Solving Vehicle Routing Problems With Intermediate Stops Using VRPSolver Models

Marcos Roboredo, Ruslan Sadykov, Eduardo Uchoa

Solving Vehicle Routing Problems With Intermediate Stops Using VRPSolver Models

Marcos Roboredo[1], Ruslan Sadykov[2], and Eduardo Uchoa[1]

[1]Departamento de Engenharia de Produção, Universidade Federal Fluminense - Rua Passo da Pátria, 156, São Domingos, Niterói, RJ, 24210-240, Brazil
[2]Inria centre at the University of Bordeaux, 200 Avenue de la Vieille Tour, Talence 33405, France

November 25, 2022

Abstract

In this paper, we propose graph-based models for several vehicle routing problems with intermediate stops: the capacitated multi-trip vehicle routing problem with time windows, the multi-depot vehicle routing problem with inter-depot routes, the arc routing problem with intermediate facilities under capacity and length restrictions and the green vehicle routing problem. In these models, the set of feasible routes is represented by a set of resource constrained paths in one or several graphs. Intermediate stops are supported by the possibility to define negative resource consumption for some arcs. The models that we propose are then solved by VRPSolver, which implements a generic branch-cut-and-price exact algorithm. Thus, a simple parameterization enables us to use several state-of-the-art algorithmic components: automatic stabilization by dual price smoothing, limited-memory rank-1 cuts, reduced cost-based arc elimination, enumeration of elementary routes, and hierarchical strong branching. For each problem, we numerically compare the proposed methodology with the best exact approach found in the literature. State-of-the-art computational results were obtained for all problems except one.

1 Introduction

Vehicle Routing Problems (VRPs) form a highly studied class of combinatorial optimization problems with applications in a large number of fields, most often related to freight transportation and logistics. [13] introduced the basic Capacitated VRP, in which the set of customers should be served by a set of identical vehicles from a single depot under customer demand and vehicle capacity constraints to minimize the total transportation cost. Since then, a large number of VRP variants has been proposed in the literature involving different product types, vehicle types, multiple locations, and additional operational constraints.

Some VRP variants involve a possibility for a vehicle to make an intermediate stop inside a route in order to improve its capability to fulfil service tasks. These variants are combined under the class of Vehicle Routing Problems with Intermediate Stops (VRPIS). In most VRPIS, the vehicles stop at so called Intermediate Facilities (IFs) in order to replenish a certain resource. The stops at IFs can model several practical situations such as replenishment or unloading of a cargo, refuelling, or ensuring breaks and rests periods for the driver [36].

Nowadays, branch-cut-and-price (BCP) approach is the leading methodology for exactly solving VRPs, as surveyed in [11]. BCP algorithms solve VRPs by formulating them as integer programs with an exponential number of variables, each one corresponding to a feasible vehicle route. The variables are then generated dynamically by solving pricing subproblems modelled

as resource constrained shortest path problems (RCSPP). Families of valid cutting planes are also generated to improve the linear relaxation of the problem. Finally, branching is used to close the primal-dual gap. Numerous algorithmic components have been proposed in literature to improve the efficiency of BCP algorithms for VRPs.

Unfortunately, the implementation of a BCP algorithm with state-of-the-art components for a particular VRP requires a lot of time even for a skilled team. In order to mitigate this problem, [33] proposed a generic model and an associated BCP algorithm for it for solving a wide range of VRPs. The modelling approach relies on 1) representing the set of feasible routes as a set of resource constrained paths in one or several graphs, 2) defining mapping between arcs in graphs and variables in the integer programming part of the model, and 3) specifying so-called packing sets of vertices or arcs in the graphs. An implementation of this modelling interface and the associated generic BCP algorithm as the VRPSolver package has shown excellent results for many classic VRP variants, being often better than the best specific solvers proposed for those variants.

The efficiency of the VRPSolver models can be explained by the fact that its associated BCP algorithm already includes most techniques/features known in the literature to be necessary for achieving a state-of-the-art performance. The main techniques are automatic stabilization by dual price smoothing, limited-memory rank-1 cuts, rounded capacity cuts, *ng*-route relaxation, reduced cost-based arc elimination, enumeration of elementary routes, bucket-graph based labeling algorithm for the pricing problem, and hierarchical strong branching. Many of these techniques are often missing in algorithms for specific problems, as they are very complex and time consuming to implement. However, their implementation pays off in a generic code, as it can be reused for many problems. In spite of all those advantages, there are also disadvantages of using VRPSolver: (i) it is always possible to make a faster code for a specific problem (however, at the expense of a large implementation effort); ii) there may exist problem-specific techniques which are not available in the generic solver; iii) there is a learning curve to be able to understand and efficiently use VRPSolver.

This paper aims to show that several VRPIS can be successfully tackled by the modelling approach just mentioned. Describing intermediate stops inside the proposed paradigm is not obvious. We show that this can be done by exploiting the possibility to define a negative resource consumption for arcs in graphs. Modeling and solving the pricing subproblem for VRPIS as a RCSPP is not new. [26] pointed out that the labelling algorithm to solve the standard RCSPP can be adapted to the case with intermediate stops by resetting a certain resource when visiting some special nodes. However, this modification requires a possibility to access and modify an implementation of a sophisticated BCP algorithm and the technical competence to do so. Our contribution is to show that instead one can obtain an easily reproducible state-of-the-art approach for several VRPIS just by modelling.

The current best approaches for VRPIS often work with so-called trips, i.e. parts of routes between two consecutive intermediate stops. Thus, trip variables are used instead of route variables. An advantage of trip variables is that they can be generated more easily and sometimes even completely enumerated in the preprocessing step. Route variables are often considered to be difficult to dynamically generate due to the fact that routes in VRPIS are usually long. The second contribution of this work is to show that fast dynamic generation of route variables for VRPIS is possible if state-of-the-art algorithmic components are used. Moreover, our route-based approach outperforms trip-based approaches from the literature for some problems.

The following VRPIS are considered in this work:

- The Capacitated Multitrip Vehicle-Routing Problem With Time Windows (CMTVRPTW) [31].
- The Multi-depot vehicle routing problem with inter-depot routes (MDVRPI) [12].
- The Arc Routing Problem with Intermediate Facilities under Capacity and Length Restrictions (CLARPIF)[18].

- The Green Vehicle Routing Problem (G-VRP) [16]

The CMTVRPTW consists of assigning a sequence of trips (paths starting and ending at a given depot) to each available vehicle. The objective is to minimize the total distance travelled by the vehicles. Each customer demand is served by a single vehicle and the total demand served in a trip cannot exceed the vehicle capacity. Besides, the service of every customer should start within a pre-specified time window. The CMTVRPTW can be viewed as a VRPIS because the capacity of a vehicle is replenished every time this vehicle passes through the depot between two trips.

The MDVRPI consists of assigning routes to each available vehicle in order to minimize the total travelled distance. The demand of each customer is served by a single vehicle. There are several depots and each route has to start and end at the same depot. While following a route, a vehicle can stop at the depots. The total demand served between two consecutive depots in a route can not exceed the vehicle capacity. In other words, the capacity of a vehicle is replenished every time it stops at a depot. Thus the depots can be view as IFs.

The CLARPIF consists of assigning routes that start and end at a given depot to each available vehicle in order to minimize the total travelled distance. There is a set of required edges, each one with a different demand. Intermediate facilities are available, and any vehicle can stop at these facilities during while following a route in order to renew its capacity. Again, the total demand served between two IFs can not exceed the vehicle capacity.

The G-VRP consists of assigning routes to Alternative Fuel Vehicles (AFVs) that start and end at a given depot in order to minimize the total travelled distance travelled. Each customer is visited exactly once by a single AFV. The problem involves a set of Alternative Fuel Stations (AFS). Each one of them can be visited by an AFV to refuel as many times as necessary while following a route. There is a limit on the total travel time for each route. Moreover, there is another limit on the total distance travelled by an AFV between two consecutive refuels. The G-VRP can be considered as a VRPIS because the fuel of a vehicle is replenished every time this vehicle passes through an AFS.

We highlight that, although VRPSolver is the most generic existing solver for VRPs, there are still VRP variants for which no efficient VRPSolver model exists. In particular, several extensions of the considered problems known in the literature, cannot be modelled and solved using VRPSolver in an efficient way.

This paper has the following structure. Section 2 presents a literature review on vehicle routing problems with intermediate stops. Section 3 presents an overview of the VRPSolver generic model which serves as a foundation of our models, as well as the associated solution approach . In Section 4, we present specific models for four problems we consider. Section 5 presents numerical results and comparison with the best exact approaches found in the literature. Finally, Section 6 draws conclusions and presents some future research directions.

2 Literature Review

In VRPIS, the vehicles stop at IFs in order to replenish a certain resource. [36] classified the stops into three following types.

(i) Stops for replenishment and unloading of goods or waste. The most common situation occurs when the capacity of a vehicle is fully replenished whenever this vehicle stops at an IF. In the literature, several works deal with VRPIS with this characteristic. In some cases, the depots act as IFs. It happens for example in the multi-trip VRP [10] where there is a single depot, or in the MDVRPI [12] where there are several depots. In other cases, the IFs are usually represented by special nodes in the network [2, 17, 18, 38, 19, 41, 42, 43].

(ii) Stops for refuelling. In recent years, researchers have started to pay attention to VRPs with a fleet of alternative fuel vehicles (AFVs). These problems involve refueling stations where

the vehicles can stop en route to increase its remaining range and keep itself operational [16, 14, 25].

(iii) Stops for rests and breaks. Some VRPs incorporate break or rest periods for the drivers that arise from hour-of-service regulations or multiday planning [20, 39, 15, 21, 35]. In these problems, the places where the breaks or rests occur can be viewed as IFs.

Currently in the literature, best exact approaches for VRPIS are based on mixed integer linear programming (MILP) formulations with an exponential number of variables. Branch-and-price and branch-price-and-cut algorithms are used to solve such formulations. In some approaches, one variable per complete feasible route is defined. These route variables are generated dynamically by solving the pricing subproblems, usually modeled as RCSPP. Route-based formulations are used for example in [14, 25, 21, 29, 40].

A disadvantage of route-based formulations is that the pricing problem can be time consuming to solve as routes are usually long in VRPIS. Thus, alternative formulations are also considered in the literature. They are based on variables representing parts of routes. Trip-based formulations are employed in [28, 24, 31, 44] for solving multi-trip VRPs. [6] tackle the Traveling Salesperson Problem With Hotel Selection using the formulation in which every variable represents a single-day part of the multi-day TSP tour. [8] used a MILP formulation based on pre-enumerated trips (which they call paths) to solve the G-VRP.

We now review the existing exact approaches for solving the problems addressed in this paper: CMTVRPTW, MDVRPI, CLARPIF and G-VRP.

