in logistics and supply chain management. It seeks to satisfy the demands of a set of geographically distributed customers using a fleet of identical vehicles based at a central depot, while minimizing the total travel distance. Each vehicle must depart from and return to the depot without exceeding its capacity constraint [5].

Numerous CVRP variants have been explored, incorporating constraints related to time windows, resource availability, and customer accessibility [2, 13]. One practical challenge in routing problems arises when certain customers are located in areas where direct vehicle access is difficult or impractical. This limitation can be

---

*Keywords.* Covering routing problems, integer linear programming, metaheuristic, matheuristic, combinatorial optimization.

Institute of Computing, University of Campinas, Av. Albert Einstein 1251, 13083-852 Campinas, SP, Brazil.

\*Corresponding author: [fusberti@ic.unicamp.br](mailto:fusberti@ic.unicamp.br)

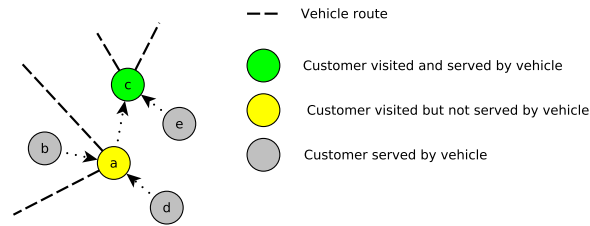


FIGURE 1. Example of covering ranges.

addressed through the concept of service by coverage, where a customer is considered serviced if they fall within the coverage range of a visited location, rather than requiring a direct visit.

For example, in Figure 1, customers  $b$  and  $d$  are within the coverage range of customer  $a$ . Similarly, customers  $a$  and  $e$  can be indirectly serviced *via* vertex  $c$ , provided the vehicle serving  $c$  has sufficient remaining capacity.

The first problem using the concept of servicing by coverage is the Covering Salesman Problem (CSP) [6], stated as follows. Given an undirected graph with cost attributed to the edges, the objective is to determine a minimum cost cycle such that every vertex out of the cycle is covered by at least one vertex in the cycle. The CSP generalizes the Travelling Salesman Problem (TSP) [3] in the case where each vertex only covers itself, from which follows that CSP is NP-hard.

Generalizations of the Covering Salesman Problem (CSP) have been investigated in the literature. For instance, Golden *et al.* [14] presents a variant in which each vertex has a covering demand, specifying the number of times it must be covered by the tour. Additionally, each vertex is associated with a fixed cost incurred when visited. A heuristic based on local search is proposed, exploring exchange, removal, and insertion neighborhoods.

The Covering Tour Problem (CTP) was introduced as a variant in which the vertex set is partitioned into those that can be visited ( $V$ ), must be visited ( $T \subseteq V$ ), and cannot be visited ( $W$ ). The goal is to find a minimum-cost Hamiltonian cycle over a subset  $S \subseteq V$  such that all vertices in  $T$  are included, no vertex from  $W$  is visited, and every vertex in  $W$  is covered by at least one vertex in  $S$ . Both exact and heuristic methods have been proposed to solve the problem [12].

The Multi-Vehicle Covering Tour Problem ( $m$ -CTP) extends the CTP by introducing multiple vehicles, each subject to constraints on route length and the number of visited vertices [18]. The  $m$ -CTP was used as the basis to formulate a problem of locating distribution centers for humanitarian aid in disaster areas [32]. Methodologies to solve the  $m$ -CTP include branch-and-cut [17], column generation [31], branch-and-price [22], constructive heuristics [18], evolutionary metaheuristic [17], variable neighborhood descent [23].

The Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP) combines elements of the Multi-Depot Vehicle Routing Problem (MDVRP) [36] and the Covering Salesman Problem (CSP), and was first introduced in [2]. In the MDCTVRP, customer demands can be fulfilled either by direct visits or by coverage, *i.e.*, by being within the coverage range of at least one visited customer. The authors proposed two Mixed-Integer Linear Programming (MILP) formulations, along with a hybrid metaheuristic that integrates Greedy Randomized Adaptive Search Procedure (GRASP), Iterated Local Search (ILS), and Simulated Annealing (SA).

Matheuristic methods, which combine heuristic and exact approaches, have recently been studied in the literature and applied to a variety of routing problems. The Multi-Period Home Health Care Routing and Scheduling Problem (HHCRSP) consists of assigning suitable caregivers to serve patients at their homes while designing a set of optimized visit routes [29]. The authors formulated the problem as a MILP model and proposed two matheuristic strategies that combine Adaptive Large Neighborhood Search (ALNS) with the MILP model.

Another application of matheuristics is the design of forest fire monitoring systems that incorporate surveillance towers, monitoring balloons, and drones [8]. The goal is to integrate these monitoring technologies to

maximize terrain coverage. The problem is formulated as a MILP formulation that includes both location decisions and drone routing among monitoring sites. The authors also proposed a matheuristic that integrates other components: perturbation procedures, local search, a global reset strategy, a local reset strategy, and an acceptance criterion. The proposed methodologies are evaluated on both randomly generated instances and a real-world case study in Chile.

Matheuristics have also been extended to hybrid truck–drone systems designed for emergency response operations in areas with vulnerable road networks and infrastructure [39]. Most existing approaches adopt a fixed-binding mode between trucks and drones, which limits drones’ flexibility and reduces overall efficiency. To address this limitation, the authors proposed a dynamic truck–drone collaboration (DTDC) strategy that allows drones to change their take-off and landing points across different trucks. However, the DTDC strategy introduces additional complexity into the scheduling problem. To tackle this challenge, the authors developed a matheuristic that decomposes the scheduling problem into three decision processes: demand allocation, truck routing, and drone scheduling. Furthermore, two alternative matheuristic algorithms are proposed, one emphasizing solution accuracy and the other computational efficiency.

It is important to note that, unlike the CTP, the CSP lacks a well-established multi-vehicle variant. This work addresses that gap by proposing the Capacitated Covering Salesman Problem (CCSP), an NP-hard problem that generalizes both the CVRP and the CSP. In the CCSP, vertices with positive demands must be serviced by a fleet of capacitated vehicles based at a central depot. The objective is to minimize the total cost of a set of routes that collectively cover all demands. The CCSP can be seen as a natural extension of the CSP where servicing capacity is limited, and as a generalization of the CVRP, where indirect service *via* coverage is permitted. The main differences between CCSP and *m*-CTP are listed below.

- The CCSP associates demands with vertices, unlike the *m*-CTP;
- The *m*-CTP requires mandatory visits to a subset of vertices ( $T \subseteq V$ ), which is not the case in CCSP;
- The *m*-CTP imposes constraints on route length and number of visited vertices, whereas CCSP constrains vehicle routes by demand capacity;
- The CCSP generalizes the CSP, while the *m*-CTP generalizes the CTP.

The CCSP provides a modeling framework that integrates key elements of coverage and routing, which are specially relevant for real-world applications where servicing all customers *in loco* is impractical or when customers are located in hard-to-reach areas. For example, the design of satellite distribution centers for humanitarian aid requires relief items to be delivered to affected regions, even when certain customers are located in areas with difficult access, thus requiring strategic coverage decisions combined with effective vehicle routing [32, 34]. Similar applications arise in the context of providing primary health care in hard-to-reach rural areas, where mobile health care facilities or supply vehicles must be assigned to service locations that guarantee coverage of dispersed customers while respecting vehicle capacity constraints [20, 30]. Disaster relief further demonstrates the importance of the CCSP, as relief centers must quickly deploy fleets of vehicles to cover affected areas under constrained resources [9, 21, 25].

