Optimization Letters (2024) 18:1053–1069
https://doi.org/10.1007/s11590-023-02072-y

ORIGINAL PAPER

On the exact solution of the multi-depot open vehicle routing problem

Vinícius Carvalho Soares[1] · Marcos Roboredo[2]

Received: 7 July 2022 / Accepted: 26 September 2023 / Published online: 23 October 2023

Abstract
The multi-depot open vehicle routing problem (MDOVRP) is a variant of the classical capacitated vehicle routing problem, where the vehicles can depart from different depots and they do not need to go back to the depot at the end of the route. This paper presents an exact branch-cut-and-price algorithm for the problem that is coded within the VRPSolver framework. In order to prove the robustness of the proposed method, we present several computational experiments over 24 MDOVRP benchmark instances. The results allow us to draw two important conclusions. The first one is that the proposed algorithm solves to the optimality all literature instances including eight that were open so far while the second one is that our algorithm outperforms the former state-of-art. Besides, we propose 440 MDOVRP instances with up to 200 nodes, where 418 of them are solved to the optimality by the proposed method. Another contribution of this paper is to show how to adapt the proposed method for the MDOVRP variant with time windows constraints (MDOVRPTW). We also apply our approach to MDOVRP and MDOVRPTW variants with a single depot. The results for all variants show that our method find the optimal solution for almost all tested instances in reasonable computational times.

Keywords Open vehicle routing · Multi-depot · Branch-and-cut-and-price

✉ Marcos Roboredo
mcroboredo@id.uff.br

Vinícius Carvalho Soares
vinicius_soares@id.uff.br

[1] Pós-Graduação em Engenharia de Produção, Universidade Federal Fluminense, Niterói, Brazil

[2] Departamento de Engenharia de Produção, Universidade Federal Fluminense, Niterói, Brazil

1 Introduction

The vehicle routing problem (VRP) consists of designing a set of minimal cost routes for a set o vehicles aiming to serve the demand of several customers. That problem was presented in [5]. Since then, several variants have been proposed involving different characteristics such as multiple depots, heterogeneous fleet, time windows, among others.

This paper deals with the multi-depot open vehicle routing problem (MDOVRP) that is formally defined as follows. Given an undirected graph $G = (V, A)$. The set of vertices $V = \{1, \dots, n + m\}$ is partitioned into a subset of customers $V^+ = \{1, \dots, n\}$ and a subset of depots $D = \{n + 1, \dots, n + m\}$. The set of arcs A is defined as $A = \{(i,j)|i,j \in V^+, i \neq j\} \cup \{(k,i)|k \in D, i \in V^+\}$. For each customer $i \in V^+$, a positive demand w_i is given. We assume $w_k = 0$ for each $k \in D$. For each arc $a = (i,j) \in A$, a positive travel cost c_{ij} is given. To serve the customers, there is a homogeneous fleet of vehicles, each one with capacity Q. The problem consists of designing a set of routes minimizing the total travel cost where the designed routes satisfy the following constraints: each route starts at one of the depots and ends at one of the customers; each customer is visited exactly once in a single route; the sum of the demands of the visited customers in a given route does not exceed the vehicle capacity Q.

The MDOVRP was presented in [17]. The authors presented a stochastic search meta-heuristic algorithm termed as the list-based threshold accepting (LBTA) algorithm. Since then, several methodologies has been proposed for the problem, where we can highlight the ones proposed in [6–8, 10, 13, 15, 19].

To the best of our knowledge, the best exact approach for the MDOVRP is the Mixed Integer Linear Programming (MILP) formulation with two indexes proposed in [8]. That formulation was able to solve to the optimality several instances with up to 240 customers and outperformed the other two existing exact formulations proposed in [7, 10]. Despite the good results presented in [8], 8 out of 24 instances remained open after that research.

This paper proposes an exact branch-and-cut-and-price (BCP) algorithm for the MDOVRP that is coded within the VRPSolver framework [11]. The proposed algorithm contains the main state-of-art BCP elements such as automatic stabilization by dual price smoothing, limited-memory rank-1 cuts, reduced cost-based arc elimination, enumeration of elementary routes, and hierarchical strong branching.

Another contribution of this paper is to show how to adapt the proposed model to the MDOVRP variant that considers time windows constraints so-called Multi-Depot Open Vehicle Routing with Time Windows (MDOVRPTW). To the best of our knowledge the MDOVRPTW that we deal with in this paper was studied only in [1] where the authors proposed a heuristic algorithm approach.

We report several computational experiments over the benchmark set of 24 MDOVRP instances with up to 288 customers frequently used by the literature about this problem. The results show that our method outperforms the MILP formulation proposed in [8]. Moreover, our method solves to the optimality all instances including eight that were open so far. For the MDOVRPTW, we test the proposed method on 20 benchmark instances and the results show that our method solved to the optimality all of them in reasonable computational times. We also test the proposed approach on benchmark instances of the Open Vehicle Routing Problem (OVRP) and Open Vehicle Routing Problem With Time Windows (OVRPTW), which are particular cases of respectively MDOVRP and MDOVRPTW when there is a single depot. The results on OVRP and OVRPTW literature instances show that just few instances were not solved to the optimality by our method.