The CMTVRPTW is a capacitated multi-trip VRP with time windows. In addition to the time windows, some variants consider service-dependent loading times (CMTVRPTW-LT), limited trip duration (CMTVRPTW-LD) and release dates (CMTVRPTW-R). In CMTVRPTW-LT, each customer i is associated with a loading time l_i. For each trip, the vehicle spends $\beta \times \sum_{i \text{ is visited}} l_i$ before departing, where β is a given constant such that $0 < \beta < 1$. In CMTVRPTW-LD, there is a maximum time limit for each trip. To the best of our knowledge, all papers about CMTVRPTW-LD also consider service-dependent loading times. In CMTVRPTW-R, each customer i is associated with a release date r_i. The trip where i is served can not start before r_i. For more details about the multi-trip VRPs classification, we refer to [10]. To solve the CMTVRPTW-LD, we highlight three representative papers: [4] proposed a route-based formulation and the corresponding BCP algorithm, [28] proposed a pseudo-polynomial network flow model and [23] proposed an exact two-phase algorithm. The first phase enumerates possible ordered lists of clients which match the maximum trip duration criterion. The second phase uses a BCP scheme based on a trip-based formulation. For multi-trip CMTVRPTW-LT, Both route-based and trip based formulations and corresponding BCP approaches are considered by [24]. For the CMTVRPTW-R, [9] proposed a hybrid genetic algorithm. The consideration of service-dependent loading times, limited trip duration or release dates is very common in multi-trip problem variants not only because of theoretical aspect of these characteristics but also because they can model several real-life situations. As an example, we highlight the researches developed by [45], [27], [46] and [30].

Recently, [31] and [44] proposed exact algorithms for the four variants: CMTVRPTW, CMTVRPTW-LT, CMTVRPTW-LD and CMTVRPTW-R. [31] introduced a novel structure-based formulation that involves fewer variables than the trip-based and route-based formulations. Based on this novel formulation, the authors propose an exact solution framework that relies on column generation, column enumeration, and generation of cutting planes. [44] improve the methodology proposed by [31]. In specific for CMTVRPTW, focus of this paper, the methodology proposed by [31] optimally solve instances with up to 50 customers while the algorithm proposed by [44] optimally solved all instances with up to 70 customers and obtained near-optimal solutions with the average optimality gap less than 0.3% for instances with 80 and 100 customers.

The MDVRPI is introduced by [12], who suggest a heuristic for this problem. [29] propose two branch-and-price algorithms for the problem based on the standard route-based formulation. The first algorithm is a standard one in which the pricing problem is solved as a RCSPP. In the

second algorithm, the pricing problem is solved by a two-phase approach which first enumerates trips, and then applies a labeling algorithm over the set of enumerated trips. [40] propose a BCP algorithm based on the route-based formulation. The algorithm incorporates several state-of-the-art algorithmic components, including *ng*-routes, variable fixing, elementary route enumeration, generation of rounded capacity inequalities and limited-memory subset-row cuts. [40] obtain optimal solutions for instances with up to 70 customers and 7 depots.

The first version of the CLARPIF is introduced by [17]. A generalised version which involves the constraint on the maximum total route distance is considered in several papers [18, 34, 19, 42, 43], which propose heuristic approaches. To the best of our knowledge, no exact algorithm exist for the CLARPIF in the literature.

VRPs that consider environmental impact are widely studied in the literature in recent years. We refer to [3] for a review of these problems. To our knowledge, there are two exact approaches for the G-VRP in the literature. [1] use the route-based formulation and apply a BCP algorithm to solve it. They employ the concept of multi-graph to solve the pricing problem. In the multi-graph, there can be several arcs between a pair of nodes, where each arc corresponds to a different set of refuelling stops visited between two customer nodes. The authors also separate robust non-standard k-path cuts which are based on maximum route duration constraint. [8] propose a two-phase solution approach in which a route is seen as the composition of partial paths, each one visiting a set of customers without intermediate stops at AFS. In the first phase, all feasible partial paths are generated. Dominance rules are used to reduce the number of partial paths. In the second phase, an ILP is generated in which every variable corresponds to a partial path. This integer linear program is then solved by a MIP solver. [8] show that small- and medium-sized instances can be solved by their two-phase approach faster than by the BCP algorithm proposed by [1]. For large-size instances, the partial path-based approach cannot be used as an exact algorithm, but can be converted to a heuristic.

There exist several alternative generic solvers for vehicle routing problems such as `Vroom` (`http://vroom-project.org`), `OptaPlanner` (`https://www.optaplanner.org`, `jsprit` (`https://jsprit.github.io`), `Google OR-Tools` (`https://developers.google.com/optimization`), or `LocalSolver` (`https://www.localsolver.com`). However, all these solvers are heuristic. To the best of our knowledge, `VRPSolver` (`https://vrpsolver.math.u-bordeaux.fr`) is the only generic exact solver for VRPs. Thus, direct performance comparison is difficult, as VRPSolver is focused on improving dual bounds, whereas heuristic solvers are designed for improving primal bounds. Modeling capabilities of solvers are different, there are more generic and less generic ones. We should nevertheless mention that modelling with VRPSolver is usually more difficult for a user, as VRPSolver models involve abstract entities, such as variables, graphs, resources, etc. Heuristic VRP solvers usually offer more practice-oriented models which directly involve vehicles, depots, capacities, time windows, etc.

3 Overview of the VRPSolver Generic Model

In this section, in order to make the paper self-contained, we present the generic VRPSolver model. VRPSolver a framework proposed by [33] to facilitate the design of BCP algorithms for VRPs and related problems. The generic model contains a MIP formulation that contains variables representing feasible routes. The set of feasible routes is modeled by a set of resource constrained paths in one or several directed graphs. VRPSolver solves the model using a generic BCP algorithm that generates path variables dynamically by solving the pricing subproblems which are RCSPP. In addition to the MIP formulation and directed graphs, one can also define so-called *packing sets* that allow the user to activate some state-of-art BCP components, such as *ng*-path relaxation, generation of rounded capacity and limited-memory rank-1 cuts, and route enumeration.

3.1 Graphs for RCSP subproblems

Let $\mathcal{K}$ be a finite set of directed graphs $G^k = (V^k, A^k)$, $k \in \mathcal{K}$. Each graph G^k has special vertices v^k_{source} and v^k_{sink} that can be different vertices or can be the same vertex. For each $k \in \mathcal{K}$, there is a set of resources R^k. For each resource $r \in R^k$ and each arc $a \in A^k$, value $q_{ar} \in \mathbb{R}$ indicates the consumption of resource r on arc a. This value can be both positive or negative. However, we highlight that for every G^k with cycles, it is mandatory to have at least one resource in R^k with non-negative consumption on all arcs. For each $k \in \mathcal{K}$, for each resource $r \in R^k$ and for each vertex $v \in V^k$, there is an interval $[l_{vr}, u_{vr}]$ for accumulated consumption of resource r. A path $p = (v^k_{source} = v^p_0, a^p_1, v^p_1, \ldots, a^p_{n-1}, v^p_{n-1}, a^p_n, v^p_n = v^k_{sink})$ in G^k is *feasible* if, for every $r \in R^k$, the accumulated resource consumption $\rho^p_{j,r}$ at visit j traversing arc a^p_j, $1 \leq j \leq n$, where

$$\rho^p_{j,r} = \max\{l_{v^p_j,r}, \rho^p_{v^p_{j-1},r} + q_{a^p_j,r}\} \quad \text{and} \quad \rho^p_{0,r} = l_{v^k_{source},r},$$

does not exceed $u_{v^p_j,r}$. By this definition, resources have *disposability* property: they can be consumed for free in order to satisfy lower bounds on accumulated resource consumption. This property is very useful to model resetting of accumulated resource consumption at intermediate facilities.

3.2 MIP formulation

Let P^k be the set of feasible paths in graph G^k, $k \in \mathcal{K}$, and let $P = \cup_{k\in\mathcal{K}} P^k$ be the set of feasible paths in all graphs. For each $k \in \mathcal{K}$, $a \in A^k$, and $p \in P^k$, the number of times that an arc a appears in path p is denoted by h^p_a. The master formulation uses integer variables x_j, $1 \leq j \leq n_1$. The formulation also uses integer variables λ_p, $p \in P$ which computes how many times path p is used. Finally, the formulation uses some more constants. The constant m indicates the total number of constraints. The constants $c_j \in \mathbb{R}$, $j = 1, \ldots, n_1$ indicate the objective function coefficients of variables x. For each constraint $i = 1, \ldots, m$, the constants d_i and $\alpha_{ij} \in \mathbb{R}$, $j = 1, \ldots, n_1$ indicate respectively the right-side and coefficients of variables x for that constraint.

$$\min \quad \sum_{j=1}^{n_1} c_j x_j \tag{1a}$$

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

$$x_j = \sum_{k=1}^{\mathcal{K}} \sum_{p\in P^k} \left(\sum_{a \in M(x_j)} h^p_a \right) \lambda_p, \qquad j = 1 \ldots, n_1, \tag{1c}$$

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

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

$$x_j \in \mathbb{Z}_+, \qquad j = 1, \ldots, \bar{n}_1. \tag{1f}$$

$$\tag{1g}$$

The Formulation (1) is a very generic one where the Equations (1c) define the relation between the x and λ variables. Each one of these equations uses a set $M(x_j)$, representing the *mapping* of the variable x_j. This mapping is defined by the user in a way that $M(x_j) \subseteq \cup_{k=1,\ldots,\mathcal{K}} A^k$. The mapping sets M are not necessarily disjoint and the union of all mapping sets are not necessarily equal to $\cup_{k=1,\ldots,\mathcal{K}} A^k$. Equations (1d) show that for each graph G^k, there is a lower bound L^k and there is an upper bound U^k for the number of feasible paths from this graph participating in a solution. These bounds are also defined by the user.

The VRPSolver solves formulation (1) using a generic BCP algorithm. It works with the following *master* formulation which is the linear programming (LP) relaxation of (1), in which variables x are eliminated:

$$\min \sum_{k=1}^{\mathcal{K}} \sum_{p \in P^k} \left(\sum_{j=1}^{n_1} c_j \sum_{a \in M(x_j)} h_a^p \right) \lambda_p \tag{2a}$$

$$\text{s.t.} \sum_{k=1}^{\mathcal{K}} \sum_{p \in P^k} \left(\sum_{j=1}^{n_1} \alpha_{ij} \sum_{a \in M(x_j)} h_a^p \right) \lambda_p \geq d_i, \qquad i = 1, \ldots, m, \tag{2b}$$

$$L^k \leq \sum_{p \in P^k} \lambda_p \leq U^k, \qquad k \in \mathcal{K} \tag{2c}$$

$$\lambda_p \geq 0, \qquad p \in P. \tag{2d}$$

The master formulation (2) is solved by the iterative column generation procedure. In each iteration a restricted master formulation (RMF) with a subset of variables λ is solved, and then variables λ absent in the RMF with a negative reduced cost are added to it. To find such variables, the *pricing problem* is solved. If no negative reduced cost variable is found, the current solution of the RMF is optimal for the master.