*Our contributions.* This work introduces a unified framework that integrates two classical combinatorial optimization problems, the Covering Salesman Problem (CSP) and the Vehicle Routing Problem (VRP), to model routing scenarios involving multiple vehicles with limited capacity under service by coverage. We formulate the resulting Capacitated Covering Salesman Problem (CCSP) using integer linear programming (ILP) and discuss two alternative formulations. Due to the inherent complexity of solving large instances optimally, we develop heuristic methods engineered to address large instances. Specifically, we propose a Biased Random-Key Genetic Algorithm (BRKGA) for solving the CCSP and complement it with a matheuristic approach to enhance search intensification.

This paper is organized as follows. Section 2 formally describes the CCSP and introduces two ILP formulations. Section 3 details the BRKGA and the associated intra-route and inter-route improvement procedures. Computational results on representative benchmark instances are presented and analyzed in Section 4. Finally, Section 5 concludes the paper with a summary of findings and perspectives for future work.

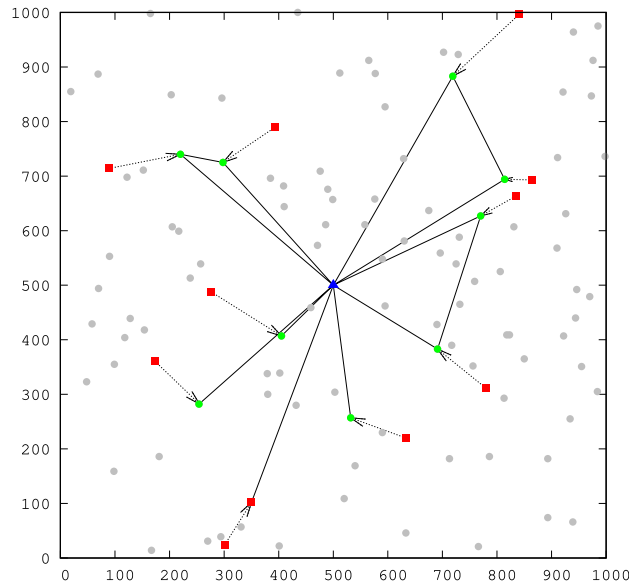


FIGURE 2. Optimal solution for the CCSP instance  $X-n115-w11-c7$ , where 115 denotes the total number of vertices, 11 indicates the number of vertices with positive demand, and 7 represents the number of vertices that can be covered by each vertex.

## 2. PROBLEM DESCRIPTION

Consider a complete undirected graph  $G = (V, E)$ , where each vertex  $v \in V$  is associated with a non-negative demand  $d_v$ , and each edge  $e \in E$  has a non-negative metric cost  $c_e$ . A distinguished vertex  $v_0 \in V$  represents the depot. Define  $V_0 = V \setminus \{v_0\}$  as the set of customer vertices, and let  $V_d = \{v \in V : d_v > 0\}$  denote the subset of vertices with positive demand. A fleet of  $M$  identical vehicles, each with capacity  $Q$ , is available to service all vertices in  $V_d$ .

For each vertex  $v \in V$ , let  $C(v) \subseteq V$  denote the set of vertices that can cover  $v$ , and  $D(v) \subseteq V$  the set of vertices covered by  $v$ . We assume that coverage is reflexive, *i.e.*,  $v \in C(v)$  and  $v \in D(v)$  for all  $v \in V$ .

A route is defined as a non-empty subset  $R \subseteq E$  of edges such that the subgraph  $G[R]$  forms a simple cycle that includes the depot  $v_0$ . The objective of the CCSP is to determine  $M$  such routes of minimum total cost, subject to the following constraints:

- Each vertex is visited at most once;
- Each demand  $d_v$  for  $v \in V_d$  must be serviced by some route  $R$ , meaning that either  $v$  or at least one vertex in  $C(v)$  must be visited by  $R$ , and the corresponding demand  $d_v$  must be accounted for in the vehicle's load;
- The total demand serviced by any vehicle must not exceed its capacity  $Q$ .

Figure 2 illustrates an optimal solution for a CCSP instance. Routes are shown as black lines, with the depot  $v_0$  represented by a blue triangle. Vertices with positive demand are marked as red squares, while visited vertices with zero demand appear as green circles. Arrows indicate which route services each demand.

The ILP formulation  $\text{CCSP}_1$  models the CCSP. Let  $\delta(v)$  denote the cut-set of edges incident to vertex  $v$ , and  $\delta(S)$  the edge cut-set of a subset  $S \subseteq V$ . The model uses the following decision variables:  $x_e \in \mathbb{Z}^+$  indicates how many times edge  $e \in E$  is traversed;  $y_v \in \{0, 1\}$  indicates whether vertex  $v \in V$  is visited;  $z_{uv} \in \{0, 1\}$  specifies whether vertex  $u \in V_d$  is serviced *via* vertex  $v \in C(u)$ ; and  $K \in \mathbb{Z}^+$  represents the number of vehicles used.

The objective function minimizes the total cost of traversed edges across all vehicle routes. The constraints are interpreted as follows:

- Constraint (2): Ensures that exactly  $2K$  edges are incident to the depot  $v_0$ , *i.e.*, each of the  $K$  vehicle routes must both depart from and return to the depot.
- Constraint (3): Guarantees degree-2 connectivity for each visited customer vertex  $v \in V_0$ , ensuring that if  $v$  is visited ( $y_v = 1$ ), then two edges must be incident to  $v$ ; otherwise, no edges are used.
- Constraint (4): Imposes that each vertex with positive demand must be covered by at least one visited vertex in its coverage set  $C(u)$ .
- Constraint (5): Enforces that a demand  $d_u$  can only be assigned to a vertex  $v$  if  $v$  is visited, ensuring consistency between  $y_v$  and  $z_{uv}$ .
- Constraint (6): Requires that each demand  $d_u$  is assigned to exactly one covering vertex in  $C(u)$ .
- Constraint (7): Enforces capacity constraints using a generalized connectivity inequality over subsets  $S \subseteq V_0$ . The right-hand side provides a lower bound on the number of edges required for vehicles to enter or exit the subset  $S$  in order to service demands  $d_u$  that are covered by vertices in  $S$ .
- Constraints (8)–(11): Specify the domain of the decision variables. Edges not incident to the depot can be used at most once, while depot-adjacent edges can be used up to twice (for two vehicles entering/exiting). Binary variables  $y_v$  and  $z_{uv}$  indicate visitation and demand assignment, respectively.  $K$  is a positive integer.

(CCSP<sub>1</sub>)

$$\text{minimize } \sum_{e \in E} c_e x_e, \tag{1}$$

subject to

$$\sum_{e \in \delta(v_0)} x_e = 2K, \tag{2}$$

$$\sum_{e \in \delta(v)} x_e = 2y_v \quad \forall v \in V_0, \tag{3}$$

$$\sum_{v \in C(u)} y_v \geq 1 \quad \forall u \in V_d, \tag{4}$$

$$z_{uv} \leq y_v \quad \forall u \in V_d, \quad \forall v \in C(u), \tag{5}$$

$$\sum_{v \in C(u)} z_{uv} = 1 \quad \forall u \in V_d, \tag{6}$$

$$\sum_{e \in \delta(S)} x_e \geq \frac{2}{Q} \sum_{u \in V_d} \sum_{v \in (S \cap C(u))} d_u z_{uv} \quad \forall S \subseteq V_0, \tag{7}$$

$$x_e \in \{0, 1\} \quad \forall e \notin \delta(v_0), \tag{8}$$

$$x_e \in \{0, 1, 2\} \quad \forall e \in \delta(v_0), \tag{9}$$

$$y_v \in \{0, 1\} \quad \forall v \in V_0, \tag{10}$$

$$z_{uv} \in \{0, 1\} \quad \forall u \in V_d, \quad \forall v \in C(u), \tag{11}$$

$$K \in \mathbb{Z}^+. \tag{12}$$

A second formulation, denoted CCSP<sub>2</sub>, can be obtained by eliminating the binary variables  $y_v$  through substitution using the degree constraints in (3). This leads to a more compact formulation relying solely on edge and service assignment variables.

