Exactly solving the Split Pickup and Split Delivery Vehicle Routing Problem on a bike-sharing system

Marco Casazza

To cite this version:
Marco Casazza. Exactly solving the Split Pickup and Split Delivery Vehicle Routing Problem on a bike-sharing system. [Research Report] Universita degli Studi di Milano; Universite Paris 13. 2016. <hal-01304433>

HAL Id: hal-01304433
https://hal.sorbonne-universite.fr/hal-01304433
Submitted on 19 Apr 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Exactly solving the Split Pickup and Split Delivery Vehicle Routing Problem on a bike-sharing system

Marco Casazza
April 19, 2016

1 Introduction

Over the past decade an increasing number of cities around the world have adopted bike-sharing systems. A bike-sharing system is a public service in which bicycles are made available for shared use to individuals on a short term basis. Typically, bikes are stored in rack stations. People rent a bike at a cost to travel around the city, and drop it back at either the same rack station or at a different one.

The Velib system in Paris, started in 2007, is a success story with more than 50 millions trips in its first two years of service [21]; in 2014 more than 800 cities across the globe had similar bike-sharing systems.

One of the main issues of bike-sharing systems is ensuring the availability of the bikes. In fact, during peak hours, flows along particular direction are registered, leading to high risk of empty racks in departure stations, and full racks at destination. Both represent a disservice, and may even prevent people to use the system, since the users are forced to spend time in searching for alternative stations in the neighbourhood.

One of the solutions chosen by many operators is to iteratively rebalance the system by means of a fleet of dedicated trucks: transportation demand is forecast, and bikes are picked up from stations where congestion is expected, and delivered to those expected to become empty. Due to the high costs of running trucks in a urban environment, efficient rebalancing operations are a key factor for the success of the whole system. Unfortunately, such operations require to solve very hard optimization problems.

In this paper we face a Split Pickup and Split Delivery Vehicle Routing Problem (SPSDVRP) arising on such a bike-sharing system. We assume that it is given a homogeneous fleet of vehicles of limited capacity, a network of stations, the travel cost and time between them, a forecast of transportation demand and a current status of the network, expressed in terms of desired (resp. currently available) number of bikes at each station.

The SPSDVRP requires therefore to find a route for each vehicle, that is a pattern defining which stations need to be visited, the order of visits, and the amount of bikes loaded or unloaded at each station. Due to capacity restrictions, no bikes can be loaded into a full vehicle, not unloaded from an empty one;
similarly, no bike can be unloaded to a full station, and no more bikes can be loaded from a station than those actually available. Route length cannot exceed a given limit, representing the operator shift duration. We assume that each vehicle always starts and ends at a central depot with no bikes on board. We also assume that loading and unloading times are negligible with respect to travelling times. We finally assume that no station is used as a temporary unloading location; that is, in each station bikes can be only loaded or unloaded, and therefore during the balancing operations the number of bikes in each station is monotone. From a logistics point of view, without the latter assumption complex synchronization issues would arise, that would be very difficult to be implemented in practice. As a consequence, each station is classified since the beginning as either pickup, when more bikes are parked than the desired ones, or delivery, when instead more bikes are expected to be needed than the currently available ones.

A solution to the SPSDVRP consists of a set of routes respecting the above conditions, and such that the desired target demand is achieved for each station of the network. A solution is considered to be optimal when minimizing the sum of the travelling costs of all vehicles.

From an application point of view, there is currently a lively research trend in optimizing bike-sharing systems. In [31] the authors adopt a statistical approach to discuss the performances of existing systems. In [43], data from the Vienna bikes sharing system are gathered and studied to give a model that could be used to further expand the network. In [32] it is described a model that gives a strategic planning of a bike-sharing system by considering service level requirements. In [36] the authors propose several models and algorithms to solve bike repositioning problems. Their objective is to find the best repositioning that can be achieved by several vehicles within time limits in a static case, that is they assume a negligible usage rate of the system. The satisfaction function introduced in [35] is used to evaluate the quality of a repositioning. Both service level requirements and bike repositioning are combined in [39]. In [17] the authors propose to solve a dynamic public bike-sharing balancing problem. They introduce a time-discretized model of the system and use column generation techniques to obtain in short time instructions to be given to the drivers, in order to minimize the number of uncovered users.

From a methodological point of view, our SPSDVRP is NP-Hard, generalizing several problems in transportation. For instance, when all stations have a target number of bikes that is higher than the initial one, except for a single depot station, the SPSDVRP becomes a Split Delivery Vehicle Routing Problem (SDVRP) [2]. SDVRPs have been first studied in [25]. The authors proved that the split feature of the problem may lead to substantial savings, while increasing at the same time the complexity of the problem [2]. In [4] a tabu search algorithm to solve this problem is presented, while in [5] and [6] the authors propose exact algorithms exploiting column generation and cutting planes, respectively. In [22] a further degree of complexity is considered, forcing the deliveries to be satisfied within given time windows. The author proposes a branch-and-price approach to solve the problem to optimality.