The reduced cost of an arc $a \in A$ is given by:

$$\bar{c}_a = \sum_{j \in M^{-1}(a)} c_j - \sum_{i=1}^{m} \sum_{j \in M^{-1}(a)} \alpha_{ij} \pi_i,$$

where vector $\pi \in \mathbb{R}_+^m$, indicates the dual variables for constraints (2b) and $M^{-1}(a)$ is defined as $M^{-1}(a) = \{j | a \in M(x_j)\}$. The reduced cost $\bar{c}(p)$ of a path $p = (v_0, a_1, v_1, \ldots, a_{n-1}, v_{n-1}, a_n, v_n) \in P^k$ is given by

$$\bar{c}(p) = \sum_{j=1}^{n} \bar{c}_{a_j} - \mu_+^k - \mu_-^k,$$

where values $\mu_+^k \in \mathbb{R}_+$ and $\mu_-^k \in \mathbb{R}_-$ indicate the dual variables for constraints (2c). The pricing subproblem for graph G^k, $k \in \mathcal{K}$, consists in finding one or several paths in P^k with a negative reduced cost. This problem is the RCSPP, and it is solved by a labeling dynamic programming algorithm. We refer to [33] for details about this algorithm. The generic BCP algorithm solves the master formulation (2) on every node of the branch-and-bound search tree. Branching on variables x is then used to solve MIP formulation (1) to optimality. Advanced algorithmic components such as dual price automatic smoothing, arc elimination by reduced costs, and strong branching are employed to reduce the solution time. It is worth mentioning that the column generation and the pricing algorithms are defined automatically from the model given by the user. Thus, the VRPSolver does not allow one to define user-specific column generation and pricing procedures.

3.3 The concept of packing sets

Let $V' = \cup_{k \in \mathcal{K}} V^k \setminus \{v_{source}^k, v_{sink}^k\}$ be the set of all vertices that are neither source or sink of their graphs. Let $\mathcal{P}^V \subset 2^{V'}$ be a collection of mutually disjoint subsets of V'. We say that the elements of $\mathcal{P}^V$ are packing sets if there is at least one optimal solution of the formulation (1) satisfying the following constraints:

$$\sum_{p \in P} \left(\sum_{v \in S} h_v^p \right) \lambda_p \leq 1, \quad \forall S \in \mathcal{P}^V, \tag{3}$$

where the constants h_v^p computes how many times the vertex v is present in a path p. According to (3), the number of times vertices in a packing set $S \in \mathcal{P}^V$ appear in all paths of some optimal solution is at most one. Packing sets often correspond to customers in VRPs. The definition of a proper collection $\mathcal{P}^V$ is a part of the VRPSolver model. As mentioned above, definition of packing sets allows VRPSolver to activate some algorithmic components, namely ng-path relaxation, generation of rounded capacity and limited-memory rank-1 cutting planes, and elementary route enumeration. Activation of such components is necessary for the BCP algorithm to attain the state-of-the-art performance for VRPs. We refer to [33] for details on these components.

4 Models for the Considered Problems

4.1 Capacitated multi-trip vehicle routing problem with time windows

The CMTVRPTW is formally defined as follows. Given a directed graph $G = (V, A)$, the vertex set $V = \{0\} \cup V^+$ consists of the depot with index 0 and the set of customers $V^+ = \{1, 2, ..., n\}$. For each customer $i \in V^+$, a demand $w_i > 0$, a service time $st_i \geq 0$ and a time window $[a_i, b_i]$ are given. For the depot, The demand and the service time of the depot are equal to 0: $w_0 = st_0 = 0$. Let t_{ij} be the travel time between two nodes i, j in V. The arc set A is defined as $A = \{(i, j) \in V \times V : \ i \neq j,\ a_i + t_{ij} + st_i \leq b_j \text{ and } w_i + w_j \leq Q\}$. For each arc $(i, j) \in A$, the travel cost from i to j is denoted by c_{ij}. A fleet of K homogeneous vehicles of capacity Q is available. A trip is defined as a sequence of visited nodes starting and ending at the depot. A journey is a sequence of trips non-overlapping in time with no common customers. The problem aims to find at most K journeys aiming to minimize the total travel cost such that:

- each customer is visited exactly once by a single trip;
- the sum of demands visited during each trip does not exceed Q;
- each node is visited within its time windows (including the depot node);

4.1.1 VRPSolver model for the CMTVRPTW

We define a single path generator graph $G^1 = (V^1, A^1)$. In the set of vertices $V^1 = \{v_0^1, v_1^1, ..., v_{n+1}^1\}$, vertex v_0^1 corresponds to the depot when it is used as the first or the last element of a journey, vertex v_{n+1}^1 corresponds to depot when it is used as an IF, and other vertices correspond to customers. The set of arcs $A^1 = \{(v_i^1, v_j^1) : \ 0 \leq i, j \leq n,\ i \neq j\} \cup \{(v_i^1, v_{n+1}^1) : \ 1 \leq i \leq n\} \cup \{(v_{n+1}^1, v_i^1) : \ 1 \leq i \leq n\}$ includes all possible arcs except the arcs between two depot vertices v_0^1 and v_{n+1}^1. The complete proposed VRPSolver model for the CMTVRPTW is the following.

VRPSolver Model for CMTVRPTW. A single graph $G^1 = (V^1, A^1)$, where V^1 and A^1 are defined just above. $v^1_{\text{source}} = v^1_{\text{sink}} = v^1_0$. $R^1 = \{r_1, r_2\}$, where resource r_1 ensures the vehicle capacity constraint, and resource r_2 guarantees that time windows are respected. $[l_{v,r_1}, u_{v,r_1}] = [0, Q]$ for $v \in V^1 \setminus \{v^1_{n+1}\}$, and $[l_{v^1_{n+1},r_1}, u_{v^1_{n+1},r_1}] = [0, 0]$. $[l_{v^1_i,r_2}, u_{v^1_i,r_2}] = [a_i, b_i]$, for $i \in V^+$, and $[l_{v^1_{n+1},r_2}, u_{v^1_{n+1},r_2}] = [a_0, b_0]$. For all $a = (v^1_i, v^2_j) \in A^1$ such that $j \neq n+1$, $q_{a,r_1} = w_j$. $q_{(v^1_i,v^1_{n+1}),r_1} = -Q$, for $i \in V^+$. For all $a = (v^1_i, v^2_j) \in A^1$, $q_{a,r_2} = t_{ij} + st_i$, where $t_{n+1,i} = t_{0,i}$, $t_{i,n+1} = t_{i,0}$, for all $i \in V^+$, and $st_{n+1} = st_0$. Let $\delta^-(j)$ be the set of incoming arcs in G for each node $j \in V^+$. The following MIP formulation uses integer variables x_a, $a \in A$, to indicate the number of times that the arc a is traversed.

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

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

$M(x_{(i,j)}) = \{(v^1_i, v^1_j)\}$, for $(i,j) \in A$ with $i, j \neq 0$. $M(x_{(i,0)}) = \{(v^1_i, v^1_0), (v^1_i, v^1_{n+1})\}$, for $i \in V^+$. $M(x_{(0,i)}) = \{(v^1_0, v^1_i), (v^1_{n+1}, v^1_i)\}$, for $i \in V^+$. $L^1 = 0$, $U^1 = K$. $\mathcal{P}^V = \cup_{i \in V^+}\{\{v^1_i\}\}$. Rounded capacity cut separator on $((\cup_{i \in V^+}\{(\{v^1_i\}, w_i)\}, Q)$. Branching on variables x. Enumeration is on.

We remark that specifying negative consumption $-Q$ for capacity resource on arcs incoming to vertex v^1_{n+1} allows us to model resetting of accumulated capacity resource consumption to zero when visiting the depot between two trips. This happens due to the disposability property of the capacity resource: any negative accumulated consumption of this resource is automatically updated to zero, i.e., to the lower bound on accumulated capacity resource consumption for vertex v^1_{n+1}. Similar modelling scheme is employed for other problems we consider below.

As we mention in Section 2, in addiction to time windows, some other characteristics can be considered in a multi-trip VRP such as service-dependent loading times, limited trip duration, release dates, etc. However, in the variant addressed in this paper, CMTVRPTW, only the time windows is considered. The current version of the VRPSolver is not able to model in an efficient way the other multi-trip variants CMTVRPTW-LD, CMTVRPTW-R and CMTVRPTW-LT. In CMTVRPTW-LD, trip duration depends on the starting time of the route. In CMTVRPTW-R, route feasibility depends on the starting time of the route. Finally, in CMTVRPTW-LT, the service time of at the depot depends on the demands of the subset of customers that are served in the route. For the moment, the dependency between resources can only be modelled using tricks that lead to very large models and a non-competitive solver efficiency.

4.1.2 VRPSolver model example for the CMTVRPTW

Consider a CMTVRPTW instance with $n = K = 2$, $w_1 = w_2 = 1$, $Q = 1$, $[a_0, b_0] = [a_1, b_1] = [a_2, b_2] = [0, 4]$, $s_1 = s_2 = 1$, $c_{ij} = t_{ij} = 1$ for each $(i,j) \in A$. The path generator graph $G^1 = (V^1, A^1)$ is illustrated by Figure 1. The intervals for accumulated consumption of resources r_1 and r_2 are shown close to each vertex. For each arc, we show the consumption of both resources, denoted by q_1 and q_2. We also indicate the variable which is mapped to the arc.

Set P is composed of feasible paths $p_1 = (v^1_0, v^1_1, v^1_0)$, $p_2 = (v^1_0, v^1_2, v^1_0)$, $p_3 = (v^1_0, v^1_1, v^1_3, v^1_2, v^1_0)$, $p_4 = (v^1_0, v^1_2, v^1_3, v^1_1, v^1_0)$, $p_5 = (v^1_0, v^1_1, v^1_3, v^1_1, v^1_0)$, $p_6 = (v^1_0, v^1_2, v^1_3, v^1_2, v^1_0)$. The complete formulation (1) for the instance is the following.

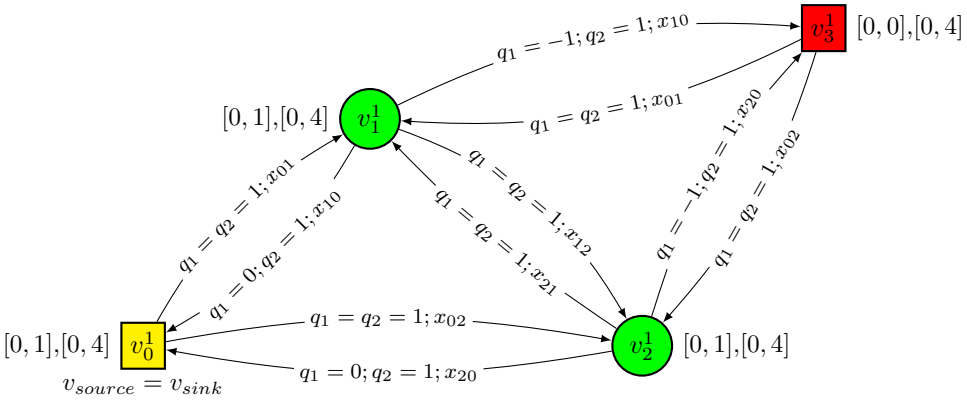

Figure 1: Path generator graph for the illustrative CMTVRPTW instance.