- Constraint (13): Ensures that if a vertex  $v \in V_0$  is used to service any demand  $u \in V_d$  (*i.e.*,  $z_{uv} = 1$ ), then  $v$  must have degree at least 2 in the solution, guaranteeing its participation in a feasible route.
- Constraint (14): Limits the degree of each vertex  $v \in V_0$  to at most 2 if it services any demand (*i.e.*,  $\sum_{u \in V_d} z_{uv} > 0$ ), and enforces a degree of 0 otherwise. This ensures that each vertex is visited by at most one vehicle and is not included in more than one route.

- Constraints (2), (6)–(9), (11)–(12): These are inherited from formulation  $\text{CCSP}_1$  and collectively enforce depot visitation, correct service assignment, vehicle capacity limitations, and variable domain restrictions.

( $\text{CCSP}_2$ )

$$\text{minimize } \sum_{e \in E} c_e x_e,$$

subject to

$$\sum_{e \in \delta(v)} x_e \geq 2z_{uv} \quad \forall v \in V_0, \quad \forall u \in V_d, \quad (13)$$

$$\sum_{e \in \delta(v)} x_e \leq \min \left\{ 2 \sum_{u \in V_d} z_{uv}, 2 \right\} \quad \forall v \in V_0, \quad (14)$$

(2), (6)–(9), (11)–(12).

Preliminary experiments have shown that, even though the  $\text{CCSP}_2$  formulation has fewer variables than  $\text{CCSP}_1$ , the overall quality of the upper and lower bounds obtained using  $\text{CCSP}_1$  is superior. Therefore, only the  $\text{CCSP}_1$  formulation is considered in the computational experiments presented in this work.

### 3. BRKGA FOR THE CCSP

Inspired by the Darwinian principle of survival of the fittest, the BRKGA [15] is an evolutionary metaheuristic in which a population of individuals, each representing a solution to a combinatorial optimization problem, evolves toward high-quality solutions over successive generations.

Each individual is encoded as a chromosome, which is a vector of real numbers called random keys, where each allele is independently drawn from the uniform distribution over the interval  $[0, 1)$ . A problem-specific decoder function maps these chromosomes into feasible solutions of the problem, and therefore plays a central role in the quality of the solutions produced.

An initial population of random chromosomes is generated and subjected to selective pressure, where individuals with higher fitness values are more likely to survive and contribute offspring to the next generation. The BRKGA partitions the population into two groups: *elite* and *non-elite* individuals, with sizes determined by fixed parameters. The elite set consists of the best-performing individuals, while the non-elite set includes all others. A subset of the non-elite group is composed of *mutants*, which are randomly generated chromosomes introduced to promote genetic diversity.

In each generation, the BRKGA executes the following steps:

- (1) Decode all chromosomes and evaluate their fitness;
- (2) Identify the best individuals to form the elite set;
- (3) Preserve the elite set in the population for the next generation;
- (4) Introduce mutant individuals to promote diversity;
- (5) Generate offspring through biased crossover between elite and non-elite individuals and insert them into the next generation.

The BRKGA has proven to be an effective and versatile metaheuristic for solving various routing problems [1, 11, 26, 27, 33]. In the following subsections, we describe how the BRKGA framework is adapted to address the Capacitated Covering Salesman Problem (CCSP).

#### 3.1. Solution encoding and decoder function

Each solution is encoded as a chromosome  $\mathcal{X} = (x_1, \dots, x_n)$  of length  $n = |V_d|$ , where  $x_i \in [0, 1)$  for  $i = 1, \dots, n$ . Each component  $x_i$  is a random key associated with a demand vertex in  $V_d$ . This encoding defines

a permutation over  $V_d$  by sorting the components of  $\mathcal{X}$  in non-descending order  $\mathcal{X}'$ . The resulting permutation guides the order in which demand vertices are considered for service during decoding.

The proposed decoder operates in two phases (Algorithms 1 and 2), described in the following subsections.

*Phase I: Bin-Packing-Based Demand Assignment.* The minimum number of vehicles required to service all demands can be estimated by solving a *Bin Packing Problem* (BPP) [24], which is known to be NP-hard. In our decoder, demand vertices from  $V_d$  are assigned to vehicles using an approximate solution to the BPP, implemented *via* the Best Fit Algorithm (BFA). The BFA assigns each vertex to the vehicle with the least residual capacity that can still accommodate its demand. If no such vehicle exists, a new vehicle is allocated to start a new route (see Fig. 3).

Algorithm 1 presents the pseudo-code of the BFA as applied in our decoder. The algorithm can be efficiently implemented using self-balancing search trees, resulting in a worst-case time complexity of  $O(n \log n)$ .

---

**Algorithm 1.** Best Fit Algorithm.

---

**Input:** A sorted vector  $\mathcal{X}'$  of random keys (permutation of  $V_d$ )

**Output:** A set of vehicles  $\mathcal{M} = \{1, \dots, M\}$  and their assigned demand vertices  $\mathcal{A} = \{A_1, \dots, A_M\}$

```

1:  $m \leftarrow 0$ ;  $\mathcal{M} \leftarrow \emptyset$ ;  $\mathcal{A} \leftarrow \emptyset$ 
2: for each  $x'_i \in \mathcal{X}'$  do
3:    $v \leftarrow \text{getVertex}(i)$   $\triangleright$  Returns the vertex associated with  $x'_i$ 
4:   if  $\exists m \in \mathcal{M}$  such that  $\sum_{u \in A_m} d_u \leq Q - d_v$  then
5:      $m \leftarrow \arg \max_{m' \in \mathcal{M}} \left\{ \sum_{u \in A_{m'}} d_u : \sum_{u \in A_{m'}} d_u \leq Q - d_v \right\}$ 
6:      $A_m \leftarrow A_m \cup \{v\}$ 
7:   else
8:      $m \leftarrow m + 1$ 
9:      $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$ 
10:     $A_m \leftarrow \{v\}$ 
11:     $\mathcal{A} \leftarrow \mathcal{A} \cup \{A_m\}$ 
12:   end if
13: end for

```

---

*Phase II: Route Construction and Coverage Assignment.* Let the route  $R_m$  of vehicle  $m$  be a sequence of vertices denoted by  $R_m = \{R_m(0), \dots, R_m(r_m + 1)\}$ , where  $R_m(i)$  represents the  $i$ -th vertex visited by vehicle  $m$ , and  $r_m$  is the number of customer vertices (*i.e.*, in  $V_0$ ) visited. Each route starts and ends at the depot, *i.e.*,  $R_m(0) = R_m(r_m + 1) = v_0$ .

In this second phase, a route is constructed for each vehicle using a greedy insertion strategy guided by the following cost function:

$$g(v, R_m) = \min_{i \in \{0, \dots, r_m\}} \{c_{(u,v)} + c_{(v,w)} - c_{(u,w)} : u = R_m(i), w = R_m(i + 1)\},$$

which represents the minimum incremental cost of inserting vertex  $v$  into route  $R_m$  between two consecutive visited vertices  $u$  and  $w$ .

For each vehicle  $m \in \mathcal{M}$  and each demand vertex  $u \in A_m$ , all unvisited vertices  $v \in C(u)$  are evaluated for possible insertion into route  $R_m$ . The vertex  $v$  with the smallest value of  $g(v, R_m)$  is selected for insertion.

Algorithm 2 presents the construction of routes pseudo-code used in our decoder.

A vertex  $v$  is called *redundant* if its removal from a route does not compromise the feasibility of the solution. In other words,  $v$  is redundant when all the vertices for which  $v$  provides service can be re-assigned to other visited vertices without violating any vehicle capacity constraints. After applying Algorithm 2, the decoder greedily eliminates such redundancies by evaluating the cost decrease associated with removing each candidate vertex, and continues this process until no further removals can be made while maintaining feasibility.