The SPSDVRP belongs to the wide class of Pickup and Delivery Vehicle Routing Problems (PDVRPs), where a fleet of vehicles is used to transport supplies from either a depot or some selected vertices of the network, to either other vertices
or back to the depot. For recent surveys on PDVRPs we refer to [8], covering freight transportation, and to [24] covering transportation of people. PDVRPs involving additional operational constraints have recently been addressed, as LIFO constraints on the loading/unloading [9], time windows [28], and both [15]. However, a substantial difference stands between a standard PDVRP and the SPSDVRP: while in the former the requests for each pair of pickup and delivery points are given, in the latter the quantity of supply transported from each pickup to each delivery point is a decision variable.

Finally, the SPSDVRP can be classified as a many-to-many (M-M) vehicle routing problem, in which a request has multiple origins (in our case pickup stations) and multiple destinations (in our case delivery stations). These kind of transportation problems arise for instance in maritime oil transportation [29].

The routing problem induced by balancing in bike-sharing systems with a single vehicle is addressed in [14], where the authors succeed in providing a strong lower bound and an effective heuristic. In [27] the authors propose a first branch-and-cut exploiting Benders' cuts. In [26] a variant of the problem is tackled, where the rebalancing requires to satisfy an interval demand; the authors exploit cutting planes methods to design exact algorithms. In [20], the authors propose new models and valid inequalities for the pure combinatorial version of the problem, that is without considering the option of partially serving demands at each station.

A first mathematical programming algorithm for the SPSDVRP on a bike-sharing system has been proposed in [13]. The author models the problem by means of a set partitioning extended formulation in which each variable represents a full vehicle route, that is including its rebalancing pattern. First, the author obtains strong lower bounds by solving the continuous relaxation of his extended formulation by means of column generation techniques; to solve the pricing problem the author adapts an ad-hoc algorithm for the VRP first described in [7]. Secondly, he obtains tight upper bounds by means of a memetic algorithm. Third, he generates all columns with a reduced cost smaller than the gap between lower and upper bounds, following the technique of [7], and solves to integer optimality the resulting problem by means of general purpose integer programming solver.

The method of [13] has two main drawbacks: first, it is not designed to handle travelling times, that are instead approximated by a limit on the number of visits in each route; second, it is designed to tackle only instances with a very limited number of stations, due to the nature of its third step, and of the pricing problem to be solved during column generation.

We propose a new exact method for the SPSDVRP that overcomes these two drawbacks.

In Section 2 we formalize the problem; in Section 3 we propose a new mathematical programming model that makes use of combinations of suitable combinatorial structures to reduce the complexity of the problem. We then discuss about a few theoretical properties of such a model, and we propose an extended formulation obtained through Dantzig-Wolfe decomposition. In Section 4 we explain the details of our algorithm, while in Section 5 we show our computational results. Brief conclusions follow in Section 6.
2 Problem formalization and notation

The SPSDVRP for a bike-sharing system can be formalized as follows: a set of station nodes \( N = \{1 \ldots n\} \) is given, each with an initial number \( \text{stock}_i \) and a target number \( \text{target}_i \) of bikes. When \( \text{stock}_i > \text{target}_i \), \( i \) is defined as a pickup node, when \( \text{stock}_i < \text{target}_i \) as a delivery node, and when \( \text{stock}_i = \text{target}_i \) as a balanced node. Let us define \( N^+ = \{ i \in N \mid \text{stock}_i > \text{target}_i \} \) and \( N^- = \{ i \in N \mid \text{stock}_i < \text{target}_i \} \) as the set of pickup nodes and the set of delivery nodes, respectively. The demand of each node \( d_i = |\text{stock}_i - \text{target}_i| \) is the quantity of bikes to pickup from (resp. deliver to) that node.

Let \( G = (N_0, A) \) be a directed graph in which \( N_0 = N \cup \{0\} \) is the set of nodes including the depot 0, and \( A = \{(i, j) \mid i, j \in N_0\} \) is the set of arcs connecting them. Let \( c_{ij} \) be the travelling cost of arc \((i, j) \in A\). W.l.o.g we assume that travelling costs satisfy the triangular inequality, that is \( c_{ij} \leq c_{ik} + c_{kj} \) for all \( i, j, k \in N_0 \). Let \( t_{ij} \) be the travelling time of arc \((i, j) \in A\); we assume that triangular inequalities hold also for travelling times, that is \( t_{ij} \leq t_{ik} + t_{kj} \) for all \( i, j, k \in N_0 \).

An example of such an input of the SPSDVRP is shown in Figure 1; for clarity only a few arcs are depicted. In such figure, each node has two labels: one label that identifies the station \( i \), and one attached label that is its demand \( d_i \). Also, each node is denoted by a + or by a − if it is a pickup or a delivery node, respectively.