The remainder of this paper is organized as follows. Section 2 presents an overview of the VRPSolver framework. Section 3 presents the VRPSolver model proposed in this paper for the MDOVRP. Instances and the results are discussed in Sect. 4. Finally, Sect. 5 presents our conclusions and suggestions for future researches.

2 Overview of the VRPSolver framework

We present in this section an overview of the VRPSolver framework in order to make this paper self-contained.

The VRPSolver solves a Mixed Integer Linear Programming (MILP) formulation that contains variables associated with resource constrained paths in one or several directed graphs. That formulation is solved by a generic BCP algorithm where the path variables are generated on demand by solving the pricing subproblems that are Resource Constrained Shortest Path Problems (RCSPP). The so-called VRPSolver model is composed of the MILP formulation and all graph information that are provided by the framework user and have to follow some generic structures that we describe now. We provide simpler versions of those structures that are enough to understand the proposed VRPSolver model. For a detailed description about those structures, we refer to [11].

2.1 Generic path generator graphs

One of the VRPSolver framework user modelling tasks are to define one or more directed graphs, as well all the information necessary to obtain resource constrained paths on them that follow the generic structures described now. Let K be the finite set of graphs defined by the user. The following elements must to be defined by the modeller for each graph $G^k = (V^k, A^k) \in K$: two vertices v^k_{source} and v^k_{sink} in V^k to indicate where each path should start and end respectively. v^k_{source} and v^k_{sink} can be even the same vertex; a set of resources R^k; a resource consumption of the resource r on the arc a denoted by q^r_a, for each $a \in A^k$ and $r \in R^k$; an interval $[l_{v,r}, u_{v,r}]$, for each $r \in R^k$ and $v \in V^k$.

For a given graph $G^k \in K$, we say that a path $p = (v^k_{source} = v_0, a_1, v_1, \dots, a_n, v_n = v^k_{sink})$ over G^k is resource constrained if the following constraints are satisfied: $n \geq 1$; $v_j \neq v^k_{source}$ and $v_j \neq v^k_{sink}$, $1 \leq j \leq n-1$; For a given resource r, the total consumption of r when the vertex v_j is visited denoted by S_{jr} is within the interval $[l_{v_j,r}, u_{v_j,r}]$, $0 \leq j \leq n$, where S_{jr} is calculated in the following way: $S_{j,r} = 0$ if $j = 0$. Otherwise, we have two possibilities: $S_{j,r} = S_{j-1,r} + q^r_{a_j}$ if $S_{j-1,r} + q^r_{a_j} \geq l_{v_j,r}$ and $S_{j,r} = l_{v_j,r}$ otherwise. In other words, the consumption of the resource is null at the beginning of the path and it increases as the arcs are being used by the path. For VRP variants, it is very common that each resource constrained path represents a route for the problem. In these cases, the use of resources can be very useful to model classical constraints such as capacity and time windows ones.

2.2 Generic formulation

The VRPSolver user also defines a formulation that must be encompassed by the generic one described in this section. For each graph $G^k \in K$, let P^k be the set of all resource constrained paths on G^k. Let also $P = \bigcup_{k \in K} P^k$. For each path $p \in P$ and each arc $a \in \cup_{k \in K} A^k$, the constant h^p_a computes how many times the arc a appears in the path p. The master formulation uses generic integer variables x_j, $1 \leq j \leq n_1$, where n_1 is the number of variables x. For VRPs, is very common to define a variable x for each arc that can be traveled by the routes indicating how many times this arc is traveled. In this case, we would have n_1 equal to the number of arcs. The formulation also uses a variable λ_p, $p \in P$, to indicate how many times the path p appears in the optimal solution. Finally, the formulation uses generic constants $m \in \mathbb{Z}_+, c' \in \mathbb{R}^{n_1}, \alpha \in \mathbb{R}^{m \times n_1}$ and $d \in \mathbb{R}^m$. The generic Mixed Integer Linear Programming (MILP) formulation follows.

$$\text{Min} \sum_{j=1}^{n_1} c'_j x_j, \tag{1a}$$

$$\text{S.t.} \sum_{j=1}^{n_1} \alpha_{ij} x_j \geq d_i, \quad \forall i = 1, \dots, m, \tag{1b}$$

$$x_j = \sum_{p \in P} \left(\sum_{a \in M(x_j)} h^p_a \right) \lambda_p, \quad \forall j = 1, \dots, n_1, \tag{1c}$$

$$L^k \leq \sum_{p \in P^k} \lambda_p \leq U^k, \quad \forall k \in K, \tag{1d}$$

$$\lambda_p \in \mathbb{Z}_+, \quad \forall p \in P, \tag{1e}$$

$$x_a \in \mathbb{Z}, \quad \forall a \in A. \tag{1f}$$

The MILP Formulation (1) contains a very generic objective function (1a) and the very generic constraints (1b). The constraints (1c) show the relation between the variables x and λ. That relation is based on the so-called Mapping sets M that is defined for each variable x_j in a way that $M(x_j) \subseteq \cup_{k\in K} A^k$. The definition of a proper function M is an important user modelling task. The constraints (1d) present, for each graph $G^k \in K$, a lower and an upper bounds for the number of paths on that graph. Such bounds are also defined by the framework user.