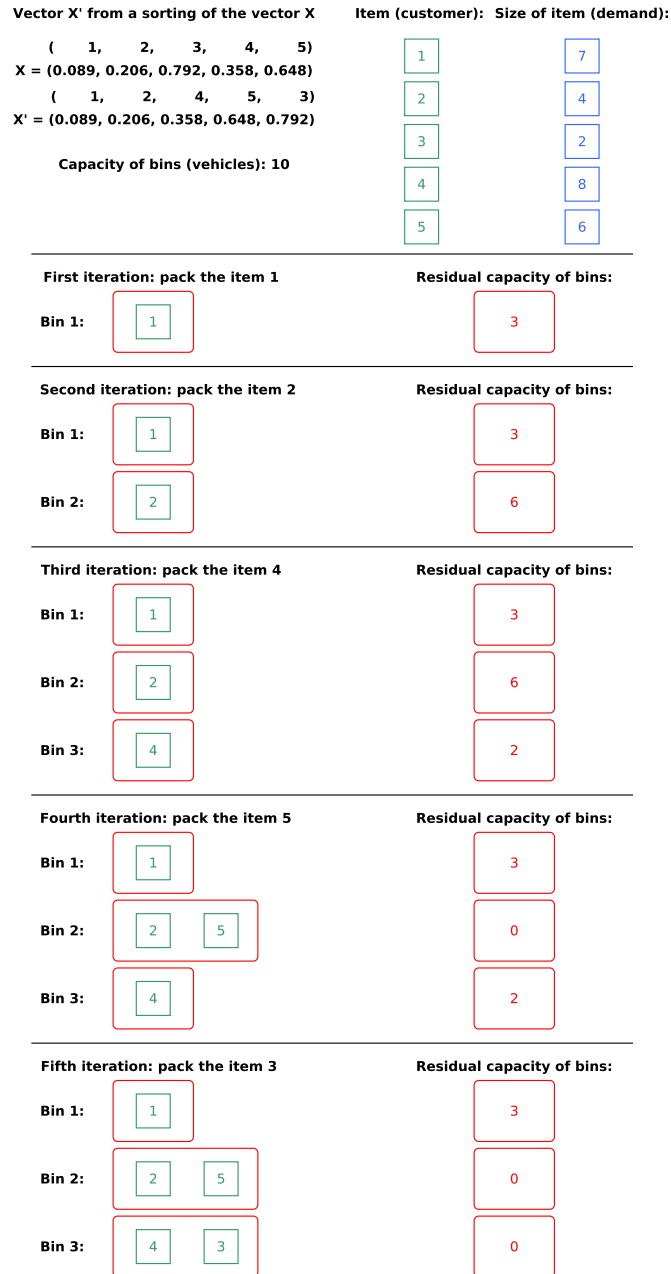


FIGURE 3. Illustration of the Best Fit Algorithm.

---

**Algorithm 2.** Construction of Routes.

---

**Input:** A set of vehicles  $\mathcal{M} = \{1, \dots, M\}$  and their assigned vertices  $\mathcal{A} = \{A_1, \dots, A_M\}$

**Output:** A set of routes  $\mathcal{R} = \{R_1, \dots, R_M\}$

```

1:  $\mathcal{R} \leftarrow \emptyset$ 
2: for each vehicle  $m \in \mathcal{M}$  do
3:   Initialize  $R_m(0) \leftarrow v_0$  Start at the depot
4:    $r_m \leftarrow 0$  ▷ Number of customer vertices in the route
5:   for each demand vertex  $v \in A_m$  do
6:      $S \leftarrow \{u \in C(v) : u \notin R_{m'}(i), \forall m' \in \{1, \dots, m\}, \forall i \in \{1, \dots, r_{m'}\}\}$  ▷ Set of unvisited candidate vertices to cover  $v$ 
7:      $u \leftarrow \arg \min_{u' \in S} g(u', R_m)$  ▷ Select candidate with least insertion cost
8:     insert( $u, R_m$ ) ▷ Insert  $u$  in the best position of  $R_m$ 
9:      $r_m \leftarrow r_m + 1$ 
10:  end for
11:   $R_m(r_m + 1) \leftarrow v_0$  ▷ Return to the depot
12:   $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_m\}$ 
13: end for

```

---

### 3.2. Intra-route intensification

Once the stopping criterion of the BRKGA is met, a final intra-route improvement phase is performed using the Lin–Kernighan (LK) heuristic [28]. The LK heuristic operates on the  $k$ -opt neighborhood, which involves performing up to  $k$  edge exchanges within a route. It is widely regarded as one of the most effective local search methods for the Traveling Salesman Problem (TSP) and is employed here to further reduce the cost of individual routes while preserving feasibility.

### 3.3. Matheuristic approach

Following the approach proposed in [35], this paper introduces a matheuristic for the CCSP based on a set covering formulation with packing constraints. Let  $F$  denote the set of all feasible routes for the CCSP. Define  $a_{if}$  as the coverage matrix, where  $a_{if} = 1$  if and only if demand vertex  $i \in V_d$  is serviced by route  $f \in F$ . Similarly, let  $b_{if}$  be the visiting matrix, where  $b_{if} = 1$  if and only if vertex  $i \in V_0$  is visited by route  $f$ . For each route  $f \in F$ , the binary variable  $\lambda_f$  indicates whether the route is selected in the final solution ( $\lambda_f = 1$ ) or not ( $\lambda_f = 0$ ).

The resulting formulation MIH is as follows:

$$\begin{aligned}
 & \text{(MIH)} \\
 & \text{minimize } \sum_{f \in F} c_f \lambda_f, \tag{15}
 \end{aligned}$$

subject to:

$$\sum_{f \in F} a_{if} \lambda_f \geq 1 \quad \forall i \in V_d, \tag{16}$$

$$\sum_{f \in F} b_{if} \lambda_f \leq 1 \quad \forall i \in V_0, \tag{17}$$

$$\lambda_f \in \{0, 1\} \quad \forall f \in F.$$

The objective function (15) minimizes the total cost of the routes. Constraint set (16) ensures that every vertex in  $V_d$  is serviced by at least one route, while constraint set (17) restricts each vertex in  $V_0$  to be visited at most once. Because the total number of feasible routes ( $F$ ) grows exponentially, we generate a restricted

TABLE 1. BRKGA parameters.

| Parameter             | Value |
|-----------------------|-------|
| Population size       | 512   |
| Elite fraction        | 15%   |
| Mutant fraction       | 15%   |
| Crossover probability | 70%   |

pool of routes, denoted by  $F'$ , for the matheuristic formulation. The construction of  $F'$  proceeds as follows. First, an exhaustive search is performed to identify all optimal routes that service up to three vertices; all such routes are then added to  $F'$ . Next,  $F'$  is supplemented with elite individuals extracted from successive BRKGA generations, starting from the most recent, until either the predetermined size limit for  $F'$  is reached or all elite individuals have been incorporated.

## 4. COMPUTATIONAL EXPERIMENTS

### 4.1. Instances benchmark

The instances for the CCSP were derived from classical CVRP benchmarks proposed in [4, 38], which include between 101 and 303 vertices. These instances are labeled as **E-nA-kB** and **X-nA-kB**, where **A** indicates the total number of vertices (including the depot), and **B** denotes the number of vehicles required to serve all customer demands.

To construct CCSP instances, the following parameters were defined:

- $|V_d|$ : number of vertices with demand;
- $|D(v)|$ : coverage size, where  $v \in V_0$ .

The parameter  $|V_d|$  was varied across three levels: 10%, 20%, and 40% of  $n$ , where  $n$  is the total number of vertices. Demand vertices were selected as the first  $|V_d|$  customer vertices of the CVRP instance (excluding the depot). Importantly, the demand values associated with these vertices remain identical to those in the original CVRP instance. Likewise, the vehicle capacity  $Q$  in the CCSP instance is preserved from the corresponding CVRP input.