![Figure 1: Example of instance with 10 station. Symbols + and − denote a pickup or a delivery station, respectively. At each station is attached a label reporting its demand.](image)

A homogeneous fleet of vehicles \( M = \{1 \ldots m\} \) each with capacity \( C \) is given to satisfy station node demands. Whenever a vehicle visits a station node, it may pick up or deliver a certain amount of bikes, depending on its current load. Since we assume that all vehicles begin and end their route empty, the
sum of pickup demands must be equal to the sum of delivery demands, that is \( \sum_{i \in N} (stock_i - target_i) = 0 \) Moreover, each vehicle has a resource \( T \) that represent the available travelling time of such vehicle.

The SPSDVRP on a bike-sharing system is the problem of redistributing bikes in the network at minimum travelling cost, satisfying node demands while not exceeding neither vehicles capacity nor their time resource. In Figure 2 we depict an example in which we assume 2 vehicles with capacity \( C = 10 \) each. Both vehicles start empty from the depot and visit pickup node 5 splitting its demand. The load of vehicles after each operation is reported as a label on the arcs of the solutions. Also the demand of the delivery node 10 is split; then the two vehicles end their routes empty.

As discussed in the introduction, the convergence to the target state is required to be monotonous and drops are not allowed. It means that bikes can only be loaded at pickup vertices and unloaded at delivery vertices. As a consequence initially balanced vertices are not visited by any vehicle and so from now on we assume them to be removed from \( G \). Instead, pickup and delivery vertices can be visited several times, either by the same vehicle or by different ones.

### 3 Groups formulation and properties

The approach to the SPSDVRP proposed in [13], that modelled the problem as a set covering extended formulation in which each variable is a specific route pattern, revealed that solving the continuous relaxation of such formulation was
very challenging due to the structure of the pricing problem. Indeed, we also performed preliminary solution attempts and experiments in that modelling direction; these however, confirmed the findings of [13]. That motivated us to elaborate on a different approach, identifying particular regularities and properties of combinatorial substructures of the routes, and trying to reduce the complexity of the pricing problem by exploiting these properties. Indeed, this kind of approach is in nature similar to those proposed in [33] and [12], that proved to be successful in similar contexts.

We first present some observations that led to our intuition (subsection 3.1), then we describe in detail our approach (subsections 3.1, 3.2, 3.3 and 3.4).

3.1 Routes and groups

We first observe that, due to triangular inequality:

**Observation 3.1.** There always exists an optimal solution in which no node is visited without collecting at least one unit of its demand.

Furthermore, since vehicles always start and end the route empty at the depot, we can observe that:

**Observation 3.2.** There always exists an optimal solution in which no vehicle visits a delivery or a pickup station at the beginning or at the end of its route, respectively.

These observations can be simply generalized as follows:

**Observation 3.3.** A route always starts with a sequence composed only by pickup nodes, always ends with a sequence composed only by delivery nodes, and in general always interleaves sequences in which a set of pickup nodes are visited, without deliveries in between, followed by a set of visits to delivery nodes, without pickups in between.

Our intuition is therefore that the structure of a route can be much simplified by explicitly encoding such an interleaved behaviour. Indeed, in the following we formalize such an intuition, and prove a few key properties of such an encoding.

**Definition 3.1.** We denote as group a sequence of one or more pickup nodes followed by a sequence of one or more delivery nodes.

Therefore, a route is itself a sequence of one or more groups linked together, plus an additional stop at the depot at the begin and at the end. An example of group structure in a route is depicted in Figure 3: the first route is partitioned in two groups, one with four and one with two nodes. The second route is partitioned in three groups with two nodes each. All groups start with a pickup node and end with a delivery node.

**Definition 3.2.** A group is feasible if both the sum of loaded bikes and the sum of unloaded bikes do not exceed the capacity of the vehicle.

**Definition 3.3.** The cost of a group is given by the sum of the arcs connecting the nodes of the inner pickup and delivery sequences.

Let $g$ and $g'$ be two consecutive groups of a route, and let $i$ and $j$ be the last and first node of group $g$ and $g'$, respectively. Then groups $g$ and $g'$ are connected in the route by an arc $(i, j)$.
Figure 3: Group partitioning of routes in Figure 2

**Definition 3.4.** The cost of a route is the sum of the cost of its groups and the cost of its connecting arcs.

We readily observe that:

**Observation 3.4.** There always exists an optimal solution in which no node is visited more than once in the same group.