2.3 Solving a specific problem through VRPSolver framework

The idea behind the VRPSolver framework is that if the user is able to model a problem by defining one or more resource constrained graphs and a MILP formulation within the generic characteristics described respectively in Sect. 2.1 and in the formulation (1) then the framework solves the formulation (1) through a generic BCP algorithm. In Sects. 3.1 and 3.2 we present the graph and the formulation that defines the proposed VRPSolver model for the MDOVRP.

The master formulation for the BCP algorithm solved by the framework is the linear programming relaxation of (1) without the variables x that are eliminated through the constraints (1c). The master formulation is then solved by an iterative procedure based on column generation algorithm. In each iteration, a restricted version of the formulation with a subset of variables λ is solved. Variables λ with negative reduced cost are then added to the restricted master formulation. The framework finds such variables by solving RCSPP over the provided resource constrained path generator graphs. If there is no variable with negative reduced cost then the current solution is optimal for the master formulation.

In every node of the branch-and-bound search tree, the master formulation is solved by the BCP algorithm. Branching on variables x are then used to solve MILP formulation (1) to optimality.

2.4 Using state-of-art BCP algorithm components

When the user defines the so-called packing sets for a VRPSolver model, then the main state-of-art BCP algorithms components can be activated such as ng-path relaxation, rounded capacity cuts separators, limited-memory rank-1 cutting planes, and elementary route enumeration.

The packing sets can be defined on vertices or arcs. In this paper we present the definition for vertices. Let $\mathcal{P} \subset 2^{V'}$ be a collection of mutually disjoint subsets of V', where $V' = \bigcup_{k\in K} V^k \setminus \{v^k_{source}, v^k_{sink}\}$. We say that the subsets of $\mathcal{P}$ are packing sets if there is at least one optimal solution for the master problem satisfying the constraints (2), where each h^p_v computes the number of times that a vertex v is visited in a path p.

$$\sum_{p\in P}\left(\sum_{v\in S} h_v^p\right)\lambda_p \le 1, \quad \forall S \in \mathcal{P} \tag{2}$$

According to constraints (2), for a given packing set $S \in \mathcal{P}$, at most one vertex of S appears in the optimal solution and at most one time. The definition of proper packing sets is an important modelling task of the framework user. In VRPs, commonly the routes are modeled as paths on the graphs and the customers are modeled as packing sets.

The use of packing sets on vertices also allows the framework user to use some state-of-art BCP components, such as *ng*-path relaxation, generation of rounded capacity and limited-memory rank-1 cuts, and route enumeration. For more details about it, we refer to [11]. Besides, the user can define Rounded Capacity Cuts (RCCs) separator. To add a RCC separator, the modeler has to define a capacity $\mathcal{Q}$ and a demand function $d : \mathcal{P} \cup \emptyset \to \mathbb{R}_+$ such that $d(\emptyset) = 0$ and such taht there is an optimal solution (x^*, y^*, λ^*) of Formulation (1) such the following conditions are satisfied:

1. $\sum_{v\in p} d(S(v)) \le \mathcal{Q}, \forall p \in P$, where $S(v)$ is the packing set that contains v ($S(v) = \emptyset$ if v does not belong to any packing set).
2. for all $S \in \mathcal{P}$ such that $d(S) > 0$, the corresponding constraints in (2) should be satisfied with equality by (x^*, y^*, λ^*).

Based on the function d and capacity $\mathcal{Q}$ provided by the user, the following valid inequality represents a Rounded Capacity Cut for a given $\mathcal{S} \subseteq \mathcal{P}$:

$$\sum_{p\in P} h_{\mathcal{S}}^p \lambda_p \ge \left\lceil \frac{\sum_{S\in\mathcal{S}} d(S)}{\mathcal{Q}} \right\rceil, \tag{3}$$

where $h_{\mathcal{S}}^p$ is the number of times that an arc in path $p \in P$ enters in $\mathcal{S}$. The definition of proper and valid RCC separators is a modelling user task. For classical capacity constraint that is considered by the MDOVRP and MDOVRPTW, a RCC separator can be defined by associating the function d with the demand of the customers and by considering $\mathcal{Q}$ equal to the vehicle capactity. This kind of RCC separator is defined for several VRPSolver models in [11].

3 A VRPSolver model for the MDOVRP

In this section, we present the proposed VRPSolver model for the MDOVRP according to the notation described in Sect. 2.

3.1 Path generator graph for the proposed VRPSolver model

We define a path generator graph $G^1 = (V^1, A^1)$ with the following characteristics: $V^1 = \{v_i | i \in V \cup \{0\}\}$ and $A^1 = \{(v_0, v_k) | k \in D\} \cup \{(v_k, v_i) | k \in D, i \in V^+\} \cup \{(v_i, v_j), (v_j, v_i) | i, j \in V^+\} \cup \{(v_i, v_0) | i \in V^+\}$; Each path starts and ends at v_1^0 ($v^1_{source} = v^1_{sink} = v_0$); $R^1 = \{r\}$, where r is a resource created in order to ensure the capacity constraint for each vehicle; $q^r_a = (w_i + w_j)/2, a = (v_i, v_j) \in A^1$ (we consider $w_0 = 0$); $[l^r_i, u^r_i] = [0, Q], i \in V^1$.