$$\begin{aligned}
\min\ & x_{01} + x_{10} + x_{02} + x_{20} + x_{12} + x_{21} && \text{(5a)}\\
\text{s.t.}\ & x_{01} + x_{21} = 1 && \text{(5b)}\\
& x_{02} + x_{12} = 1 && \text{(5c)}\\
& x_{01} = \lambda_1 + \lambda_3 + \lambda_4 + 2\lambda_5 && \text{(5d)}\\
& x_{10} = \lambda_1 + \lambda_3 + \lambda_4 + 2\lambda_5 && \text{(5e)}\\
& x_{02} = \lambda_2 + \lambda_3 + \lambda_4 + 2\lambda_6 && \text{(5f)}\\
& x_{20} = \lambda_2 + \lambda_3 + \lambda_4 + 2\lambda_6 && \text{(5g)}\\
& x_{12} = 0 && \text{(5h)}\\
& x_{21} = 0 && \text{(5i)}\\
& 0 \leq \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 \leq 2 && \text{(5j)}\\
& x_{01}, x_{10}, x_{02}, x_{20}, x_{12}, x_{21} \in \mathbb{Z}_+ && \text{(5k)}\\
& \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6 \in \mathbb{Z}_+ && \text{(5l)}
\end{aligned}$$

Variables λ_5 or λ_6 cannot take a positive value in any feasible solution for (5). This happens because paths p_5 and p_6 visit a customer twice. Such paths are called non-elementary paths and they never appear in an integer feasible solution. Presence of these paths make the linear relaxation or master formulation weaker. BCP algorithm implemented in VRPSolver uses *ng*-path relaxation [5] that eliminates many non-elementary paths which tend to appear in a solution of the master formulation. *ng*-path relaxation, limited-memory rank-1 cuts [32] and other state-of-art BCP components use the information provided by the collection $\mathcal{P}^V$ of packing sets. For this instance, there are two packing sets $\{v_1^1\}$ and $\{v_2^1\}$. These packing sets are implied by the following constraints corresponding to (3) which are valid for (5).

$$\begin{aligned}
& \lambda_1 + \lambda_3 + \lambda_4 + 2\lambda_5 \leq 1 && \text{(6a)}\\
& \lambda_2 + \lambda_3 + \lambda_4 + 2\lambda_6 \leq 1 && \text{(6b)}
\end{aligned}$$

In a feasible solution to Formulation (5), variable λ_3 takes value one, and other λ variables take value zero. λ_3 is associated with path $p_3 = (v_0^1, v_1^1, v_3^1, v_2^1, v_0^1)$ composed of two trips. The first trip visits only customer 1 while the second one only visits customer 2.

The master formulation solved by column generation for this instances is the following.

$$\min \ 2\lambda_1 + 2\lambda_2 + 4\lambda_3 + 4\lambda_5 + 4\lambda_6 \tag{7a}$$
$$\text{s.t.} \ \lambda_1 + \lambda_3 + \lambda_4 + 2\lambda_5 = 1 \tag{7b}$$
$$\lambda_2 + \lambda_3 + \lambda_4 + 2\lambda_6 = 1 \tag{7c}$$
$$0 \leq \lambda_1 + \lambda_2 + \lambda_3, \lambda_4 + \lambda_5 + \lambda_6 \leq 2 \tag{7d}$$
$$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6 \geq 0 \tag{7e}$$

4.2 Multi-depot vehicle routing problem with inter-depot routes

The MDVRPI is formally defined as follows. Given a directed graph $G = (V, A)$, the vertex set $V = V^+ \cup V_d$ consists of the set $V^+ = \{1, ..., n\}$ of customers and the set $V_d = \{n + 1, ..., n + m\}$ of depots. The set of arcs is defined as $A = \{(i, j), (j, i) : \ i \in V_d, \ j \in V^+\} \cup \{(i, j) : \ i, j \in V^+, \ i \neq j\}$. For each customer $i \in V^+$, a demand $w_i > 0$ and a service time $st_i \geq 0$ are given. For each arc $(i, j) \in A$, a travel time $t_{ij} > 0$ is known. A fleet of K homogeneous vehicles of capacity Q is available. In a route, a vehicle can visit any depot (including the depot where the route is started) in order to fully replenish its capacity. Thus each depot can be viewed as an IF in MDVRPI. The time spent for replenishing the vehicle capacity at a depot $d \in V_d$ is called docking time and denoted by $\pi_d > 0$. The docking time is also applied in the beginning of a route. A route is feasible if i) it starts and ends at a same depot $d \in V_d$; ii) the total demand served between two consecutive visits to depots does not exceeds vehicle capacity Q; and iii) the total route time (sum of docking times, service times and travel times) does not exceed a given upper bound T_{max}. The objective of the MDVRPI is to find at most K feasible routes of minimum total travel time such that each customer is visited exactly once by a single vehicle.

4.2.1 VRPSolver model for the MDVRPI

We define a path generator graph $G^k = (V^k, A^k)$ for each depot $k \in V_d$, where $V^k = \{v_i^k : 0 \leq i \leq n + m\}$ and $A^k = \{(v_i^k, v_j^k), (v_j^k, v_i^k) : \ i \in V_d \cup \{0\}, \ j \in V^+\} \cup \{(v_i^k, v_j^k) : \ i, j \in V^+\}$. Node v_0^k corresponds to the depot k when it is used as the starting point for the route. Vertices v_{n+1}^k,...,v_{n+m}^k correspond to depots $n + 1, \ldots, n + m$ which are used as IFs. Other vertices correspond to customers. We present below the complete description of the proposed VRPSolver model.

VRPSolver Model for MDVRPI. For each $k \in V_d$, a graph $G^k = (V^k, A^k)$ defined just above. $v^k_{\text{source}} = v^k_{\text{sink}} = v^k_0$. $R^k = \{r_1, r_2\}$, where resource r_1 ensures the vehicle capacity constraint, and r_2 guarantees that the maximum route total time is respected. $[l_{v^k_i,r_1}, u_{v^k_i,r_1}] = [0, Q]$ for $k \in V_d$ and $i \in V^+ \cup \{0\}$. $[l_{v^k_i,r_1}, u_{v^k_i,r_1}] = [0, 0]$, for $k \in V_d$ and $i \in V_d$. $[l_{v,r_2}, u_{v,r_2}] = [0, T_{max}]$ for $k \in V_d$ and $v \in V^k$. $q_{(v^k_i,v^k_j),r_1} = -Q$ for $k \in V_d$, $i \in V^+$, and $j \in V_d$. $q_{(v^k_i,v^k_0),r_1} = 0$, for $k \in V_d$ and $i \in V^+$. $q_{(v^k_i,v^k_j),r_1} = w_j$ for $k \in V_d$, $i \in V^k$ and $j \in V^+$. $q_{(v^k_0,v^k_j),r_2} = \pi_k + t_{kj} + st_j$ for $k \in V_d$ and $j \in V^+$. $q_{(v^k_i,v^k_0),r_2} = t_{ik}$ for $k \in V_d$ and $i \in V^+$. $q_{(v^k_i,v^k_j),r_2} = t_{ij} + st_j$ for $k \in V_d$, $i \neq 0$, and $j \in V^+$. $q_{(v^k_i,v^k_j),r_2} = t_{ij} + \pi_j$ for $k \in V_d$ and $j \in V_d$. As before, let $\delta^-(j)$ be the set of incoming arcs in G for each node $j \in V^+$. The following MIP formulation uses integer variables x_a for $a \in A$, to indicate the number of times that the arc a is traversed and integer variables y_k for $k \in V_d$, to indicate the number of routes that start and end at depot k.

$$\min \sum_{a \in A} t_a x_a \tag{8a}$$

$$\text{s.t.} \sum_{a \in \delta^-(j)} x_a = 1, \qquad j \in V^+, \tag{8b}$$

$$\sum_{k \in V_d} y_k \leq K; \tag{8c}$$

$M(x_{(i,j)}) = \{(v^k_i, v^k_j) : k \in V_d\}$ for $(i,j) \in A$ such that $i, j \in V^+$, $M(x_{(i,d)}) = \{(v^k_i, v^k_d) : k \in V_d\} \cup \{(v^d_i, v^d_0)\}$ for $i \in V^+$ and $d \in V_d$, $M(x_{(d,i)}) = \{(v^k_d, v^k_i) : k \in V_d\} \cup \{(v^d_0, v^d_i)\}$ for $d \in V_d$ and $i \in V^+$. $M(y_k) = \{(v^k_0, i) : i \in V^+\}$ for $k \in V_d$; $L^k = 0$ and $U^k = K$ for $k \in V_d$. $\mathcal{P}^V = \cup_{i \in V^+}\{\{v^k_i\}_{k \in V_d}\}$. Rounded capacity cut separator on $(\cup_{i \in V^+}\{(\{v^k_i\}_{k \in V_d}, w_i)\}, Q)$. Branching on x and y variables. Enumeration is on.

4.2.2 VRPSolver model example for the MDVRPI

Consider a MDVRPI instance with $n = m = K = 2$, $w_1 = w_2 = 1$, $Q = 1$, $T_{max} = 8$, $st_1 = st_2 = \pi_3 = \pi_4 = 1$, and $t_{ij} = 1$ for $(i,j) \in A$. There are two path generator graphs $G^3 = (V^3, A^3)$ and $G^4 = (V^4, A^4)$. Graph G^3 is illustrated by the Figure 2. As in the previous example, the intervals for accumulated consumption of resources r_1 and r_2 are shown close to each vertex. For each arc, we show the consumption of the both resources, denoted by q_1 and q_2. We also indicate the variable which is mapped to the arc.

4.3 Arc Routing Problem With Intermediate Facilities Under Capacity and Length Restrictions

The CLARPIF is formally defined as follows. We are given an undirected graph $G = (V, E)$. For each edge $e \in E$, a cost (or distance) $c_e \geq 0$ and a demand $w_e \geq 0$ are defined. We define the set E' of required edges, i.e., edges with positive demand: $E' = \{e \in E : w_e > 0\}$. Each edge can be traversed any number of times, except that required edges should be traversed at least once in order to collect the demand. An unlimited number of homogeneous vehicles with capacity Q is available. Vertex set V contains the depot vertex 0 and a set $F \subseteq V$ of IFs. The CLARPIF consists in finding the set of routes of minimum total cost under the following constraints.

- Each route starts and ends at the depot.
- The demand of every required edge $e \in E'$ is collected by exactly one vehicle.
- In each route, the total demand collected between two consecutive visits of the depot or an IF does not exceed vehicle capacity Q.

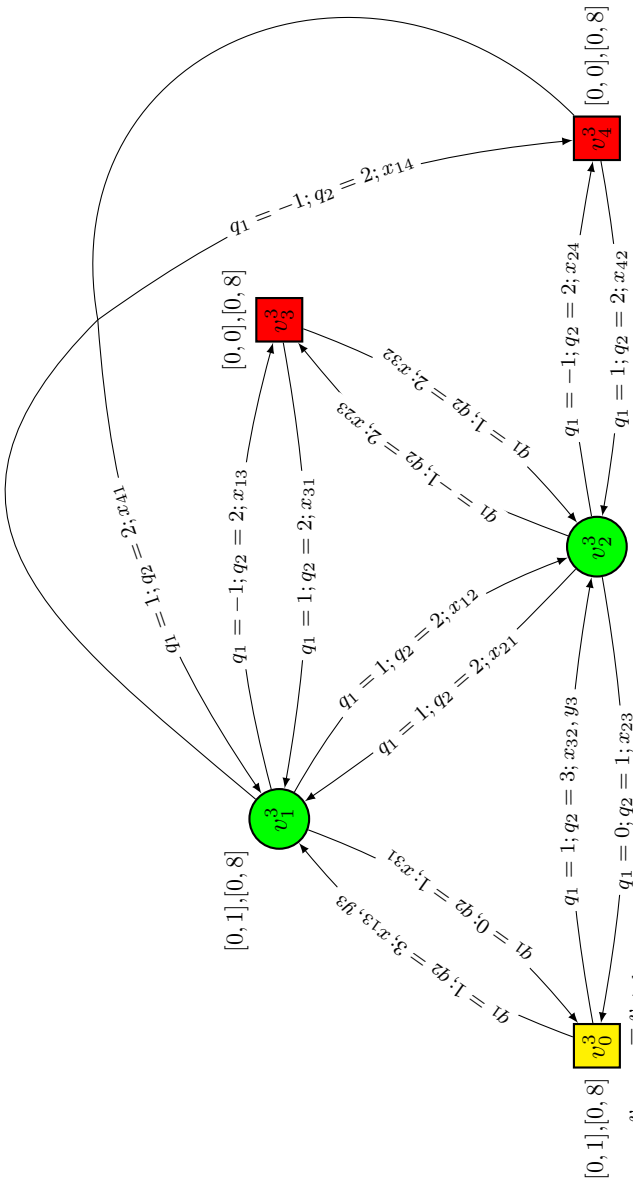

Figure 2: Path generator graph $G^3(V^3, A^3)$ for the illustrative MDVRPI instance.