For each vertex  $v \in V_0$ , the set  $D(v)$ , which consists of the vertices that can be covered by  $v$ , is defined as the set of its  $k$  nearest neighbors. The parameter  $k$  was set to 7, 9, and 11 to represent different coverage ranges.

For each CVRP instance, all combinations of  $|V_d|$  and  $|D(v)|$  were evaluated, resulting in nine parameter configurations per instance and a total benchmark of 495 CCSP instances. A preprocessing step was applied to each instance to remove any vertex  $v \in V_0$  for which the coverage set satisfies  $D(v) \cap V_d = \emptyset$ , ensuring that only vertices capable of servicing demands are included in the problem.

### 4.2. Computational settings

The ILP formulations were implemented and solved using Gurobi solver [16] version 8.1.1, with a time limit of one hour per instance. All experiments were executed on a PC running Ubuntu 10.12, equipped with an Intel Xeon(R) Silver 3114 CPU at 2.20 GHz and 32 GB of RAM. The BRKGA developed for the CCSP was based on the C++ framework proposed in [37]. The stopping criteria was set to 2000 generations and the parameters values are summarized in Table 1. The implementation of the Lin–Kernighan (LK) heuristic follows the version proposed in [19]. For the matheuristic, the size of the route pool  $F'$  was limited to a maximum of one million routes.

The BRKGA parameters setting was guided by the work of Gonçalves and Resende [15], who originally proposed the metaheuristic and provided recommendations for defining suitable parameters values. In our

implementation, the selected parameters strike a balance between exploration and exploitation. The elite fraction controls the intensity of exploitation by ensuring that the best individuals are preserved and combined more frequently, while the mutant fraction promotes exploration by introducing diversity and preventing premature convergence. The crossover probability further regulates the balance between inheriting features from elite and non-elite individuals. This configuration proved effective in maintaining convergence stability while preserving robustness across different instance sizes.

### 4.3. Results

Four methodologies were implemented and evaluated:

- $\text{CCSP}_1$ : solution of the  $\text{CCSP}_1$  model, initially ignoring Constraints (7) due to their exponential number, and later including them in the formulation using a *lazy constraint* strategy;
- $\text{BRKGA}$ : implementation of the BRKGA for the CCSP described in Section 3;
- $\text{CCSP}_{1s}$ : same as  $\text{CCSP}_1$ , however the solution obtained by  $\text{BRKGA}$  is given as an initial feasible solution (warm start) to the  $\text{CCSP}_1$  model;
- $\text{MIH}$ : solution of the matheuristic described in Section 3.3.

Table 2 presents a comparative summary of the computational results obtained by the four methodologies evaluated in this study: the exact formulation ( $\text{CCSP}_1$ ), the ( $\text{BRKGA}$ ), the warm-started exact formulation ( $\text{CCSP}_{1s}$ ), and the matheuristic approach ( $\text{MIH}$ ). Each row of the table corresponds to a specific instance type, each consisting of nine configurations (instances), and the table reports averages over them. The first two columns report the best upper bound ( $\text{UB}_{\text{best}}$ ) and lower bound ( $\text{LB}_{\text{best}}$ ) achieved across all methods, as well the optimality gap (“gap”), computed as a percentage between the best upper and lower bounds. For  $\text{CCSP}_1$ ,  $\text{BRKGA}$ ,  $\text{CCSP}_{1s}$  and  $\text{MIH}$ , the table reports the average deviation from the best upper bound ( $\Delta\text{UB}$ ), given as a percentage, along with the average runtime (in seconds). Special flags such as “opt” denote proven optimality, “best” indicates the best solution found among all methods, “tle” indicates that the time limit was exceeded before a solution could be found or improved, and “–” means that the exact formulation ( $\text{CCSP}_1$ ) was not able to achieve feasible solutions for all configurations of the specific instance type.

Out of 495 instances, the ILP-based method  $\text{CCSP}_1$  found feasible solutions for 430 instances. Among these, it proved optimality for 71 instances, all with up to 101 vertices. These results highlight  $\text{CCSP}_1$ ’s robustness on smaller instances, where it reliably obtains and proves optimal solutions in an average of 260 s. However, its performance deteriorates significantly for larger instances, where time limits were frequently exceeded and large optimality gaps persisted. The average optimality gap across all instances solved by  $\text{CCSP}_1$  was approximately 43.4%, and many larger instances ( $\text{X-n*-kX}$ ) were left unsolved or had gaps exceeding 50%.

The  $\text{BRKGA}$  proved highly effective and scalable, obtaining feasible solutions for all instances and outperforming the best upper bounds of  $\text{CCSP}_1$  in 407 cases. On average,  $\text{BRKGA}$  solutions showed an 18.35% improvement in cost over  $\text{CCSP}_1$  feasible solutions, with this figure reaching over 25% on larger instances with more than 200 vertices. Notably,  $\text{BRKGA}$  obtained an average deviation from the best upper bound of just 1.30%, confirming its robustness and quality in practice.

The hybrid approaches  $\text{CCSP}_{1s}$  and  $\text{MIH}$  were evaluated for their ability to refine  $\text{BRKGA}$  solutions. The  $\text{CCSP}_{1s}$  method, which uses the  $\text{BRKGA}$  solution as a warm start, improved the initial  $\text{BRKGA}$  solution in 88 instances (18%) with an average cost reduction of 1.69%. The average deviation from the best upper bound was brought down to 0.94%. In contrast, the matheuristic ( $\text{MIH}$ ) demonstrated stronger performance, improving  $\text{BRKGA}$  results in 187 instances (38%) with an average cost reduction of 2.3%. This led to a remarkably low average deviation of 0.28% from the best upper bound. These results confirm that while  $\text{BRKGA}$  provides a strong balance between runtime and quality, the  $\text{MIH}$  was the most effective approach in terms of robustness and overall solution quality across the full range of benchmark instances. This also highlights  $\text{MIH}$ ’s greater effectiveness in post-processing and improving heuristic solutions.

The trends observed in Table 2 indicate that all methods perform comparably on small instances (*e.g.*,  $\text{E-n22-k4}$  to  $\text{E-n51-k5}$ ), with most methods finding optimal solutions quickly. As instance size increases (notably

TABLE 2. Summary of average results comparing the proposed methodologies.