**Proof.** In fact, let us suppose by contradiction that such an optimal solution exists and that \( i \) is a pickup node visited twice in a group. Let \( q'_i \) and \( q''_i \) be the two quantities loaded on the vehicle, and that \( \tilde{d}_i \) is the demand of \( i \) loaded, that is \( \tilde{d}_i = q'_i + q''_i \). Since the visits occur in the same group, we know that between the first and the second visit there are only pickup nodes, and that loading \( \tilde{d}_i \) units of demand does not exceed vehicle capacity. Therefore, we can set \( q'_i = \tilde{d}_i \) and \( q''_i = 0 \), avoiding the second visit due to Observation 3.1. The same holds if \( i \) is a delivery node, visited more than once.

We remark that such a property does not hold outside the group structure; that is, in general, visiting the same node twice may be both needed for feasibility or simply be profitable.

We also observe that:

**Observation 3.5.** The number of bikes loaded in each node of a pickup sequence is irrelevant to both the feasibility and the cost of a group, as long as the total amount of loaded bikes remains constant. The same applies to the number of bikes loaded in each node of a delivery sequence.

That is, each group can encode implicitly a potentially huge number of equivalent solutions.

### 3.2 Routes, groups and loading patterns.

We now consider the particular SPSDVRP subproblem arising when the nodes visiting sequence is assumed to be given, and only a suitable loading/unloading plan needs to be found. Our main result is that by exploiting the groups structure we are able to improve theoretical findings from the literature.

**Theorem 3.1.** Given the sets of nodes visited in each group of each vehicle, the problem of assigning the maximum quantity loaded (resp. unloaded) at each station can be solved in polynomial time.
Proof. Let us build a graph in which we have a source node \( s \) and a sink node \( t \), a node \( f^+_i \) for each pickup node \( i \) and a node \( f^-_j \) for each delivery node \( j \), and two nodes \( p_m^g \) and \( d_m^g \) for each group \( g \) of each vehicle \( m \). Let us add arcs from \( s \) to \( f^+_i \) and from \( f^-_j \) to \( t \) with capacity \( d_i \) and \( d_j \), respectively. For each pickup node \( i \) visited in a group \( g \) of vehicle \( m \), add an arc from \( f^+_i \) to \( p_m^g \) with infinite capacity. Similarly, for each delivery node \( j \) visited in a group \( g \) of vehicle \( m \), add an arc from \( d_m^g \) to \( f^-_j \) still with infinite capacity. From each node \( p_m^g \) add an arc to \( d_m^g \) with capacity \( C \), and from each node \( d_m^g \) add an arc to \( p_m^{g+1} \) with infinite capacity. An example of the graph is reported in Figure 4.

A maximum flow solution on such a graph ensures that at most \( \sum_{i \in N^+} d_i \) units can be loaded, and at most \( \sum_{i \in N^-} d_i \) units unloaded. For each station node, the units loaded (unloaded) are at most the demand of the station itself because of the limited capacity of ingoing arcs in \( f^+_i \) (outgoing arcs from \( f^-_j \)). Nodes \( p_m^g \) and \( d_m^g \) represent the load of the vehicle after pickups and deliveries, respectively. No vehicle is overloaded due to the limited capacity of arcs \( \langle p_m^g, d_m^g \rangle \).

The quantities loaded and unloaded at nodes \( i \) and \( j \) of a group \( g \) of vehicle \( m \) are given by the flow on the arc \( (f^+_i, p_m^g) \) and \( (d_m^g, f^-_j) \), respectively. \( \square \)

**Corollary 3.2.** If the flow reaching the depot \( t \) is less than the number of demands of pickup (delivery) nodes, then the starting assignment of nodes to groups does not represent a feasible solution.

This follows from the fact that some demands are not satisfied if the flow is...
less then their sum.

Theorem 3.1 and Corollary 3.2 imply:

**Observation 3.6.** Given a set of routes without loading (unloading) information, it is always possible to complete the solution with such quantities in polynomial time, or prove that such solution is infeasible.

Our approach is indeed similar to the one presented in [13], where given a set of routes without loading quantities, the authors use a flow formulation to obtain such a missing information. Indeed the claim of our theorem 3.1 matches the findings of [13].

However, our proof is different: from a theoretical point of view, our approach allows to build smaller support graph, and may therefore yield a better computational behaviour. In details, the graph of [13] has two nodes for each station of the problem, while our graph has one node for each station, and two nodes for each group. Therefore, our graph has always a smaller number of nodes, except in the extreme scenario in which exactly one pickup and one delivery node is visited in each group. In such scenario, the number of nodes of the two graphs is identical. Furthermore, our approach requires less information about the order of the nodes in the route, and can be used to early detect the infeasibility of a node in a branch-and-bound approach as described in Subsection 4.6.