- If $\{0\} \notin F$ then no demand can be collected between the last visit to an IF and returning to the depot.
- The total distance of any route does not exceed a given upper limit T_{max}.

4.3.1 VRPSolver model for the CLARPIF

Our model for the CLARPIF is an extension of the VRPSolver model for the Capacitated Arc Routing Problem proposed by [33]. First, we introduce some notations. For $i, j \in V$ let $D(i,j) \subseteq E$ be the set of edges in a cheapest path from i to j in graph G. The cost of this path is denoted as $C(i,j) = \sum_{e \in D(i,j)} c_e$. We define a dummy edge $b_0 = (0, 0')$ with $w_{b_0} = 0$ and $c_{b_0} = 0$ and a set of dummy edges $F_0 = \{(i, i') : i \in F\}$ with $w_{b_i} = -Q$ and $c_{b_i} = 0$ for $(i, i') \in F_0$. We denote $E_0 = E' \cup \{b_0\} \cup F_0$. For each $e \in E_0$ we define the set S_e of possible traversal directions. If $e = (i,j) \in E'$ then two directions are possible: $S_e = \{i, j\}$. If $e = (i, i') \in \{b_0\} \cup F_0$ then one "direction" is possible $S_e = \{i\}$.

We define a single path generator graph $G^1 = (V^1, A^1)$. This graph contains vertices for every required or dummy edge $e \in E_0$ and every traversal direction $s \in S_e$: $V^1 = \{v_e^s : \ e \in E_0, \ s \in S_e\}$. Set A^1 contains: i) arcs which connect vertices corresponding to edges in $E' \cup F_0$ such that at least one edges is required: $\{(v_e^s, v_{e'}^{s'}) : \ e, e' \in E' \cup F_0, \ s \in S_e, \ s' \in S_{e'}, \ e \in E'$ or $e' \in E', e \neq e'$ or $s \neq s'\}$; ii) arcs from vertex $v_{b_0}^0$ to all vertices corresponding to required edges: $\{(v_{b_0}^0, v_e^s) : \ e \in E', \ s \in S_e\}$; and iii) arcs from vertices corresponding intermediate facilities to vertex $v_{b_0}^0$ if the depot is not an IF: $\{(v_e^s, v_{b_0}^0) : \ e \in F_0, \ s \in S_e, \ \{0\} \notin F\}$.

VRPSolver Model for CLARPIF. A single graph $G^1 = (V^1, A^1)$, where V^1 and A^1 are defined just above. $v_{source}^1 = v_{sink}^1 = v_{b_0}^0$. $R^1 = \{r_1, r_2\}$, where resource r_1 ensures the vehicle capacity constraints, and resource r_2 guarantees that the total route distance does not exceed the upper limit. $[l_{v,r_1}, u_{v,r_1}] = [0, Q]$ for $v \in V^1 \backslash \{v_{sink}\}$. $[l_{v_{sink},r_1}, u_{v_{sink},r_1}] = [0, 0]$ in order to ensure that the vehicle finishes the route empty according to the CLARPIF definition. $[l_{v,r_2}, u_{v,r_2}] = [0, T_{max}]$ for each $v \in V^1$. For every arc $a = (v_e^s, v_{e'}^{s'}) \in A^1$, $q_{a,r_1} = w_{e'}$. For every arc $a = (v_e^s, v_{e'}^{s'}) \in A^1$, $q_{a,r_2} = C'_a = C(s, \bar{s}'(e')) + c_{e'}$, where $\bar{s}'(e')$ is the extremity of e' different from s' if $e' \in E'$, and $\bar{s}'(e') = s'$ if $e' \notin E'$. Let $\delta^-(v)$ denote the set of incoming arcs in A^1 entering a vertex $v \in V^1$. Let also $\delta^-(e) = \cup_{s \in S_e} \delta^-(v_e^s)$. The following MIP formulation uses integer variable x_a for each $a \in A^1$, to indicate the number of times that the arc a is traversed. The formulation follows.

$$\min \quad \sum_{a \in A^1} C'_a x_a \tag{9a}$$

$$\text{s.t.} \quad \sum_{a \in \delta^-(e)} x_a = 1, \qquad e \in E'; \tag{9b}$$

$M(x_a) = \{a\}, a \in A^1$; $L^1 = 0$ and $U^1 = |E'|$. $\mathcal{P}^V = \cup_{e \in E'} \{\{v_e^s\}_{s \in S_e}\}$. Rounded capacity cut separator on $\left(\cup_{e \in E'} \left\{\left(\{v_e^s\}_{s \in S_e}, w_e\right)\right\}, Q\right)$. We first branch on the aggregations of x variables corresponding to node degrees in original graph G. We then branch on the aggregations of x variables corresponding to pairs of required or dummy edges $e, e' \in E_0$, $e \neq e'$: $\sum_{s \in S_e} \sum_{s' \in S_{e'}} \left(x_{(v_e^s, v_{e'}^{s'})} + x_{(v_{e'}^{s'}, v_e^s)}\right)$. Finally, we branch on variables x.

4.3.2 A VRPSolver model example for the CLARPIF

Consider a CLARPIF instance with original graph $G = (V, E)$, $V = \{0, 1, 2\}$, $v_0 = 0$, $E = \{(0,1), (0,2), (1,2))\}$, $E' = \{(1,2), (0,2)\}$, $w_{(1,2)} = w_{(0,2)} = 1$, $F = \{2\}$, $Q = 1$, $c_e = 1$ for each $e \in E$, $T_{max} = 5$. Figure 3 presents a graph representation for G where we marked depot vertex 0 in yellow and vertex 2 corresponding to the intermediate facility in red. We also indicate the

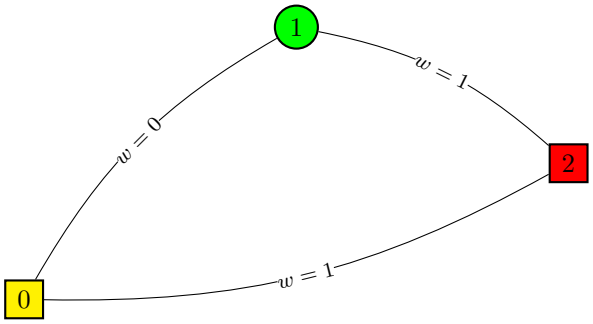

Figure 3: A graph representation for the illustrative CLARPIF instance.

respective demand on each edge.

A feasible solution for the proposed illustrative CLARPIF instance is the single route $0 \rightarrow 1 \Rightarrow 2 \Rightarrow 0 \rightarrow 2 \rightarrow 0$, where demand collection happens during traversals "$\Rightarrow$". In this solution, a single vehicle with capacity $Q = 1$ can collects two demand units, as it visits the intermediate facility between collecting them. The vehicle has to visit vertex 2 again before returning to the depot, as the second collected demand unit cannot be dropped at the depot which is not an intermediate facility.

For this instance, we define $E_0 = E' \cup \{(0, 0')\} \cup \{(2, 2')\}$, with $w_{(0,0')} = c_{(0,0')} = c_{(0,0')} = 0$, and $w_{(2,2')} = -Q = -1$. The set of vertices V_1 in the path generator graph contains two vertices for edges $(1, 2)$ and $(0, 2)$ and one vertex for edges $(0, 0')$ and $(2, 2')$. Thus, we have $V^1 = \{v^0_{(0,0')}, v^1_{(1,2)}, v^2_{(1,2)}, v^0_{(0,2)}, v^2_{(0,2)}, v^2_{(2,2')}\}$, where $v_{source} = v_{sink} = v^0_{(0,0')}$. The set of arcs A^1 is built in the following way. Let $e = (i, j)$ and $e' = (i', j')$ be two different edges in E_0. If $e \in E'$ and $e' \in E'$ then we add eight arcs to A^1 between vertices corresponding to these two edges. If $e = (i, i') \notin E'$ and $e' \in E'$ then we add four arcs to A^1. Path generator graph G^1 for the proposed instance is illustrated in Figure 4. Again, the intervals for accumulated consumption of resources r_1 and r_2 are shown close to each vertex. Due to the lack of space in the figure, we show the consumption of the both resources, denoted by q_1 and q_2, only for arcs participating in the feasible solution $0 \rightarrow 1 \Rightarrow 2 \Rightarrow 0 \rightarrow 2 \rightarrow 0$ presented above.

Route $0 \rightarrow 1 \Rightarrow 2 \Rightarrow 0 \rightarrow 2 \rightarrow 0$ for the CLARPIF corresponds to path $\left(v^0_{(0,0')}, v^2_{(1,2)}, v^2_{(2,2')}, v^0_{(0,2)}, v^2_{(2,2')}, v^0_{(0,0')}\right)$ in graph G^1. Arc $\left(v^0_{(0,0')}, v^2_{(1,2)}\right)$ corresponds to taking the shortest path to vertex 1, and then following required edge $(1, 2)$ while collecting the demand. Arc $\left(v^2_{(1,2)}, v^2_{(2,2')}\right)$ corresponds to staying at vertex 2 and then dropping the demand at IF 2. Arc $\left(v^2_{(2,2')}, v^0_{(0,2)}\right)$ corresponds to following required edge $(0, 2)$ while collecting the demand. Arc $\left(v^0_{(0,2)}, v^2_{(2,2')}\right)$ corresponds to taking the shortest path from vertex 0 to vertex 2 and dropping the collected demand at IF 2. Finally, arc $\left(v^2_{(2,2')}, v^0_{(0,0')}\right)$ corresponds to returning to the depot by taking the shortest path from vertex 2.