| Instances | Best Bounds        |                    |            | CCSP <sub>1</sub> |            | BRKGA       |        | CCSP <sub>1s</sub> |            | MH          |         |
|-----------|--------------------|--------------------|------------|-------------------|------------|-------------|--------|--------------------|------------|-------------|---------|
|           | UB <sub>best</sub> | LB <sub>best</sub> | gap        | ΔUB               | time       | ΔUB         | time   | ΔUB                | time       | ΔUB         | time    |
| E-n22-k4  | <b>69.33</b>       | 69.33              | <b>opt</b> | <b>opt</b>        | 1.00       | 1.28        | 1.00   | <b>opt</b>         | 1.00       | <b>opt</b>  | 1.00    |
| E-n23-k3  | <b>73.67</b>       | 73.67              | <b>opt</b> | <b>opt</b>        | 1.00       | 0.15        | 1.00   | <b>opt</b>         | 1.00       | <b>opt</b>  | 0.15    |
| E-n30-k3  | <b>100.56</b>      | 100.56             | <b>opt</b> | <b>opt</b>        | 8.33       | 0.11        | 1.00   | <b>opt</b>         | 6.11       | <b>opt</b>  | 0.11    |
| E-n33-k4  | <b>149.22</b>      | 149.22             | <b>opt</b> | <b>opt</b>        | 2.56       | 5.44        | 1.00   | <b>opt</b>         | 2.89       | <b>opt</b>  | 2.98    |
| E-n51-k5  | <b>86.33</b>       | 86.33              | <b>opt</b> | <b>opt</b>        | 29.00      | 1.67        | 11.22  | <b>opt</b>         | 7.11       | <b>opt</b>  | 0.13    |
| E-n76-k7  | 137.19             | 120.69             | 12.03      | 7.65              | 2025.28    | 0.30        | 18.56  | <b>best</b>        | 1907.61    | 0.30        | 15.00   |
| E-n101-k8 | 186.89             | 144.78             | 22.53      | 13.67             | 2631.56    | 0.95        | 55.94  | <b>best</b>        | 3011.89    | 0.51        | 34.06   |
| X-n101-kX | 5318.67            | 4013.44            | 24.54      | –                 | <i>tle</i> | 3.50        | 37.56  | 1.73               | <i>tle</i> | <b>best</b> | 16.67   |
| X-n106-kX | 6641.11            | 4719.78            | 28.93      | 22.86             | <i>tle</i> | <b>best</b> | 96.22  | <b>best</b>        | <i>tle</i> | 0.22        | 1259.67 |
| X-n110-kX | 3334.44            | 2344.44            | 29.69      | 22.11             | <i>tle</i> | 0.21        | 92.11  | <b>best</b>        | <i>tle</i> | 0.43        | 46.22   |
| X-n115-kX | 5626.33            | 3921.67            | 30.30      | 30.05             | <i>tle</i> | 2.82        | 90.22  | 2.26               | <i>tle</i> | <b>best</b> | 41.44   |
| X-n120-kX | 3468.67            | 2213.33            | 36.19      | 22.17             | <i>tle</i> | 0.20        | 6best  | 0.20               | <i>tle</i> | <b>best</b> | 114.22  |
| X-n125-kX | 12 793.78          | 8027.44            | 37.26      | –                 | <i>tle</i> | 0.20        | 69.22  | 0.17               | <i>tle</i> | <b>best</b> | 1503.67 |
| X-n129-kX | 7272.33            | 3968.78            | 45.43      | 36.63             | <i>tle</i> | 0.12        | 66.89  | <b>best</b>        | <i>tle</i> | 0.37        | 49.11   |
| X-n134-kX | 2686.11            | 1466.67            | 45.40      | 25.43             | <i>tle</i> | 0.46        | 88.11  | 0.46               | <i>tle</i> | <b>best</b> | 67.22   |
| X-n139-kX | 3462.89            | 1952.89            | 43.61      | 18.11             | <i>tle</i> | 0.19        | 139.22 | 0.19               | <i>tle</i> | <b>best</b> | 70.78   |
| X-n143-kX | 4228.22            | 2399.22            | 43.26      | 19.30             | <i>tle</i> | 0.30        | 212.67 | 0.30               | <i>tle</i> | <b>best</b> | 913.44  |
| X-n148-kX | 9053.78            | 5194.56            | 42.63      | –                 | <i>tle</i> | 2.05        | 93.11  | 2.05               | <i>tle</i> | <b>best</b> | 39.89   |
| X-n153-kX | 12 759.89          | 7992.56            | 37.36      | –                 | <i>tle</i> | 1.99        | 109.44 | 1.75               | <i>tle</i> | <b>best</b> | 23.78   |
| X-n157-kX | 4118.33            | 2135.78            | 48.14      | 10.58             | <i>tle</i> | 0.43        | 86.44  | 0.40               | <i>tle</i> | <b>best</b> | 792.22  |
| X-n162-kX | 3774.67            | 2245.78            | 40.50      | 15.96             | <i>tle</i> | 0.11        | 84.89  | 0.01               | <i>tle</i> | <b>best</b> | 68.89   |
| X-n167-kX | 5236.33            | 2294.11            | 56.19      | 20.58             | <i>tle</i> | 0.27        | 262.89 | 0.27               | <i>tle</i> | <b>best</b> | 186.78  |
| X-n172-kX | 8864.56            | 5918.89            | 33.23      | –                 | <i>tle</i> | 5.15        | 104.67 | 5.14               | <i>tle</i> | <b>best</b> | 54.44   |
| X-n176-kX | 30 263.00          | 12 449.22          | 58.86      | –                 | <i>tle</i> | 0.49        | 108.78 | 0.49               | <i>tle</i> | <b>best</b> | 23.56   |
| X-n181-kX | 6118.33            | 3778.00            | 38.25      | 38.02             | <i>tle</i> | 0.11        | 133.78 | <b>best</b>        | <i>tle</i> | 0.28        | 528.33  |
| X-n186-kX | 5970.00            | 2378.00            | 60.17      | 40.19             | <i>tle</i> | 0.09        | 234.33 | <b>best</b>        | <i>tle</i> | 0.33        | 186.33  |
| X-n190-kX | 4185.11            | 2342.56            | 44.03      | 21.39             | <i>tle</i> | <b>best</b> | 168.11 | <b>best</b>        | <i>tle</i> | 0.81        | 2022.44 |
| X-n195-kX | 9875.11            | 6972.56            | 29.39      | –                 | <i>tle</i> | 8.37        | 95.11  | 8.27               | <i>tle</i> | <b>best</b> | 50.11   |
| X-n200-kX | 13 256.44          | 6547.00            | 50.61      | –                 | <i>tle</i> | 0.37        | 129.44 | 0.32               | <i>tle</i> | <b>best</b> | 1298.56 |
| X-n204-kX | 5130.67            | 3006.67            | 41.40      | 29.61             | <i>tle</i> | <b>best</b> | 193.33 | <b>best</b>        | <i>tle</i> | 0.82        | 143.44  |
| X-n209-kX | 7918.11            | 3302.67            | 58.29      | 33.54             | <i>tle</i> | 0.46        | 157.44 | 0.46               | <i>tle</i> | <b>best</b> | 1305.44 |
| X-n214-kX | 2594.33            | 1307.33            | 49.61      | 31.36             | <i>tle</i> | 0.12        | 293.33 | <b>best</b>        | <i>tle</i> | 0.06        | 1290.22 |
| X-n219-kX | 26 113.78          | 10 117.22          | 61.26      | –                 | <i>tle</i> | 0.98        | 106.33 | 0.98               | <i>tle</i> | <b>best</b> | 26.67   |
| X-n223-kX | 9443.78            | 3731.89            | 60.48      | –                 | <i>tle</i> | 0.69        | 176.00 | 0.61               | <i>tle</i> | <b>best</b> | 1181.22 |
| X-n228-kX | 14 998.56          | 6250.89            | 58.32      | –                 | <i>tle</i> | 3.70        | 185.89 | 3.39               | <i>tle</i> | <b>best</b> | 319.11  |
| X-n233-kX | 6075.89            | 2784.44            | 54.17      | 31.20             | <i>tle</i> | 0.09        | 158.22 | <b>best</b>        | <i>tle</i> | 1.56        | 250.56  |
| X-n237-kX | 7014.44            | 2784.78            | 60.30      | 51.72             | <i>tle</i> | <b>best</b> | 286.56 | <b>best</b>        | <i>tle</i> | 0.30        | 1369.78 |
| X-n242-kX | 19 172.89          | 6268.33            | 67.31      | –                 | <i>tle</i> | 1.88        | 222.00 | 1.87               | <i>tle</i> | <b>best</b> | 1006.00 |
| X-n247-kX | 21 817.89          | 11 086.78          | 49.18      | –                 | <i>tle</i> | 1.33        | 143.78 | 1.28               | <i>tle</i> | <b>best</b> | 19.56   |
| X-n251-kX | 8917.67            | 4008.33            | 55.05      | 42.20             | <i>tle</i> | 1.19        | 276.44 | 1.00               | <i>tle</i> | <b>best</b> | 1002.11 |
| X-n256-kX | 5357.11            | 3198.56            | 40.29      | 21.09             | <i>tle</i> | 2.20        | 385.00 | <b>best</b>        | <i>tle</i> | 2.91        | 277.44  |
| X-n261-kX | 7752.78            | 2775.78            | 64.20      | 37.95             | <i>tle</i> | 0.36        | 526.00 | 0.36               | <i>tle</i> | <b>best</b> | 1187.22 |
| X-n266-kX | 20 923.11          | 6532.00            | 68.78      | –                 | <i>tle</i> | 4.42        | 167.89 | 4.28               | <i>tle</i> | <b>best</b> | 693.44  |
| X-n270-kX | 8454.44            | 3634.56            | 57.01      | 41.57             | <i>tle</i> | 0.62        | 164.44 | 0.03               | <i>tle</i> | <b>best</b> | 1239.89 |
| X-n275-kX | 5131.78            | 2160.44            | 57.90      | 55.03             | <i>tle</i> | 1.29        | 308.33 | 0.22               | <i>tle</i> | <b>best</b> | 307.44  |
| X-n280-kX | 18 887.33          | 5741.56            | 69.60      | –                 | <i>tle</i> | 1.12        | 359.89 | 1.12               | <i>tle</i> | <b>best</b> | 1263.11 |
| X-n284-kX | 5521.78            | 2825.44            | 48.83      | 37.79             | <i>tle</i> | <b>best</b> | 491.44 | <b>best</b>        | <i>tle</i> | 0.25        | 1363.00 |
| X-n289-kX | 20 812.11          | 6080.89            | 70.78      | –                 | <i>tle</i> | 1.76        | 231.89 | 1.73               | <i>tle</i> | <b>best</b> | 1319.22 |
| X-n294-kX | 11 565.67          | 4337.11            | 62.50      | –                 | <i>tle</i> | 5.55        | 263.44 | 5.55               | <i>tle</i> | <b>best</b> | 1253.22 |
| X-n298-kX | 9692.56            | 3603.56            | 62.82      | 45.99             | <i>tle</i> | 1.16        | 306.44 | 0.96               | <i>tle</i> | <b>best</b> | 838.89  |
| X-n303-kX | 6109.56            | 2535.89            | 58.49      | 33.06             | <i>tle</i> | 0.12        | 367.11 | <b>best</b>        | <i>tle</i> | 1.52        | 410.33  |
| Overall   |                    |                    | 43.43      |                   | 25.20      |             | 1.30   |                    | 0.94       |             | 0.28    |