### 3.3 A formulation based on groups

Then we exploit the features of groups to obtain a new formulation of the SPSDVRP. To ease notation we assume that there exists a particular additional group containing the depot only. Let $G = \{1 \ldots g\}$ be the set of available groups for each vehicle, the SPSDVRP can be formulated as follows:

$$
\begin{align*}
\min & \sum_{m \in M} \sum_{g \in G} c_{ij} \cdot x_{ij}^m + \sum_{i,j \in N} c_{ij} \cdot z_{ij}^m \\
\text{s.t.} & \sum_{m \in M} \sum_{g \in G} w_{mg}^i \geq 1 \quad \forall i \in N \\
& \sum_{m \in M} q_{i}^m = d_i \quad \forall i \in N \\
& q_{i}^m \leq d_i \cdot w_{i}^m \quad \forall m \in M \ \forall g \in G \\
& q_{i}^m \geq w_{i}^m \quad \forall m \in M \ \forall g \in G \\
& \sum_{i \in N^+} q_{i}^m \leq C \quad \forall m \in M \ \forall g \in G \\
& \sum_{i \in N^-} q_{i}^m \leq C \quad \forall m \in M \ \forall g \in G \\
& f_{mg} + \sum_{i \in N^+} q_{i}^{m+1} - \sum_{i \in N^-} q_{i}^{m+1} = f_{mg+1} \quad \forall m \in M \ \forall g \in G
\end{align*}
$$
\[ f_{mg} + \sum_{i \in N^+} q_i^{mg} \leq C \cdot (1 - w_{mg}^0) \quad \forall m \in M \forall g \in G \quad (3.9) \]

\[ \sum_{i \in N} w_i^{mg} \leq |N| \cdot (1 - w_{mg}^0) \quad \forall m \in M \forall g \in G \quad (3.10) \]

\[ \sum_{j \in N} z_{ij}^{mg} = \sum_{j \in N^+} z_{ij}^{mg} + s_i^{mg} = w_i^{mg} \quad \forall m \in M \forall g \in G \quad (3.11) \]

\[ \sum_{j \in N^+} z_{ij}^{mg} + e_i^{mg} = \sum_{j \in N} z_{ij}^{mg} = w_i^{mg} \quad \forall m \in M \forall g \in G \quad (3.12) \]

\[ 1 \geq \sum_{i \in N^+} z_{ij}^{mg} \geq w_i^{mg} \quad \forall m \in M \forall g \in G \quad (3.13) \]

\[ \sum_{i \in N \setminus S} z_{ij}^{mg} + z_{ji}^{mg} \geq w_i^{mg} \quad \forall m \in M \forall g \in G \forall S \subset N \quad |S| > 0 \forall u \in S \quad (3.14) \]

\[ \sum_{i \in N_0^+} e_i^{mg} \leq 1 \quad \forall m \in M \forall g \in G \quad (3.15) \]

\[ \sum_{i \in N_0^+} s_i^{mg} \leq 1 \quad \forall m \in M \forall g \in G \quad (3.16) \]

\[ w_{mg}^0 \geq e_0^{mg} + s_0^{mg} \quad \forall m \in M \forall g \in G \quad (3.17) \]

\[ e_i^{mg} \leq \sum_{j \in N_0^+} x_{ij}^{mg} \quad \forall m \in M \forall g \in G \forall i \in N_0^+ \quad (3.18) \]

\[ s_i^{mg+1} \leq \sum_{j \in N_0^+} x_{ij}^{mg} \quad \forall m \in M \forall g \in G \forall i \in N_0^+ \quad (3.19) \]

\[ e_0^{mg+1} = \sum_{g \in G} e_0^{mg+1} = 1 \quad \forall m \in M \quad (3.20) \]

\[ \sum_{i, j \in N} x_{ij}^{mg} \leq \sum_{i, j \in N_0} x_{ij}^{mg-1} + \sum_{i, j \in N_0} x_{ij}^{mg-1} \quad \forall m \in M \forall g \in G \quad (3.21) \]

\[ \sum_{i \in N_0^+} x_{ij}^{mg} \leq 1 \quad \forall m \in M \forall g \in G \quad (3.22) \]

\[ \sum_{i \in N_0^+} t_{ij} \cdot x_{ij}^{mg} + \sum_{i, j \in N} t_{ij} \cdot z_{ij}^{mg} \leq T \quad \forall m \in M \forall g \in G \quad (3.23) \]

\[ x_{ij}^{mg} \in \mathbb{B} \quad \forall m \in M \forall g \in G \forall i \in N \forall j \in N \quad (3.24) \]

\[ z_{ij}^{mg} \in \mathbb{B} \quad \forall m \in M \forall g \in G \forall i \in N \forall j \in N \quad (3.25) \]

\[ w_i^{mg} \in \mathbb{B} \quad \forall m \in M \forall g \in G \forall i \in N \quad (3.26) \]

\[ 0 \leq f_{mg} \leq C \quad \forall m \in M \forall g \in G \quad (3.27) \]
Variables $x_{ij}^{mg}$ and $z_{ij}^{mg}$ correspond respectively to the linking arcs between groups and the linking arcs inside a group; the former are set to 1 if there is an arc between group $g$ and group $g+1$ that connect node $i$ and $j$, while the latter are set to 1 if there is an arc between node $i$ and node $j$ inside the group. Variables $w_{ij}^{mg}$ are set to 1 if node $i$ is visited by vehicle $m$ in group $g$. Variables $s_{i}^{mg}$ and $e_{i}^{mg}$ are set to 1 if $i$ is the starting node of the group $g$ or if $i$ is the ending node respectively. Variable $q_{i}^{mg}$ is the quantity loaded (unloaded) at node $i$ by vehicle $m$ in group $g$, while variable $f_{i}^{mg}$ is the load of the vehicle $m$ after visiting group $g$.