4.4 Green Vehicle Routing Problem

The G-VRP is formally defined as follows. Given a complete graph $G = (V, A)$, the vertex set $V = \{0\} \cup V^+ \cup F$ contains the depot vertex 0, a customer set $V^+ = \{1, ..., n\}$ and an alternative fuel station (AFS) set $F = \{n+1, ..., n+m\}$. For each customer a service time $st_i \geq 0$ is defined. For each arc $a \in A$ a travel time t'_a and a distance d_a are defined. The customers are served by Alternative Fuel Vehicles (AFVs). Each AFV can travel a maximum distance D without refuelling. The AFVs can stop at a AFS to refuel, incurring a refuelling delay of duration π. The refuelling delay also occurs when the vehicles leave the depot. To simplify the notation, we include the service and refueling times in the arc travel times. Thus, the complete travel times

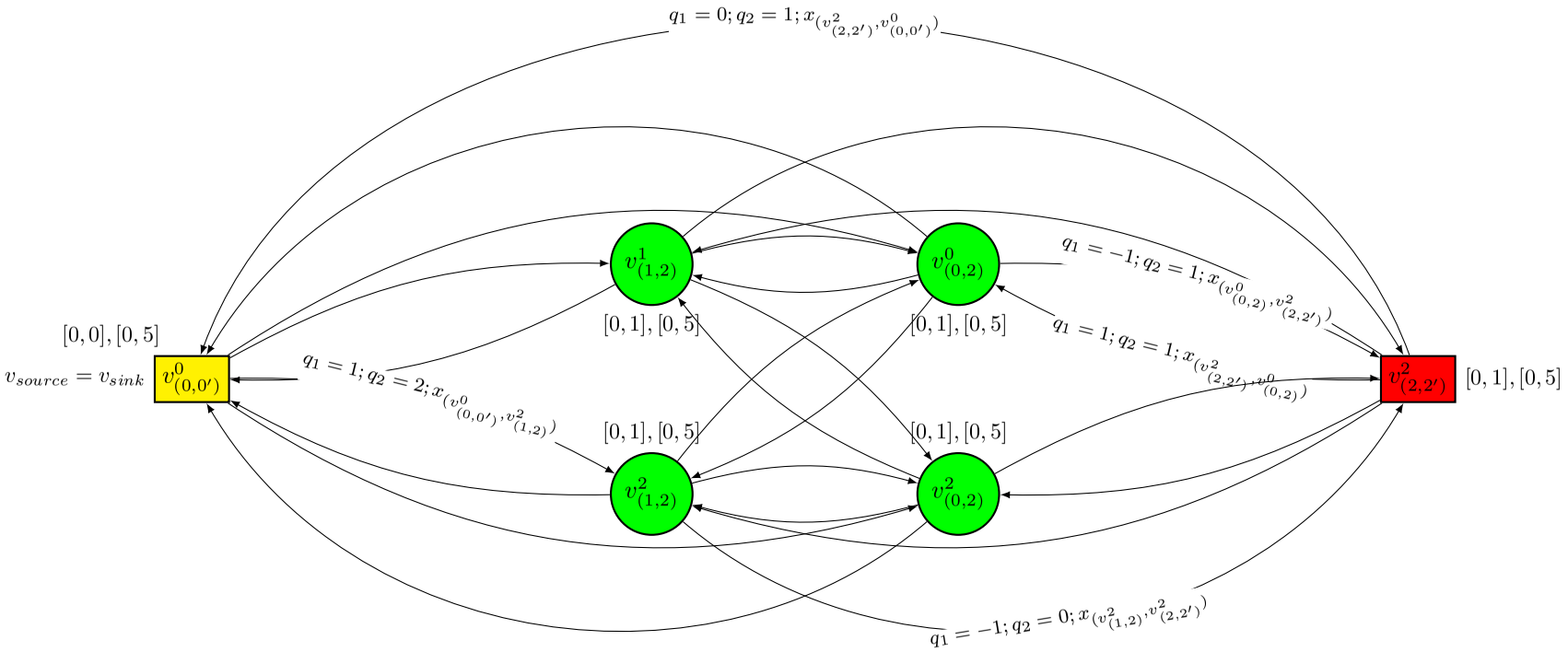

Figure 4: Path generator graph G^1 for the proposed CLARPIF example.

of an arc $a = (i, j)$ equals $t_{(i,j)} = t'_{(i,j)} + st_i$ when $i \in V^+$, or equals $t_{(i,j)} = t'_{(i,j)} + \pi$ when $i \in F \cup \{0\}$. The G-VRP consists of finding a set of routes which minimizes the total travelled distance under the following constraints.

- Each route starts and ends at the depot.
- Each customer is visited by exactly one AFV.
- In each route, the total distance travelled between two successive AFS does not exceed the maximum distance without refueling D.
- The total travel time in any route does not exceed a given bound T_{max}.

4.4.1 VRPSolver model for the G-VRP

Now we present our VRPSolver model for the G-VRP.

VRPSolver Model for G-VRP. A single path generator complete graph $G^1 = (V^1, A^1)$ where $V^1 = V$ and $A^1 = A$. $v^1_{\text{source}} = v^1_{\text{sink}} = 0$. $R^1 = \{r_1, r_2\}$, where time resource r_1 ensures that the maximum route total travel time is respected, and resource r_2 guarantees that the maximum travelled distance between two refuels is respected.. $[l_{i,r_1}, u_{i,r_1}] = [0, T_{max}]$, $i \in V^1$. $[l_{i,r_2}, u_{i,r_2}] = [0, D]$ for $i \in V^1 \setminus F$, and $[l_{i,r_2}, u_{i,r_2}] = [0, 0]$ for $i \in F$. $q_{a,r_1} = t_a$ for $a \in A^1$. $q_{a,r_2} = d_a$ for $a = (i, j) \in A^1$ with $j \notin F$. $q_{a,r_2} = d_a - D$ for $a = (i, j) \in A^1$ with $j \in F$. Let $\delta^-(i)$ denote the set of incoming arcs to a vertex $i \in V^+$. The following MIP formulation uses integer variables x_a, $a \in A$, to indicate the number of times that the arc a is traversed.

$$\min \sum_{a \in A} d_a x_a \tag{10a}$$

$$\text{s.t.} \sum_{a \in \delta^-(i)} x_a = 1, \qquad i \in V^+; \tag{10b}$$

$M(x_a) = \{a\}$, for $a \in A^1$; $L^1 = 0$ and $U^1 = |V^+|$. $\mathcal{P}^v = \cup_{i \in V^+}\{\{i\}\}$. We then branch on the aggregations of x variables corresponding to incoming arcs to the depot: $\sum_{a \in \delta^-(0)} x_a$. Finally, we branch on variables x. Enumeration is on.

For the particular case where the distance and travel time matrices are symmetric, we do the following changes: we create a variable $x_{(i,j)}$ only if $i < j$, we replace the constraints (10b) by $\sum_{a \in \delta(i)} x_a = 2, \forall i \in V_+$, where $\delta(i)$ represents the set of edges adjacent to i and we replace the mapping set M by $M(x_{(i,j)}) = \{(i, j), (j, i)\}$, $(i, j) \in A^1$.

4.4.2 An example for the G-VRP VRPSolver model

Consider a G-VRP instance with $n = 3$, $m = 1$, $T_{max} = 8$, $D = 2$, $\pi = st_1 = st_2 = 1$, $t'_a = d_a = 1$ for $a \in A$. Thus, we have $t_a = 2$ for all $a \in A$. The path generator graph for this instance is illustrated in Figure 5. The intervals for accumulated consumption of resources r_1 and r_2 are shown close to each vertex. For each arc, we show the consumption of the both resources, denoted by q_1 and q_2. We also indicate the variable which is mapped to the arc. Depot, customers and AFS are coloured in yellow, green, and red, respectively.

5 Computational Experiments

In this section, we present computational results for the four problems addressed in this paper. All experiments are performed on a computer with an Intel Core i7-4790 processor with 3.6 GHz

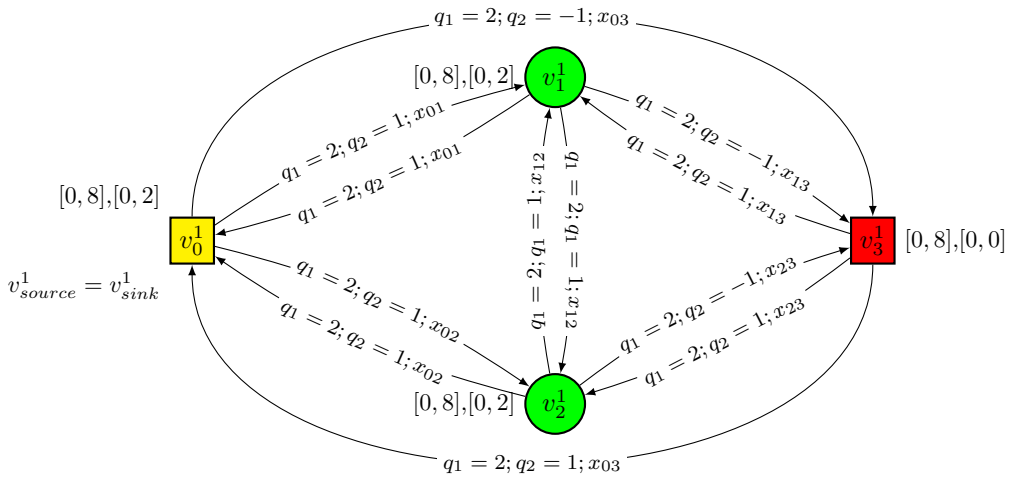

Figure 5: Path generator graph for the illustrative GVRP instance.

Problem	Best exact method	Time limit(s)	Initial *ub*
CMTVRPTW	[44]	3h	[44]
MDVRPI	[40]	2h	This paper
CLARPIF	-	2h	[41]
G-VRP	[1]	10h	[1]

Table 1: Best exact method, time limit and Initial upper bound for each problem

and 16 GB of RAM on Ubuntu 18.04.2 LTS operating system. Each instance is solved using a single thread. CPLEX v12.9 is used as the LP and MIP solver. For each problem, we compare our approach with the best exact algorithm available in the literature. Each instance is solved with and without the information about a known feasible solution. Table 1 presents for each problem the reference with the best known exact method, the time limit in hours for solving each instance, and the source we used to obtain the known feasible solutions. For the MDVRPI, we use the feasible solutions obtained by ourselves in preliminary runs, as a part of instances we use were not previously considered in the literature.

In the reminder of this section, we present aggregated results for each instances class. Each class is characterised by a pair of instance group name and instance size. Detailed computational results for each instance are presented in the online appendix. The tables in this section have the same structure which is the following. Each line corresponds to one instance class. Column *Group* indicates the instance group name. Column *#Inst* gives the number of instances in the class. The next column(s) specify the instance size in the class. Size units depend on the problem. Column *#Int. Solved* indicates the number of instances in the class solved to optimality within the time limit. Columns *Avg.* $Gap_0(\%)$ indicates the average root gap among optimally solved instances in the class. The root gap for an instance is the difference between the value of the best known solution and the value of the lower bound obtained at the end of the root node, taken in per cent from the value of the best known solution. Column *Avg. #Nodes* gives the average number of nodes in the branch-and-bound tree among optimally solved instances in the class. Finally, column *Time(s)* shows the average total CPU time in seconds among instances in the class solved to optimality within the time limit.