**Notes.** Instances: each line represents a group of nine instances with different configurations over the same graph. Best Bounds: best upper bound (UB<sub>best</sub>) and lower bound (LB<sub>best</sub>) obtained among all methodologies. *gap*: optimality gap calculated as  $gap = 100 \frac{UB-LB}{UB}$ . ΔUB: deviation from UB<sub>best</sub> calculated as  $\Delta UB = 100 \frac{UB-UB_{best}}{UB_{best}}$  *time*: execution time in seconds. *tle*: time limit exceeded (one hour). **opt**: optimal solution obtained. **best**: best solution obtained among all methodologies.

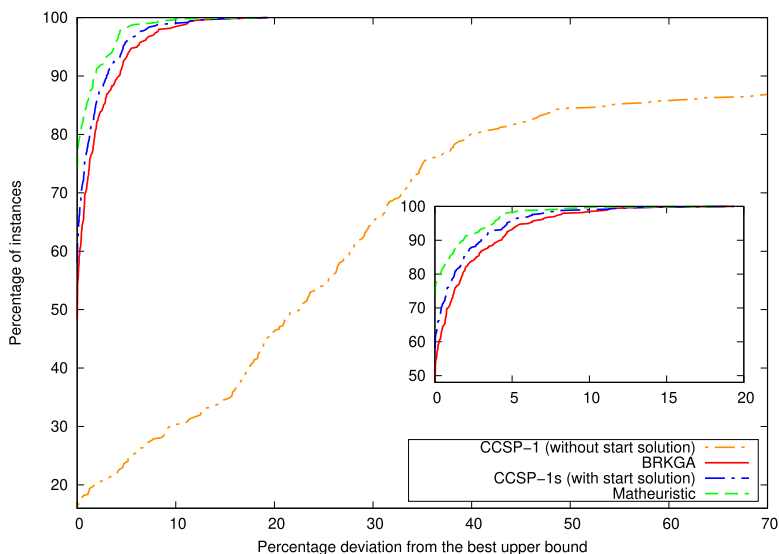


FIGURE 4. Performance profiles [10] in terms of deviation from the best upper bound.

in the  $X-n*-kX$  group),  $CCSP_1$  struggles to scale, often exceeds the 3600s time limit on larger instances without producing a feasible solution. While warm-starting *via*  $CCSP_{1s}$  avoids this in some cases, it still frequently fails to improve upon the  $BRKGA$  solution. By contrast,  $BRKGA$  maintains moderate runtime even for large instances, usually completing within a few minutes.  $MIH$  requires additional time to perform inter-route optimization but remains within reasonable bounds, typically under 30 min.

Performance profiles in Figure 4 illustrate the percentage of instances solved within a given deviation from the best-known upper bound. The profiles show the relative strength of the methods in terms of robustness. On one hand,  $BRKGA$  dominates  $CCSP_1$  across all deviation thresholds, producing the best-known solution for approximately 48% of instances. On the other hand,  $CCSP_{1s}$  and  $MIH$  progressively outperform  $BRKGA$  as the deviation narrows. Specifically,  $CCSP_{1s}$  achieves best-known solutions for 58% of instances, and  $MIH$  outperforms all other methods, attaining the best solutions for 74% of instances. The profiles highlight the consistent performance of  $BRKGA$  and the superior improvement capability of  $MIH$ .

In conclusion, the  $MIH$  approach offers the best trade-off between solution quality and runtime, outperforming all other methods in the majority of cases. The  $BRKGA$  alone provides a solid baseline, especially where exact methods are infeasible. The results strongly support hybrid approaches that combine metaheuristics with exact components to address the computational challenges of the  $CCSP$  across a wide range of instance sizes.

## 5. FINAL REMARKS

In this work, we introduced the *Capacitated Covering Salesman Problem (CCSP)*, a new routing variant that integrates the concept of coverage service into the classical capacitated vehicle routing framework. To solve this challenging problem, we developed an ILP formulation and a dedicated  $BRKGA$  metaheuristic. We also contributed a benchmark library of 495  $CCSP$  instances derived from established  $CVRP$  datasets to rigorously evaluate our approaches. The computational results demonstrate the effectiveness of the proposed methods. The ILP formulation  $CCSP_1$  found proven optimal solutions for 71 out of 198 tested instances, including scenarios with up to 101 vertices, before reaching the time limit on the larger cases. The  $BRKGA$  heuristic consistently produced high-quality solutions for all instances, often outperforming the best incumbent solutions obtained by

the exact method. This confirms that our metaheuristic can efficiently handle larger problem sizes and provide high-quality solutions when exact optimization becomes impractical.

Further solution quality improvements were achieved by combining the strengths of the exact and heuristic approaches. In particular, seeding  $\text{CCSP}_1$  with the best BRKGA solution as a warm start ( $\text{CCSP}_{1s}$ ) enabled the exact solver to improve upon the heuristic routes in several instances, and a tailored matheuristic that performs inter-route exchange intensification yielded even better solutions. These hybrid strategies led to additional cost reductions beyond the standalone BRKGA, showing the value of integration between heuristic and exact techniques for the CCSP. Overall, our study provides a comprehensive framework for the CCSP: a rigorous mathematical formulation, an effective metaheuristic, and an intensification scheme, all validated on a diverse set of benchmark instances. These contributions establish a solid foundation for tackling coverage-based vehicle routing problems and illustrate that the coverage paradigm can be successfully incorporated into capacitated routing with modern optimization tools.

A promising avenue for future research is to strengthen the ILP model with additional valid inequalities or to develop a branch-and-cut algorithm, which could improve the solvability of larger instances and close the optimality gaps faster. Another worthwhile extension is to consider a multi-objective version of the CCSP in which the coverage radius is treated as a secondary objective to be minimized (*e.g.*, balancing travel costs against service range). Such an extension would address the trade-off between route efficiency and coverage breadth, providing deeper insight into the value of flexible service coverage.