The idea behind the graph G^1 is that each resource constrained path is associated with a route for the MDOVRP. The vertex v_0 represents the beginning and ending of the paths but this vertex does not represent a vertex of the original set V. All the arcs in A^1 that start at v_0 they ends at a vertex v_k, $k \in D$. Thus when an (v_0, v_k) is traversed by a path then it indicates that the route starts at depot k. Similarly, all the arcs in A^1 that end at v_0 they start at a vertex v_i, $i \in V^+$. Those arcs indicate the last customer visited by the routes. Note that the resource r ensures that any resource constrained path is associated with a route that satisfies the capacity constraint.

3.2 MILP formulation for the proposed VRPSolver model

The MILP formulation corresponding to the Formulation (1) uses two types of non negative integer variables: x and λ. For each $a \in A$, the variable x_a indicates the number of times that the arc a is traversed. Each variable λ_p indicates the number of times that a given resource constrained path p on G^1 is used at optimal solution. Besides, the formulation uses the following notation: $\delta^-(i)$ indicates the set of arcs in A that are incident to i; h^p_a indicates the number of times that the arc a appears in the path p; P indicates the set of resource constrained paths on G^1. Finally, for each $a = (i,j) \in A$ such $i,j \neq 0$, the mapping $M(x_a)$ is defined as $M(x_a) = \{(v_i, v_j)\}$, for each arc $a \in A$. Here we emphasize that there are no arcs in A that arrive to the depots. Thus, the routes in the solution are all open as the MDOVRP definition. The formulation follows.

$$\text{Min} \sum_{a \in A} c_a x_a \tag{4a}$$

$$\text{S.a.} \sum_{a \in \delta^-(i)} x_a = 1, \qquad \forall i \in V^+. \tag{4b}$$

$$x_a = \sum_{p \in P} \left(\sum_{a' \in M(x_a)} h^p_{a'} \right) \lambda_p, \quad \forall a \in A, \tag{4c}$$

$$0 \leq \sum_{p \in P} \lambda_p \leq |V^+|, \tag{4d}$$

$$\lambda_p \in \mathbb{Z}_+, \quad \forall p \in P, \tag{4e}$$

$$x_a \in \mathbb{Z}, \quad \forall a \in A. \tag{4f}$$

The objective function (4a) aims to minimize the total travel cost. The constraints (4b) ensure that each customer is visited exactly once. The constraints (4c) show the relation between variables x and λ. Basically, each variable x_a is given by the number of times that the arcs in $M(x_a)$ are traversed considering all paths at the optimal solution. The mapping sets M are defined in a way that each feasible route for the problem is represented by a resource constrained path over G^1. The constraints (4d) ensure that at most $|V^+|$ paths are used at optimal solution.

In addiction to the graphs described in the Sect. 3.1 and the MILP Formulation (4), we also provide the packing sets $\mathcal{P} = \cup_{i \in V_+} \{v_i\}$. in order to activate state-of-art BCP algorithm elements. In other words, there is a different packing set for each customer, which is very common in VRPSolver models where each customer is visited exactly once. Moreover, we define a RCC separator such that the capacity $\mathcal{Q} = Q$ and the demand function d returns $d(S = \{v_i\}) = w_i$ from a given packing set $S = \{v_i\} \in \mathcal{P}$. This kind of RCC separator is frequently used in VRPSolver models for VRP variants that consider classical capacity constraint [11].

3.3 Toy example for the proposed VRPSolver model

In order to illustrate the main proposed VRPSolver model concepts, we present a toy MDOVRP instance. That instance has three customers ($|V^+| = 3$), two depots

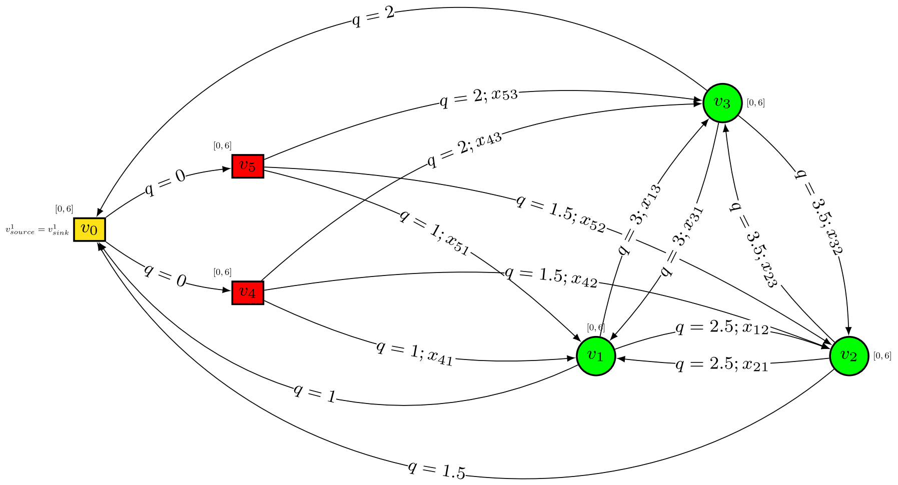

Fig. 1 Path generator graph G^1 for the toy MDOVRP instance

($|D| = 2$), $Q = 6, w_1 = 2, w_2 = 3,\ w_3 = 4,\ w_4 = 0$ e $w_5 = 0$. The travel costs are the following: $c_{(1,2)} = c_{(2,1)} = c_{(2,3)} = c_{(3,2)} = 2$, $c_{(1,3)} = c_{(3,1)} = c_{(5,2)} = 1$, $c_{(4,1)} = 4$, $c_{(4,2)} = 6$, $c_{(4,3)} = 7$, $c_{(5,1)} = 3$ and $c_{(5,3)} = 5$.