5.1 Results for the CMTVRPTW instances

The test instances for the CMTVRPTW are generated from the standard VRPTW instances by Solomon [37]. As it is common in the literature, only instances of type 2 are used. The set of instances is divided in three groups: C, R and RC with respectively 8, 11 and 8 instances for each size. Instances with different sizes are used: those with $n \in \{25, 40, 50, 70, 80, 100\}$. Instances with $n < 100$ are obtained by taking the first n customers from original Solomon instances with 100 customers. The number of available vehicles K is set to two for $n = 25$, four for $n = 40$ and $n = 50$, six for $n = 70$, seven for $n = 80$, and eight for $n = 100$. The vehicle capacity Q is fixed to 100 for all instances.

To our knowledge, the most efficient current exact algorithms for the CMTVRPTW are proposed by [31] and [44]. The latter algorithm defines the current state-of-the-art for the exact solution of the CMTVRPTW. We present the computational comparison of three algorithms in Table 2. The size of instances in each class is determined by the number of customers n. For the Yang's algorithm, we use its implementation available online: `github.com/Yu1423/CMTVRPTW`. This implementation as well as our algorithm were run on the same machine. The only difference is that the literature implementation uses Gurobi v9.1 as the LP and MIP solver while we use CPLEX v12.9 for these tasks. For the BCP algorithm proposed by [31], we present the original results from their paper which were obtained on a Windows server equipped with six virtual CPUs running at 2.59 GHz and with 16 GB of RAM. All three algorithms are use a single thread for each instance. The execution of algorithm by [44] was interrupted for several instances with 100 customers due to lack of available memory.

From the results presented in Table 2, it can be seen that our approach is more efficient in general than the state-of-the-art, as it solves more instances to optimality within the time limit even when no information about known feasible solutions is used. Our approach is particularly efficient for instances in group C (clustered instances). However, for instances in groups R and RC, some instances can be solved by [44] and cannot be solved by our approach.

Superiority of our algorithm can be explained by the fact that it incorporates the main state-of-the-art techniques known to be useful for many VRP variants. These techniques are listed in the beginning of Section 3. Existing algorithms often miss many of these techniques because they were not known at the moment of publication or because they are time-consuming to implement. In addition, our implementation is carefully designed to ensure the best possible performance.

When using the information about the best known upper bounds, the efficiency of our approach rises considerably, as more than 90% of instances can be solved to optimality. Any instance solved by [44], can also be solved by our approach when initialised with the best known solutions. This shows importance of this information for VRPSolver as well for any BCP approach. We would like also to highlight the small average size of the branch-and-bound tree generated by our algorithm.

The solution time of our algorithm is larger for small instances with 25 customers than for larger instances with 40 customers. This happens because we use a parameterisation of VRPSolver, which is adapted for difficult instances.

5.2 Results for the MDVRPI instances

For the MDVRPI, the test instances are generated from two groups of original instances proposed by [12] and available online: `chairelogistique.hec.ca/data/mdvrpi`. The number of original instances in groups *Random* and *CGL* is 12 and 10 respectively. They have between 48 and 288 customers and between three and seven depots. We generate instances of different sizes, i.e., with different number of customers n, by taking the first 25, 40, 70, 100, and 120 customers in the original instances that have an enough number of customers. For example if for some original instance file the number of customers is 48 then we create two instances: one of them with 25 customers and another one with 40 customers. The number of vehicles K is fixed to 4, 6, 10, 15, and 17, respectively for each number of customers. The values of Q and T_{max} are set

			Proposed BCP without initial ub				Proposed BCP with initial ub				[31]		[44]	
			#Inst.	Avg.			#Inst.	Avg.			#Inst.	Avg.	#Inst.	Avg.
Group	#Inst	n	Solved	Gap_0(%)	#Nodes	Time(s)	Solved	Gap_0(%)	#Nodes	Time(s)	Solved	Time (s)	Solved	Time (s)
C	8	25	8	0.0	1.0	174.2	8	0.0	1.0	1.8	8	16.1	8	3.3
R	11	25	11	0.1	1.4	38.1	11	0.0	1.0	2.7	11	83.1	11	8.1
RC	8	25	8	0.1	2.3	142.1	8	0.0	1.0	5.6	8	175.7	8	52.4
C	8	40	8	0.1	2.5	32.3	8	0.1	2.0	26.7	6	2589.9	8	135.0
R	11	40	11	0.2	2.1	93.6	11	0.0	1.0	15.0	10	377.7	11	11.0
RC	8	40	8	0.1	2.0	40.2	8	0.0	1.5	17.4	8	1223.7	8	27.3
C	8	50	8	0.1	3.3	325.2	8	0.1	1.3	62.0	2	1601.1	8	57.4
R	11	50	11	0.7	20.8	3218.1	11	0.3	5.5	1266.4	0	-	11	1284.7
RC	8	50	7	0.5	5.9	162.3	8	0.6	28.8	834.9	7	647.6	8	48.4
C	8	70	8	0.2	4.8	781.0	8	0.2	4.8	796.7			3	3949.3
R	11	70	7	0.6	15.6	2832.8	9	0.5	24.1	2880.6			9	1164.2
RC	8	70	7	0.3	18.7	2905.6	8	0.3	10.3	2416.3			8	1711.1
C	8	80	8	0.0	2.0	191.9	8	0.0	1.0	178.3			5	4809.7
R	11	80	4	0.6	25.5	4611.2	9	0.5	25.0	4224.2			6	3335.6
RC	8	80	6	0.2	8.0	2081.7	8	0.2	7.3	613.3			8	1241.2
C	8	100	8	0.1	2.8	705.2	8	0.1	1.8	456.1			1	9041.3
R	11	100	1	0.3	15.0	5279.8	3	0.4	29.7	3904.6			1	5225.4
RC	8	100	2	0.2	7.0	837.5	6	0.4	38.0	5586.7			1	120.9
All	162		131	0.2	6.8	1031.9	148	0.2	9.0	1125.5			123	986.4

Table 2: Summary of our results for CMTVRPTW instances.

respectively to 50 and 450. Other data comes from the original instances. Instances with 100 and 120 customers are considered the first time in the literature. Smaller instances with 25, 40, and 70 customers were already used by [29] and [40].

To our knowledge, the best exact approaches in the literature for the MDVRPI are a branch-and-price algorithm by [29] and a BCP algorithm by [40]. The latter algorithm defines the current state-of-the-art for the exact solution of the MDVRPI. The results for the algorithm by [29] were obtained on a computer with a 2.67 GHz Intel Westmere-EP X5650 Processor and 4 GB of RAM. The authors used the CPLEX v12.2 as a LP and MIP solver. The results for the algorithm by [40] were obtained on an Intel Xeon CPU E5-2689 v4 server running at 3.10 GHz with 128GB of RAM. We took at the web site `www.cpubenchmark.net` the score of the single thread performance of the processor used by the literature algorithms and used by ours: 1306, 2380 and 2231 points for the algorithms proposed by [29], [40] and us, respectively. The processor speed for the last two algorithms is very similar, and the efficiency of the first algorithm is clearly not on pair with others even considering the difference in processor speed. Therefore, we do not do any adjustments to the solutions times in the computational comparison of three algorithms presented in Table 3.

From the results in Table 3, one can see that our approach solves more instances than the best literature one by [40], even when no information about known feasible solution is used. 89% of all instances are solved to optimality within the time limit, including some large instances with 120 customers. However, smaller instances with up to 40 customers are solved faster by the best algorithm in the literature.

Again, when using the information about the best known solutions, the solution time our approach decreases by one order of magnitude. Nevertheless, no additional instances are solved to optimality. No solutions are known in the literature for unsolved instances with 100 and 120 customers. We suppose that efficiency of our approach will increase considerably when good solutions are obtained by heuristics.

5.3 Computational results for CLARPIF instances

For the CLARPIF, we use the test instances proposed by [41] and available online: `sites.google.com/site/wasteoptimisation`. These instances were obtained by modification of two groups of original instances for the capacitated arc routing problem (CARP). Group *gdb* was proposed by [22], and group *bccm* was proposed by [7]. There 23 modified instances with up to 55 required edges in the group *gdb-IF*, and 34 instances with up to 92 required edges in the group *bccm-IF*. We refer to [41] for more details about the instances.

The Table 4 summarises only our computational results for the CLARPIF, as there is no exact algorithm for this problem in the literature to our knowledge. Column *Avg.* $|V|$ specifies the average number of nodes in the instances of each group. Column *Avg.* $|E'|$ gives the average number of required edges in instances of each group.

From 57 test instances, our approach is able to solve 53 instances (or 93%) to optimality within the time limit, when initialised with known solutions presented by [41]. When the information about best known solutions is not used, we optimally solved all 23 instances in group *gdb-IF*. Thus, the average optimality gap of these solutions is equal to zero. On the other hand, for the 34 instances in group *bccm-IF*, we did not optimally solve 10 instances. For all of these instances, we did not find any feasible solution. Therefore, we could not calculate their optimality gaps. Instances in group *bccm-IF* are harder to solve to optimality. This is probably due to the fact that these instances are larger.

It is worth pointing out that the heuristic algorithm proposed by [41] found the optimum values for 14 out of 23 instances in group *gdb-IF*. The average optimality gap of their heuristic solutions equals to 1.59% for these instances. For the instances in group *bccm-IF*, the heuristic found the optimum or best values obtained by us for 31 out of 34 instances, and obtained the average gap of 0.88%. These results confirm a very good efficiency of the heuristic proposed by

			Proposed BCP without initial ub				Proposed BCP with initial ub				[29]		[40]	
			#Inst.	Avg.			#Inst.	Avg.			#Inst.	Avg.	#Inst.	Avg.
Group	n	#Inst	Solved	Gap_0(%)	#Nodes	Time(s)	Solved	Gap_0(%)	#Nodes	Time(s)	Solved	Time (s)	Solved	Time (s)
Random	25	12	12	0.5	2.5	23.9	12	0.0	1.0	2.1	7	451.9	12	17.5
CGL	25	10	10	0.5	3.2	31.8	10	0.0	1.0	2.3	8	1221.8	10	15.2
Random	40	12	12	0.2	3.5	131.5	12	0.0	1.0	6.8	1	4474.0	12	76.6
CGL	40	10	10	0.5	4.6	157.5	10	0.0	1.0	4.5	2	6383.5	10	81.1
Random	70	10	10	0.2	6.6	478.3	10	0.0	1.2	59.1	-	-	9	2728.0
CGL	70	9	9	0.3	7.0	540.8	9	0.0	1.0	62.5	-	-	8	1607.3
Random	100	6	5	0.3	15.0	3495.2	5	0.1	3.4	450.9	-	-	-	-
CGL	100	7	6	0.2	6.0	1893.8	6	0.0	1.0	145.4	-	-	-	-
Random	120	6	2	0.1	6.0	2650.1	2	0.0	1.0	193.7	-	-	-	-
CGL	120	7	3	0.2	4.0	2169.1	3	0.0	1.0	314.6	-	-	-	-

Table 3: Summary of our results for MDVRPI instances.

				Proposed BCP without initial ub				Proposed BCP with initial ub			
		Avg.		#Inst.	Avg.			#Inst.	Avg.		
Group	#Inst.	$\|V\|$	$\|E'\|$	Solved	$Gap_0(\%)$	#Nodes	Time(s)	Solved	$Gap_0(\%)$	#Nodes	Time(s)
gdb-IF	23	11.4	29.4	23	0.2	10.3	117.5	23	0.2	6.7	39.1
bccm-IF	34	35.0	63.3	24	1.5	24.7	581.3	30	1.0	9.7	379.9

Table 4: Summary of our results for CLARPIF instances.