The objective function (3.1) minimizes the overall cost by minimizing both group costs and linking arc costs. Constraints (3.2) and (3.3) ensure respectively that each station is visited at least once and its demand is satisfied. Constraints (3.4) avoid loading (unloading) when a node is not visited, while constraints (3.5) impose that if a node is visited, then at least one unit of demand is loaded. Constraints (3.6) and (3.7) ensure that vehicle capacity is not exceeded for each group. Constraints (3.8) ensure consistency of the flow in the route, while constraints (3.9) impose that the vehicle is empty when visiting depot and therefore a vehicle starts and ends empty. Constraints (3.10) impose that no station is visited in a group if the depot is visited, too. Constraints (3.11) and (3.12) ensure respectively that all pickup and delivery nodes visited have ingoing and outgoing arcs. Constraints (3.13) ensure that in each group with a pickup or delivery node, there is an arc going from pickup to delivery, while constraints (3.14) ensure that there are no subtours in a group. Constraints (3.15) and (3.16) impose that for each group there must be at most one ending node and one starting node, while constraints (3.17) impose that if a vehicle visits the depot, it is only to start or to end a route. Constraints (3.18) and (3.19) impose the use of linking arcs between each group, while constraints (3.20) has the double effect of ensure that each vehicle visits the depot twice, and that its route starts from the depot. Constraints (3.21) ensure that no arcs are used after the depot is visited and constraints (3.22) impose that at most one linking arc is used for each group. Finally, constraints (3.23) impose a limit on the time resource consumed by each vehicle.

An upper bound on the number of groups. In model (3.1) – (3.30), the total number of groups is not known in advance. However, since a maximal resource $T$ is given, we observe that:

**Observation 3.7.** An upper bound $n_{max}$ on the maximal number of nodes visited in a route can be obtained by solving a 0-1 Knapsack Problem (KP).

In fact, we can model the problem of finding the maximal number of nodes
visited in a single route as follows:

\[
\begin{align*}
\max & \quad \sum_{i \in N_0} w_i \\
\text{s. t.} & \quad \sum_{i \in N} w_i \cdot \left( \min_{j \in N_0} t_{ji} \right) \leq T \\
& \quad w_0 = 1 \\
& \quad w_i \in \mathbb{B} \quad \forall i \in N 
\end{align*}
\] (3.31)

(3.32)

(3.33)

(3.34)

where \( w_i \) is set to 1 if node \( i \) is visited.

The objective function (3.31) maximizes the number of nodes visited. Constraint (3.32) ensures that resource \( T \) is not exceeded. Constraint (3.33) imposes that the depot is always visited.

Problem (3.31) – (3.34) can be solved as a KP in which (a) the vector of prizes is \((1 \ldots 1)\), (b) each item weight is the minimum ingoing (outgoing) arc, and (c) node 0 is always included, therefore decreasing the resource \( T \) by \( \min_{j \in N_0} t_{j0} \).

Such a problem can be solved in polynomial time by packing items with smallest \( \min_{j \in N_0} t_{ji} \) first.

Furthermore, the maximum number of visited nodes directly influences the maximum number of groups of each vehicle:

**Observation 3.8.** For each route of a vehicle \( m \), there are at most

\[
g^m_{\text{max}} = \left\lfloor \frac{n_{\text{max}} - 1}{2} \right\rfloor + 2
\]

groups.

Given a fixed number of nodes, maximizing the number of groups is equivalent to minimize the number of nodes in each group. Therefore, we obtain the maximal number of groups when there are at most two nodes per group, one pickup node and one delivery node. Since the depot is always visited twice, once at the beginning and once at the end of the route, we need two additional groups.