The Fig. 1 illustrates the path generator graph G^1. The vertex marked in yellow represents the vertex where the paths should start and end, the vertices marked in red represent the depots and the vertices marked in green represent the customers. We put on each arc its resource consumption denoted by q. We put an interval [0, 6] close each vertex indicating that this vertex can be visited by a path only if the accumulated resource consumption does not exceed $Q = 6$ before the visit. Finally, we put a variable x on a given arc a if $a \in M(x)$.

The set P of all feasible resource constrained paths is composed of the following paths: $p_1 = (v_0 \to v_4 \to v_1 \to v_0)$, $p_2 = (v_0 \to v_4 \to v_2 \to v_0)$, $p_3 = (v_0 \to v_4 \to v_3 \to v_0)$, $p_4 = (v_0 \to v_4 \to v_1 \to v_2 \to v_0)$, $p_5 = (v_0 \to v_4 \to v_2 \to v_1 \to v_0)$, $p_6 = (v_0 \to v_4 \to v_1 \to v_3 \to v_0)$, $p_7 = (v_0 \to v_4 \to v_3 \to v_1 \to v_4)$, $p_8 = (v_0 \to v_5 \to v_1 \to v_0)$, $p_9 = (v_0 \to v_5 \to v_2 \to v_0)$, $p_{10} = (v_0 \to v_5 \to v_3 \to v_0)$, $p_{11} = (v_0 \to v_5 \to v_1 \to v_2 \to v_0)$, $p_{12} = (v_0 \to v_5 \to v_2 \to v_1 \to v_0)$, $p_{13} = (v_0 \to v_5 \to v_1 \to v_3 \to v_0)$ and $p_{14} = (v_0 \to v_5 \to v_3 \to v_1 \to v_0)$. The complete formulation, corresponding to (4), is the following

$$\begin{aligned}\text{Min } & 4x_{41} + 6x_{42} + 7x_{43} + 3x_{51} + x_{52} + 5x_{53} + 2x_{12} + 2x_{21} \\ & + x_{13} + x_{31} + 2x_{23} + 2x_{32}\end{aligned} \tag{5a}$$

$$\text{S.a. } x_{21} + x_{31} + x_{41} + x_{51} = 1 \tag{5b}$$

$$x_{12} + x_{32} + x_{42} + x_{52} = 1 \tag{5c}$$

$$x_{13} + x_{23} + x_{43} + x_{53} = 1 \tag{5d}$$

$$x_{41} = \lambda_1 + \lambda_4 + \lambda_6 \tag{5e}$$

$$x_{42} = \lambda_2 + \lambda_5 \tag{5f}$$

$$x_{43} = \lambda_3 + \lambda_7 \tag{5g}$$

$$x_{51} = \lambda_8 + \lambda_{11} + \lambda_{13} \tag{5h}$$

$$x_{52} = \lambda_9 + \lambda_{12} \tag{5i}$$

$$x_{53} = \lambda_{10} + \lambda_{14} \tag{5j}$$

$$x_{12} = \lambda_4 + \lambda_{11} \tag{5k}$$

$$x_{21} = \lambda_5 + \lambda_{12} \tag{5l}$$

$$x_{13} = \lambda_6 + \lambda_{13} \tag{5m}$$

$$x_{31} = \lambda_7 + \lambda_{14} \tag{5n}$$

$$x_{23} = x_{32} = 0 \tag{5o}$$

$$0 \le \sum_{i=1}^{14} \lambda_i \le 3 \tag{5p}$$

$$\lambda_p \in \mathbb{Z}_+, p \in P \tag{5q}$$

$$x_a \in \mathbb{Z}_+, a \in A \tag{5r}$$

The optimal routes for the proposed instance are: $r1 : 5 \rightarrow 1 \rightarrow 3$ and $r2 : 5 \rightarrow 2$. The cost for that solution is equal to 5.0.

3.4 Considering time windows constraints

We also apply the proposed VRPSolver model to the MDOVRPTW [1]. In MDOVRPTW, For each customer $i \in V^+$, a service time st_i and a time window $[a_i, b_i]$ are given. For each arc $a \in A$ is given its travel time t_a. For each depot d, the service time is equal to 0 and and a time window $[0, b_0]$ is also given. The previous time windows is equal for all depots, where b_0 represents the maximum time spent by a vehicle during a route. In addition to the other constraints of the MDOVRP, we have to ensure in MDOVRPTW that each node is visited within its time windows (including depot nodes).