[41]. Thus, we can highlight that evaluation of heuristics performance is an important use case of VRPSolver models. Such evaluation may be carried out using medium size instances which are out of reach for MIP models.

5.4 Computational results for G-VRP instances

For the G-VRP, the use two groups *AB1* and *AB2* of test instances proposed by [1] and available online: *www.vrp-rep.org/variants/item/g-vrp.html*. Each group contains 20 instances which have from 50 to 100 customers and from 21 to 25 refuelling stations.

The best exact algorithm in the literature to our knowledge was proposed by [1]. In Table 5 we compare computationally with it our approach. Column *Avg.* n presents the average number of customers for every group of instances. Columns *Avg.* s shows the average number of refuelling stations. The tests conducted by [1] did not consider a time limit. Their approach does not solve some instances due to insufficient memory when performing the route generation step in their algorithm.

It can be seen from Table 5 that our approach does not manage to outperform the state-of-the-art in general. However, we manage to solve two instances AB214 and AB216 to optimality for the first time, as indicated in the detailed results we present in the online appendix.

It is not surprising that the algorithm by [1] outperforms ours, as the former uses special k-path cuts. This family of cuts is different from the well-known rounded capacity cuts, which are sometimes also called k-path cuts. k-path cuts proposed by [1] are based on the maximum travelled distance between two successive AFS and on the maximum total route travel time. Combination of these constraints is specific to the G-VRP. Separation algorithm for these cuts is complex and requires a laborious implementation of the labelling algorithm for a particular RCSPP problem. Since the focus of this paper is on using generic tools, we decided not to use separation algorithms specific to a particular problem.

We highlight that the method proposed by [1] is not efficient without k-path cuts. As stated by the authors in [1], only one instance (AB101) in group AB1 could be solved by their algorithm with the same parameters and time limit settings if the k-path cuts are not used ignored. Thus, our algorithm is significantly more efficient than the algorithm in [1] without separation of k-path cuts.

6 Conclusions

In this paper, we propose VRPSolver models for four vehicle routing problems with intermediate stops: the CMTVRPTW, the MDVRPI, the CLARPIF and the GVRP. These models contain one variable for each feasible complete route. The set of feasible routes is expressed as a set of resource constrained paths in specified graphs. The intermediate stops en route are taken into account by defining a negative resource consumption for certain arcs in these graphs.

The performance of models, when approaching them with the VRPSolver, is compared with the current best exact algorithms in the literature. For the CMTVRPTW, our approach solved to optimality more instances than the algorithm proposed by [44]. Our results are especially good for the largest test instances. For the MDVRPI, our approach is faster for most instances than the algorithm proposed by [40]. In addition, many instances with 100 and 120 customers are solved to optimality for the first time. For the CLARPIF, our approach is the first exact one proposed in the literature. For the G-VRP, even if our approach does not outperform the state-of-the-art, two instances are solved to optimality for the first time.

Before our work, only small instances of considered problems could be solved by MIP models. Our results show that exact solution of medium-size instances of vehicle routing problems with intermediate stops is now possible by defining models. Thus, non-specialists in exact approaches for VRPs can use these models to solve instances of the considered problems or other VRPIS.

				Proposed BCP								Andelmin and Bartolini (2017)	
				without initital ub				with initial ub					
		Avg.		#Inst.	Avg.			#Inst.	Avg.			#Inst.	Avg.
Group	#Inst.	n	s	Solved	Gap_0(%)	#Nodes	Time(s)	Solved	Gap_0(%)	#Nodes	Time(s)	Solved	Time(s)
AB1	20	80.4	20.2	7	1.5	460.7	15793.9	9	1.0	10.68	6303.4	19	8260.3
AB2	20	81.3	22.2	8	1.1	138.0	7032.4	12	1.2	70.66	3524.0	15	7878.9

Table 5: Summary of our results for G-VRP instances

One of possible applications is to estimate quality of heuristic approaches, as we did for the the CLARPIF. As much larger instances can now be solved to optimality by modelling, a more precise estimation of heuristic efficiency can be performed.

We believe that the potential of VRPSolver models is not yet fully explored. Thus, finding models for other classes of vehicle routing and related problems may help a larger number of practitioners to exploit results of many years of research in exact BCP algorithms in a simple way.

There exist many vehicle routing problems which cannot be well described as VRPSolver models. These are for example the variants of the CMTVRPTW, considered by [31] and [44]. In this variants, there is a dependency between resources, which cannot be efficiently modelled in the current version of VRPSolver . Therefore, a natural research direction is to enrich the VRPSolver model to cover a larger number of vehicle routing and related problems. At the same time, the generic solver proposed by [33] should be generalised to support such an enriched model.

Acknowledgments

This study was financed in part by Capes PrInt UFF, Brazil no 88881, by CNPq, Brazil grant 313601/2018-6, and by FAPERJ, Brazil grant E-26/202.887/2017.

References

[1] J. Andelmin and E. Bartolini. An exact algorithm for the green vehicle routing problem. *Transportation Science*, 51(4):1288–1303, 2017.

[2] E. Angelelli and M. G. Speranza. The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137(2):233–247, 2002.

[3] M. Asghari, S. M. J. M. Al-e, et al. Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics*, 231:107899, 2021.

[4] N. Azi, M. Gendreau, and J.-Y. Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763, 2010.

[5] R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, 2011.

[6] L. H. Barbosa and E. Uchoa. A branch-cut-and-price algorithm for the traveling salesperson problem with hotel selection. *Computers & Operations Research*, 123:104986, 2020.

[7] J.-M. Belenguer, E. Benavent, P. Lacomme, and C. Prins. Lower and upper bounds for the mixed capacitated arc routing problem. *Computers & Operations Research*, 33(12): 3363–3383, 2006.

[8] M. Bruglieri, S. Mancini, F. Pezzella, and O. Pisacane. A path-based solution approach for the green vehicle routing problem. *Computers & Operations Research*, 103:109–122, 2019.

[9] D. Cattaruzza, N. Absi, and D. Feillet. The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, 50(2):676–693, 2016.

[10] D. Cattaruzza, N. Absi, and D. Feillet. Vehicle routing problems with multiple trips. *4OR*, 14(3):223–259, 2016.

[11] L. Costa, C. Contardo, and G. Desaulniers. Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985, 2019.

[12] B. Crevier, J.-F. Cordeau, and G. Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773, 2007.

[13] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management science*, 6 (1):80–91, 1959.

[14] G. Desaulniers, F. Errico, S. Irnich, and M. Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016.

[15] A. Divsalar, P. Vansteenwegen, and D. Cattrysse. A variable neighborhood search method for the orienteering problem with hotel selection. *International Journal of Production Economics*, 145(1):150–160, 2013.

[16] S. Erdoğan and E. Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114, 2012.

[17] G. Ghiani, G. Improta, and G. Laporte. The capacitated arc routing problem with intermediate facilities. *Networks: An International Journal*, 37(3):134–143, 2001.

[18] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions. *Journal of Mathematical Modelling and Algorithms*, 3(3):209–223, 2004.

[19] G. Ghiani, D. Laganà, G. Laporte, and F. Mari. Ant colony optimization for the arc routing problem with intermediate facilities under capacity and length restrictions. *Journal of Heuristics*, 16(2):211–233, 2010.

[20] A. Goel. Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, 43(1):17–26, 2009.

[21] A. Goel and S. Irnich. An exact method for vehicle routing and truck driver scheduling problems. *Transportation Science*, 51(2):737–754, 2017.

[22] B. L. Golden, J. S. DeArmon, and E. K. Baker. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, 10(1):47–59, 1983.

[23] F. Hernandez, D. Feillet, R. Giroudeau, and O. Naud. A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. *4or*, 12(3): 235–259, 2014.

[24] F. Hernandez, D. Feillet, R. Giroudeau, and O. Naud. Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, 249(2):551 – 559, 2016.

[25] G. Hiermann, J. Puchinger, S. Ropke, and R. F. Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995–1018, 2016.

[26] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008.

[27] A. Lim, Z. Zhang, and H. Qin. Pickup and delivery service with manpower planning in hong kong public hospitals. *Transportation Science*, 51(2):688–705, 2017.

[28] R. Macedo, C. Alves, J. Valério de Carvalho, F. Clautiaux, and S. Hanafi. Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model. *European Journal of Operational Research*, 214(3):536–545, 2011.

[29] I. Muter, J.-F. Cordeau, and G. Laporte. A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Science*, 48(3):425–441, 2014.

[30] B. Pan, Z. Zhang, and A. Lim. Multi-trip time-dependent vehicle routing problem with time windows. *European Journal of Operational Research*, 291(1):218–231, 2021.

[31] R. Paradiso, R. Roberti, D. Laganá, and W. Dullaert. An exact solution framework for multitrip vehicle-routing problems with time windows. *Operations Research*, 68(1):180–198, 2020.

[32] D. Pecin, A. Pessoa, M. Poggi, E. Uchoa, and H. Santos. Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters*, 45(3):206–209, 2017.

[33] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183(1):483–523, 2020.

[34] M. Polacek, K. F. Doerner, R. F. Hartl, and V. Maniezzo. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics*, 14(5):405–423, 2008.

[35] A. Rijal, M. Bijvank, A. Goel, and R. de Koster. Workforce scheduling with order-picking assignments in distribution facilities. *Transportation Science*, 55(3):725–746, 2021.

[36] M. Schiffer, M. Schneider, G. Walther, and G. Laporte. Vehicle routing and location routing with intermediate stops: A review. *Transportation Science*, 53(2):319–343, 2019.

[37] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[38] C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis. A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *INFORMS Journal on computing*, 20(1):154–168, 2008.

[39] P. Vansteenwegen, W. Souffriau, and K. Sörensen. The travelling salesperson problem with hotel selection. *Journal of the Operational Research Society*, 63(2):207–217, 2012.

[40] A. Wang, J. E. Arbogast, G. Bonnier, Z. Wilson, and C. E. Gounaris. Estimation of marginal cost to serve individual customers. *Technical Report 7573, Optimization Online*, 2020.

[41] E. J. Willemse and J. W. Joubert. Constructive heuristics for the mixed capacity arc routing problem under time restrictions with intermediate facilities. *Computers & Operations Research*, 68:30–62, 2016.

[42] E. J. Willemse and J. W. Joubert. Splitting procedures for the mixed capacitated arc routing problem under time restrictions with intermediate facilities. *Operations Research Letters*, 44(5):569–574, 2016.

[43] E. J. Willemse and J. W. Joubert. Efficient local search strategies for the mixed capacitated arc routing problems under time restrictions with intermediate facilities. *Computers & Operations Research*, 105:203–225, 2019.

[44] Y. Yang. An efficient adaptable exact solution framework for the capacitated multi-trip vehicle routing problem with time windows and its variants. *Technical Report 8237, Optimization Online*, 2021.

[45] Z. Zhang, M. Liu, and A. Lim. A memetic algorithm for the patient transportation problem. *Omega*, 54:60–71, 2015.

[46] L. Zhen, C. Ma, K. Wang, L. Xiao, and W. Zhang. Multi-depot multi-trip vehicle routing problem with time windows and release dates. *Transportation Research Part E: Logistics and Transportation Review*, 135:101866, 2020.