### 3.4 Extended formulation

In order to obtain tight dual bounds to be used in search tree algorithms, we built an extended formulation of the model (3.1) – (3.30) exploiting Dantzig-Wolfe decomposition [19]. Let, for each vehicle \( m \in M \) and group \( q \in G \),

\[
\Omega_{mq} = \left\{ (z, w, q, s, e) \in \mathbb{B}^{[N] \times [N]} \times \mathbb{B}^{[N_0]} \times \mathbb{N}_0^{[N]} \times \mathbb{B}^{[N_0^+] \times [N_0^-]} \right\}
\]
be a set of feasible integer points, where each vector \((z, w, q, s, e)\) satisfies the constraints

\[
\begin{align*}
q_i &\leq d_i \cdot w_i & \forall i \in N \\
q_i &\geq w_i & \forall i \in N \\
\sum_{i \in N^+} q_i &\leq C \\
\sum_{i \in N^-} q_i &\leq C \\
\sum_{i \in N} w_i &\leq |N| \cdot (1 - w_0) \\
\sum_{j \in N} z_{ij} &\leq \sum_{j \in N^+} z_{ji} + s_i = w_i & \forall i \in N^+ \\
\sum_{j \in N^-} z_{ij} + e_i &\leq \sum_{j \in N} z_{ji} = w_j & \forall i \in N^- \\
1 &\geq \sum_{i \in N^+} z_{ij} + \sum_{j \in N^-} z_{ij} & u \in N \\
\sum_{i \in N^+_S, j \in S} (z_{ij} + z_{ji}) &\geq w_u & S \subseteq N, |S| > 0, u \in S \\
\sum_{i \in N^+_0} e_i &\leq 1 \\
\sum_{i \in N^-_0} s_i &\leq 1 \\
w_0 &\geq e_0 + s_0.
\end{align*}
\]

We relax integrality conditions, but replace each \(\Omega^{mg}\) with the convex hull of its \(L^{mg}\) extreme integer points

\[
\Gamma^{mg} = \{(\bar{z}^1, \bar{w}^1, \bar{q}^1, \bar{s}^1, \bar{e}^1), \ldots, (\bar{z}^{L^{mg}}, \bar{w}^{L^{mg}}, \bar{q}^{L^{mg}}, \bar{s}^{L^{mg}}, \bar{e}^{L^{mg}})\}
\]

and we impose

\[
(z, w, q, s, e) = \sum_{k \in \Gamma^{mg}} (z^k, w^k, q^k, s^k, e^k) \cdot y^k
\]  

with \(y^k \geq 0\) for each \(k \in \Gamma^{mg}\), \(m \in M\), and \(g \in G\), and \(\sum_{k \in \Gamma^{mg}} y^k = 1\) for each \(m \in M\) and \(g \in G\). That is, each point is represented as a linear convex combination of points in \(\Gamma^{mg}\).

The model obtained by replacing in the continuous relaxations of formulation (3.1) – (3.30) the vectors \((z, w, q, s, e)\) as indicated in (3.35), and by making explicit the vector indices is

\[
\min \sum_{m \in M} \sum_{i \in N^+_0, j \in N^+_0} c_{ij} \cdot x_{ij}^{mg} + \sum_{i, j \in N} \sum_{k \in \Gamma^{mg}} c_{ij} \cdot \bar{z}_{ij}^k \cdot y^k
\]  

with \(\bar{z}_{ij}^k \geq 0\) for each \(k \in \Gamma^{mg}\), \(i, j \in N\), \(m \in M\), and \(g \in G\), and \(\sum_{k \in \Gamma^{mg}} \bar{z}_{ij}^k = 1\) for each \(i, j \in N\) and \(m \in M\) and \(g \in G\).
s.t. \[ \sum_{m \in M, g \in G, k \in \Gamma_{mg}} w_k \cdot y_k \geq 1 \quad \forall i \in N \] (3.37)

\[ \sum_{m \in M, g \in G, k \in \Gamma_{mg}} q_k \cdot y_k = d_i \quad \forall i \in N \] (3.38)

\[ f_{mg} + \sum_{i \in N^+, k \in \Gamma_{mg+1}} q_k \cdot y_k - \sum_{i \in N^-, k \in \Gamma_{mg+1}} q_k \cdot y_k = f_{mg+1} \quad \forall m \in M, g \in G \] (3.39)

\[ f_{mg} + \sum_{i \in N^+, k \in \Gamma_{mg+1}} q_k \cdot y_k = d_i \quad \forall i \in N \] (3.40)

\[ \sum_{k \in \Gamma_{mg}} e_k \cdot y_k \leq \sum_{j \in N^+_{mg}} x_{ij} \quad \forall m \in M, g \in G, i \in N^0 \] (3.41)

\[ \sum_{k \in \Gamma_{mg+1}} s_k \cdot y_k \leq \sum_{j \in N^-_{mg}} x_{ji} \quad \forall m \in M, g \in G, i \in N^0 \] (3.42)

\[ \sum_{k \in \Gamma_{mg}} q_{mg} \cdot y_k = \sum_{g \in G, g > 1} k \in \Gamma_{mg} \cdot s_{mg} \cdot y_k = 1 \quad \forall m \in M \] (3.43)