The proposed model also can handle the MDOVRPTW if we add a new resource r_2 to the path generator graph with the following characteristics. For each $a = (i,j) \in A^1$, we define $q_a^{r_2} = t_a + st_i$ if $i \neq v_0$ (we consider $t_a = 0$ if $j = v_0$). Otherwise, we define $q_a^{r_2} = 0$. Besides, $[l_i^{r_2}, u_i^{r_2}] = [a_i, b_i]$ for each $i \in V^1$ if $i \neq v_0$. Otherwise, $[l_i^{r_2}, u_i^{r_2}] = [0, b_0]$.

3.5 Hierarchical objetives

The OVRPTW instances tested in this paper consider hierarchical objectives. The first objective is to minimizing the number of routes while the second one is to minimize the total distance traveled. To deal with that characteristic, we first set $R = \sum_{i \in V^+} w_i / Q$. We round R up if it is not an integer number. Next, we execute our approach considering that the number of routes is equal to R. To ensure this characteristic, we replace the constraints (4d) with $R \le \sum_{p \in P^k} \lambda_p \le R$. If the problem is infeasible, we set $R = R + 1$ and we execute the algorithm again. Otherwise, R is the minimum number of routes and we return the minimum distance total distance traveled associated with R.

4 Computational experiments

In this section, we present several computational experiments over MDOVRP, MDOVRPTW, OVRP and OVRPTW instances. All experiments were performed on a computer with an Intel Core i7-3770 processor with 3.40 GHz and 16 GB of RAM on Ubuntu 18.04.2 LTS operating system. We used the VRPSolver v0.4.1 that can be downloaded at https://vrpsolver.math.u-bordeaux.fr/. Each instance is solved using a single thread. CPLEX v12.10 is used as linear programming and MILP solvers. All source and configuration as well all instances tested in this paper are available at https://github.com/mcroboredo/MDOVRP.

In this section, we present only average statistics but we provide a supplementary online material with detailed statistics for each instance. The tables presented in this section have the following common columns. Column *# Inst.* indicates the number of instances of a given group. Column *# opt.* indicates the number of optimal solutions found for a given group within the time limit. Column *Avg Time(s)* indicates the average CPU time in seconds considering all instances of the group. When the instance is not solved to the optimality, we compute the time limit for the average time.

The remainder of this section is organized as follows. In Sect. 4.1 we present a comparison between our method and the best exact MILP formulation for the MDOVRP proposed in [8]. In Sect. 4.2, we present results for 440 MDOVRP instances that are being proposed in this paper. In Sects. 4.3, 4.4 and 4.5, we present the results of our method over respectively MDOVRPTW, OVRP and OVRPTW instances.

4.1 Comparison with the literature for MDOVRP instances

The 24 MDOVRP literature instances tested in this paper were initially proposed in [3, 4] for another Multi-Depot Vehicle Routing Problem that considers closed routes and a maximum time limit for the routes. To adapt those instances for the MDOVRP, we ignore the maximum time limit of a route and we do not consider

Table 1 Average results of the proposed algorithm over the 24 MDOVRP literature instances

Description	# Inst.			Proposed method			
		Literature		With ub_0		Without ub_0	
		#opt	Avg. time (s)	#opt	Avg. time (s)	#opt	Avg. time (s)
Literature Inst	24	16	2518.7	24	138.3	24	239.8

the arcs incidents to depots. That adaptation for the same instances was also used by other papers that deal with MDOVRP such as [2, 6–8, 10, 13, 15].

Table 1 presents a comparison between the proposed BCP algorithm and the MILP formulation proposed in [8]. For a fair comparison, we implement the literature formulation. Therefore, the proposed and the literature algorithms are run using the same machine specifications. Our algorithm is tested in two ways for each instance: in the first one, the algorithm receives an initial valid upper bound given by the best solution value among the heuristics proposed in [2, 6, 10,13, 15]; in the second way, our algorithm does not receive any limit. We set the time limit to 7200 s.

Observing Table 1, we can note that both versions of our algorithm outperform the literature one. Besides, our method solved to the optimality all 24 lature instances even when that method did not receive any initial upper limit.

4.2 Tests on proposed MDOVRP instances

We also propose new 440 MDOVRP instances. Those instances are based on the 22 instances with up to 200 nodes proposed in [18] for the classical Capacitated Vehicle Routing Problem (CVRP). The name of each original CVRP instance has 3 terms separated by "-". The first term is always "X" that is the class of the instances. The second term indicates the total number of nodes. The third term indicates the number of vehicles. For example, the instance X-n106-k14 has 106 nodes and 14 vehicles. For each one of the 22 instances, we generate 20 different MDOVRP instances. The number of depots $|D|$ varies from 2 to 5. One of the depots is located in the same location as the instance's original depot. We randomly select the remaining depots among the other nodes of the instance. For each value of $|D| \in \{2, 3, 4, 5\}$, we generate five different instances.

Table 2 presents results of our algorithm over the proposed 440 instances. Each instance was executed initially with a time limit of 18000 s without any valid upper bound. Next, we execute the instance again with the best upper bound found.

The columns of Table 2 are described in the following way. The Column *Group* indicates the name of the CVRP instance that was used to generate 20 different MDOVRP instances. The Column *#opt* indicates the number of instances that were solved to the optimality within 18000 s of execution. The Column *Avg. Time(s)* indicates the average time considering the 20 instances of the group. When the instance is not solved to the optimality, the calculus of the average time considers 18000 s.