#### FUNDING

This work was supported by CNPq (Grant 311907/2023-7).

#### DATA AVAILABILITY STATEMENT

The experimental data, source codes, and results that support the findings of this study are publicly available at the following site: <http://www.ic.unicamp.br/~fusberti/problems/ccsp>.

#### REFERENCES

- [1] L.R. Abreu, R.F. Tavares-Neto and M.S. Nagano, A new efficient biased random key genetic algorithm for open shop scheduling with routing by capacitated single vehicle and makespan minimization. *Eng. App. Artif. Intell.* **104** (2021) 104373.
- [2] S. Allahyari, M. Salari and D. Vigo, A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *Eur. J. Oper. Res.* **242** (2015) 756–768.
- [3] D.L. Applegate, R.E. Bixby, V. Chvatal and W.J. Cook, The Traveling Salesman Problem: A Computational Study. *Princeton Series in Applied Mathematics*. Princeton University Press, Princeton, NJ, USA (2007).
- [4] N. Christofides and S. Eilon, An algorithm for the vehicle-dispatching problem. *J. Oper. Res. Soc.* **20** (1969) 309–318.
- [5] J.-F. Cordeau, G. Laporte, M.W.P. Savelsbergh and D. Vigo, Vehicle routing. *Handb. Oper. Res. Manage. Sci.* **14** (2007) 367–428.
- [6] J.R. Current and D.A. Schilling, The covering salesman problem. *Transp. Sci.* **23** (1989) 208–213.
- [7] G.B. Dantzig and J.H. Ramser, The truck dispatching problem. *Manage. Sci.* **6** (1959) 80–91.
- [8] R. De la Fuente, M.M. Aguayo and C. Contreras-Bolton, An optimization-based approach for an integrated forest fire monitoring system with multiple technologies and surveillance drones. *Eur. J. Oper. Res.* **313** (2024) 435–451.
- [9] 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*. Springer US, Boston, MA (2008) 527–550.
- [10] E.D. Dolan and J.J. Moré, Benchmarking optimization software with performance profiles. *Math. Prog.* **91** (2002) 201–213.
- [11] S. Domínguez-Casasola, J.L. González-Velarde, Y.Á. Ríos-Solís and K.A. Reyes-Vega, The capacitated family traveling salesperson problem. *Int. Trans. Oper. Res.* **31** (2024) 2123–2153.
- [12] M. Gendreau, G. Laporte and F. Semet, The covering tour problem. *Oper. Res.* **45** (1997) 568–576.

- [13] B.L. Golden, S. Raghavan and E.A. Wasil, The Vehicle Routing Problem: Latest Advances and New Challenges. Vol. 43 of *Operations Research/Computer Science Interfaces Series*. Springer, Boston, MA (2008).
- [14] B. Golden, Z. Najj-Azimi, S. Raghavan, M. Salari and P. Toth, The generalized covering salesman problem. *INFORMS J. Comput.* **24** (2012) 534–553.
- [15] J.F. Gonçalves and M.G.C. Resende, Biased random-key genetic algorithms for combinatorial optimization. *J. Heuristics* **17** (2011) 487–525.
- [16] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual (2025).
- [17] M.H. Ha, N. Bostel, A. Langevin and L.-M. Rousseau, 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.
- [18] 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.
- [19] K. Helsgaun, An effective implementation of the Lin–Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* **126** (2000) 106–130.
- [20] M.J. Hodgson, G. Laporte and F. Semet, A covering tour model for planning mobile health care facilities in suhumdistrict, Ghama. *J. Reg. Sci.* **38** (1998) 621–638.
- [21] L. Jiao, Z. Peng, L. Xi, M. Guo, S. Ding and Y. Wei, A multi-stage heuristic algorithm based on task grouping for vehicle routing problem with energy constraint in disasters. *Expert Syst. App.* **212** (2023) 118740.
- [22] N. Jozefowicz, A branch-and-price algorithm for the multivehicle covering tour problem. *Networks* **64** (2014) 160–168.
- [23] M. Kammoun, H. Derbel, M. Ratli and B. Jarboui, An integration of mixed VND and VNS: the case of the multivehicle covering tour problem. *Int. Trans. Oper. Res.* **24** (2017) 663–679.
- [24] H. Kellerer, U. Pferschy and D. Pisinger, Knapsack Problems. Vol. 1. Springer-Verlag Berlin Heidelberg (2004).
- [25] K. Kılıç, M. Meterelliyoç, İ. Güvenç Pelit and M. Soysal, Modeling a humanitarian-aid covering tour problem with location selection and vehicle assignment decisions. *Int. Trans. Oper. Res.* (2025). DOI: [10.1111/itor.70088](https://doi.org/10.1111/itor.70088).
- [26] A.F. Kummer N, L.S. Buriol and O.C.B. de Araújo, A biased random key genetic algorithm applied to the VRPTW with skill requirements and synchronization constraints, in Proceedings of the 2020 Genetic and Evolutionary Computation Conference (2020) 717–724.
- [27] A.F. Kummer, O.C.B. de Araújo, L.S. Buriol and M.G.C. Resende, A biased random-key genetic algorithm for the home health care problem. *Int. Trans. Oper. Res.* **31** (2024) 1859–1889.
- [28] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.* **21** (1973) 498–516.
- [29] W. Liu, M. Dridi, H. Fei and A.H. El Hassani, Solving a multi-period home health care routing and scheduling problem using an efficient matheuristic. *Comput. Ind. Eng.* **162** (2021) 107721.
- [30] S. Maheshwari, P.K. Jain and K. Kotecha, Route optimization of mobile medical unit with reinforcement learning. *Sustainability* **15** (2023) 3937.
- [31] K. Murakami, A column generation approach for the multi-vehicle covering tour problem, in 2014 IEEE International Conference on Automation Science and Engineering (CASE). IEEE (2014) 1063–1068.
- [32] Z. Najj-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.
- [33] E. Ruiz, V. Soto-Mendoza, A. Ernesto Ruiz Barbosa and R. Reyes, Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. *Comput. Ind. Eng.* **133** (2019) 207–219.
- [34] R. Santa González, M. Cherklesly, T. Gabriel Crainic and M.-È. Rancourt, Multi-period location routing: an application to the planning of mobile clinic operations in Iraq. *Comput. Oper. Res.* **159** (2023) 106288.
- [35] C.S. Sartori and L.S. Buriol, A matheuristic approach to the pickup and delivery problem with time windows, in International Conference on Computational Logistics. Springer (2018) 253–267.
- [36] F.A. Tillman, The multiple terminal delivery problem with probabilistic demands. *Transp. Sci.* **3** (1969) 192–204.
- [37] R.F. Toso and M.G.C. Resende, A C++ application programming interface for biased random-key genetic algorithms. *Optim. Methods Softw.* **30** (2015) 81–93.
- [38] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal and A. Subramanian, New benchmark instances for the capacitated vehicle routing problem. *Eur. J. Oper. Res.* **257** (2017) 845–858.

- [39] J. Zhao, Y. Long, B. Xie, G. Xu and Y. Liu, A matheuristic solution for efficient scheduling in dynamic truck–drone collaboration. *Expert Syst. App.* **267** (2025) 126218.



**Please help to maintain this journal in open access!**

This journal is currently published in open access under the Subscribe to Open model (S2O). We are thankful to our subscribers and supporters for making it possible to publish this journal in open access in the current year, free of charge for authors and readers.

Check with your library that it subscribes to the journal, or consider making a personal donation to the S2O programme by contacting [subscribers@edpsciences.org](mailto:subscribers@edpsciences.org).

More information, including a list of supporters and financial transparency reports, is available at <https://edpsciences.org/en/subscribe-to-open-s2o>.