Observing Table 2, we note that we do not solved to the optimality 22 and 12 instances when our method is executed respectively without and with an initial upper limit.

4.3 Tests on MDOVRPTW instances

For the MDOVRPTW, we considered the 20 instances proposed in [4] for a version of the problem with close routes (the last node of the route is the depot where the

Table 2 Average results of the proposed algorithm over the proposed 440 MDOVRP instances

Group	#Inst	Proposed method			
		Without ub_0		With ub_0	
		#opt	Avg. time (s)	#opt	Avg. time (s)
X-n101-k25	20	20	253.12	20	22.16
X-n106-k14	20	20	611.15	20	74.10
X-n110-k13	20	20	146.62	20	15.60
X-n115-k10	20	20	173.77	20	21.68
X-n120-k6	20	20	134.41	20	34.72
X-n125-k30	20	20	793.94	20	91.05
X-n129-k18	20	20	479.11	20	39.95
X-n134-k13	20	20	2124.79	20	366.15
X-n139-k10	20	20	596.96	20	59.24
X-n143-k7	20	20	1725.27	20	779.44
X-n148-k46	20	20	209.90	20	10.74
X-n153-k22	20	20	1166.88	20	64.08
X-n157-k13	20	20	180.55	20	39.96
X-n162-k11	20	20	2853.38	20	238.63
X-n167-k10	20	19	2039.27	20	918.21
X-n172-k51	20	16	5572.06	18	2677.76
X-n176-k26	20	18	6085.03	20	1,152.35
X-n181-k23	20	20	558.72	20	156.44
X-n186-k15	20	19	6205.71	20	422.50
X-n190-k8	20	19	6730.07	20	1114.80
X-n195-k51	20	12	11,938.67	14	6286.89
X-n200-k36	20	15	6782.60	16	3941.77
Total	440	418		428	

Table 3 Average results of the proposed algorithm over 20 MDOVRPTW instances

Description	# Inst.	With ub_0		Without ub_0	
		# opt	Avg. time (s)	# opt	Avg. time (s)
MDOVRTW Inst	20	20	31.3	20	315.1

route started) and a maximum time limit for the routes. To adapt those instances for the MDOVRPTW, we ignore the maximum time limit of a route and we do not consider the arcs incidents to depots.

Table 3 presents our results for the MDOVRPTW instances. Each instance was executed initially without any valid upper bound. Next, we execute the instance again with the best upper bound found.

Table 4 Comparison between our method and the branch-and-cut algorithm proposed in [9]

Class	#Inst.	#opt	
		Literature	This paper
A	27	27	27
B	23	19	23
E, F, M	17	11	15
P	24	19	24
Total	91	76	89

Table 5 Results of our approach for 9 OVRP benchmark instances

Description	# Inst.	Not fixed nr		Fixed nr	
		# opt	Avg. time (s)	# opt	Avg. time (s)
OVRP Inst	9	9	294.75	7	1621.68

Observing Table 3, we note that our method solved to the optimality all 20 instances with a reasonable average time even when that method did not receive any valid upper limit.

4.4 Tests on OVRP instances

We also test our method on benchmark instances of the OVRP, which is particular case of the MDOVRP arising when $|D| = 1$. All the instances were originally proposed for the classical Capacitated Vehicle Routing Problem (CVRP), but they can be easily adapted to the OVRP by not considering the arcs incidents to depot. In total we use 100 OVRP instances, where 91 of them were tested by the exact method proposed in [9] while 9 of them were tested by several OVRP heuristic algorithms such as those proposed by [12, 16, 21].

To the best of our knowledge, the only exact approach for the OVRP is the branch-and-cut algorithm proposed in [9]. That algorithm was tested on classical 91 CVRP instances from the classes A,B,E,F,M and P. These classes are widely used for testing CVRP approaches and for more details about them, we refer to [18]. Those instances consider a fixed number of vehicles R. To ensure this characteristic, we replace the constraints constraints (4d) by $R \leq \sum_{p \in P^k} \lambda_p \leq R, k \in D$, where R represents the fixed number of routes.

Table 4 presents a comparison between the number of optimal solutions found by our method and by the branch-and-cut algorithm proposed in [9]. The literature algorithm considered a time limit of 3600 s for each experiment. For a fair comparison, we compared the processors of the computers used by us and by the literature through the site https://www.cpubenchmark.net/. We considered the time limit 540 s for each experiment carried out by our algorithm base on that comparison. Each column in Table 4 is labeled as follows. The Column *Class* indicates the name of the corresponding class of instances. The Column *#Inst* indicates the number of instances in the corresponding class. The Columns *# opt Literature* and *# opt This*

Table 6 Average results of the proposed algorithm over OVRPTW instances

Group	#Inst	#opt	Avg. time (s)
R101-R112	12	10	832.6
C101-C109	9	9	5.7
RC101-RC108	8	8	1064.7
R201-R211	11	2	10351.7
C201-C208	8	8	860.0
RC201-RC208	8	1	5825.5

paper indicate the number of instances solved to the optimality by respectively the literature and the proposed algorithms within the time limit.

Observing Table 4, we can note that our method outperforms the literature one since our method did not solved to the optimality within the time limit just 2 instances while the literature algorithm did not prove the optimal solution for 15 instances.

We also test the proposed algorithm on 9 OVRP benchmark instances C1, C2, C3, C4, C5, C11, C12, F11 and F12. For more details about them we refer to [21]. Table 5 shows our results for those instances where each one is tested in two ways: without and with a fixed number of routes (nr). We provide our algorithm an initial valid upper bound for each instance. For the case without a fixed number of vehicles, that bound was taken from [21]. For the other case, that bound was taken from [16].

Analysing Table 5, we conclude that our method just does not solve to the optimality just two instances within 7200 s of execution.

4.5 Tests on OVRPTW instances

For the OVRPTW, we follow [20] and generated instances based on classical Solomon instances [14] which is divided in six groups R101-R112, C101-C109, RC101-RC108, R201-R211, C201-C208, RC201-RC208. As in [20], we consider that the instances have the hierarchical objectives described in Sect. 3.5. Thus, we execute the procedure described in that section. We set a time limit of 18000 s for each experiment. Table 6 presents the results of our approach over each group of OVRPTW instances.

Analysing Table 6, we conclude that our method solves to the optimality within the time limit all instances in the groups C101-C109, RC101-RC108 and C201-C208. However our approach solved to the optimality only 2 instances in the group R201-R208 and 1 instance in the group RC201-RC208. It happened because the vehicle capacity is large for the instances in those groups. Thus, the routes in the solution tend to be long and it decreases the performance of our approach.

5 Conclusions

This paper presented a BCP algorithm for the MDOVRP that was coded within the VRPSolver framework. We tested the performance of the proposed algorithm on 24 instances frequently used by the literature about the problem. The results showed that our method outperformed the best exact MILP formulation proposed in [8]. Besides, our method solved to the optimality all instances including eight instances that were open so far.

We also applied the proposed approach to instances of the MDOVRPTW, OVRP, OVRPTW. For all the variants, our method solved to the optimality almost all instances in reasonable computational times.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11590-023-02072-y.

References

1. Bezerra, S.N., de Souza, S.R., Souza, M.J.F.: A general vns for the multi-depot open vehicle routing problem with time windows. Optim. Lett. (2023). https://doi.org/10.1007/s11590-023-01990-1
2. Brandão, J.: A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem. Eur. J. Oper. Res. **284**(2), 559–571 (2020)
3. Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Netw. Int. J. **30**(2), 105–119 (1997)
4. Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. J. Op. Res soc. **52**(8), 928–936 (2001)
5. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Manage. Sci. **6**(1), 80–91 (1959)
6. Lahyani, R., Gouguenheim, A.L., Coelho, L.C.: A hybrid adaptive large neighbourhood search for multi-depot open vehicle routing problems. Int. J. Prod. Res. **57**(22), 6963–6976 (2019)
7. Lalla-Ruiz, E., Expósito-Izquierdo, C., Taheripour, S., Voß, S.: An improved formulation for the multi-depot open vehicle routing problem. OR Spectr. **38**(1), 175–187 (2016)
8. Lalla-Ruiz, E., Mes, M.: Mathematical formulations and improvements for the multi-depot open vehicle routing problem. Optim. Lett. **15**(1), 271–286 (2021)
9. Letchford, A.N., Lysgaard, J., Eglese, R.W.: A branch-and-cut algorithm for the capacitated open vehicle routing problem. J. Op. Res. Soc. **58**(12), 1642–1651 (2007)
10. Liu, R., Jiang, Z., Geng, N.: A hybrid genetic algorithm for the multi-depot open vehicle routing problem. OR Spectr. **36**(2), 401–421 (2014)
11. Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F.: A generic exact solver for vehicle routing and related problems. Math. Progr. **183**(1), 483–523 (2020)
12. Ruiz, E., Soto-Mendoza, V., Barbosa, A.E.R., Reyes, R.: Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. Comput. Ind. Eng. **133**, 207–219 (2019)
13. Sadati, M.E.H., Çatay, B., Aksen, D.: An efficient variable neighborhood search with tabu shaking for a class of multi-depot vehicle routing problems. Comput. Op. Res. **133**, 105269 (2021)
14. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. **35**(2), 254–265 (1987)
15. Soto, M., Sevaux, M., Rossi, A., Reinholz, A.: Multiple neighborhood search, tabu search and ejection chains for the multi-depot open vehicle routing problem. Comput. Ind. Eng. **107**, 211–222 (2017)
16. Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Comput. Op. Res. **40**(10), 2519–2531 (2013)
17. Tarantilis, C., Kiranoudis, C.: Distribution of fresh meat. J. Food Eng. **51**(1), 85–91 (2002)
18. Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A.: New benchmark instances for the capacitated vehicle routing problem. Eur. J. Oper. Res. **257**(3), 845–858 (2017)

19. Yao, B., Hu, P., Zhang, M., Tian, X.: Improved ant colony optimization for seafood product delivery routing problem. Promet-Traffic Transp. **26**(1), 1–10 (2014)
20. Yu, N., Qian, B., Hu, R., Chen, Y., Wang, L.: Solving open vehicle problem with time window by hybrid column generation algorithm. J. Syst. Eng. Electron. **33**(4), 997–1009 (2022)
21. Zachariadis, E.E., Kiranoudis, C.T.: An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. Comput. Op. Res. **37**(4), 712–723 (2010)