\[ \sum_{i \in N^-, j \in N^+} t_{ij} \cdot x_{ij} + \sum_{i, j \in N, k \in \Gamma_{mg}} t_{ij} \cdot s_{ij} \cdot y_k \leq T \quad \forall m \in M, g \in G \] (3.44)

\[ \sum_{i, j \in N} x_{ij} = \sum_{i, j \in N, k \in \Gamma_{mg}} x_{ij} - \sum_{i, j \in N, k \in \Gamma_{mg}} x_{ij} = 1 \quad \forall m \in M, g \in G \] (3.45)

\[ \sum_{i \in N^-} x_{ij} \leq 1 \quad \forall m \in M, g \in G \] (3.46)

\[ \sum_{k \in \Gamma_{mg}} y_k = 1 \quad \forall m \in M, g \in G \] (3.47)

\[ y_k \geq 0 \quad \forall m \in M, g \in G, k \in \Gamma_{mg} \] (3.48)

\[ 0 \leq x_{ij} \leq 1 \quad \forall m \in M, g \in G, i \in N^- \] (3.49)

\[ 0 \leq f_{mg} \leq C \quad \forall m \in M, g \in G \] (3.50)

Constraints (3.47) may be first relaxed in ≤ form, because a pattern \((0 \ldots 0)\) always exists for each set \(\Gamma_{mg}\), and then can be removed because constraints (3.46) already impose at most one linking arc and therefore at most one selected group.

**Observation 3.9.** The lower bound provided by the (3.36) – (3.50) is at least as tight as that given by the continuous relaxation of model (3.1) – (3.30).

To strengthen the formulation and reduce symmetries we add for each
vehicle \( m \in M \) and consecutive groups \( g \) and \( g + 1 \) the inequality

\[
\sum_{k \in \Gamma_{mg+1}} y^k = \sum_{k \in \Gamma_{mg}} y^k - \sum_{k \in \Gamma_{mg}} w_0^k \cdot y^k, \tag{3.51}
\]

and for each consecutive vehicles \( m \) and \( m + 1 \) and group \( g \in G \), the inequality

\[
\sum_{i \in N^+_0, j \in N^-_0} x_{ij}^{mg} \geq \sum_{i \in N^+_0, j \in N^-_0} x_{ij}^{m+1g} \tag{3.52}
\]

The model can then be rewritten as

\[
\min \sum_{m \in M} \sum_{g \in G} \sum_{i \in N_0^+} c_{ij} \cdot x_{ij}^{mg} + \sum_{j \in N_0^-} c_{ij} \cdot \bar{z}_{ij} \cdot y^k \tag{3.53}
\]

s.t. \( \sum_{m \in M} \sum_{g \in G} \sum_{i \in N_0^+} \bar{w}_i^k \cdot y^k \geq 1 \quad \forall i \in N \tag{3.54} \)

\[
\sum_{m \in M} \sum_{g \in G} \sum_{k \in \Gamma_{mg}} q_i^k \cdot y^k = d_i \quad \forall i \in N \tag{3.55}
\]

\[
f^{mg} + \sum_{i \in N^+_0} q_i^k \cdot y^k - \sum_{i \in N^-} q_i^k \cdot y^k \leq f^{mg+1} \quad \forall m \in M \quad \forall g \in G \tag{3.56}
\]

\[
f^{mg} + \sum_{i \in N^+_0} q_i^k \cdot y^k + C \cdot \sum_{k \in \Gamma_{mg}} \bar{w}_i^k \cdot y^k \leq C \quad \forall m \in M \quad \forall g \in G \tag{3.57}
\]

\[
\sum_{k \in \Gamma_{mg}} e_i^k \cdot y^k = \sum_{j \in N^+_0} x_{ij}^{mg} \leq 0 \quad \forall m \in M \quad \forall g \in G \tag{3.58}
\]

\[
\sum_{k \in \Gamma_{mg}} \bar{z}_{ij} \cdot y^k - \sum_{j \in N^-_0} x_{ij}^{mg} \leq 0 \quad \forall m \in M \quad \forall g \in G \tag{3.59}
\]

\[
\sum_{m \in M} \sum_{g \in G} \sum_{k \in \Gamma_{mg}} x_{0j}^{mg} \cdot y^k - \sum_{g \in G} \sum_{k \in \Gamma_{mg}} s_{0g}^{mg} \cdot y^k = 0 \quad \forall m \in M \tag{3.60}
\]

\[
\sum_{k \in \Gamma_{mg}} \bar{w}_0^k \cdot y^k = 1 \quad \forall m \in M \tag{3.61}
\]

\[
\sum_{i \in N^-_0} l_i \cdot x_{ij}^{mg} + \sum_{i \in N^+_0} l_i \cdot \bar{z}_{ij} \cdot y^k \leq T \quad \forall m \in M \quad \forall g \in G \tag{3.62}
\]

\[
\sum_{i,j \in N_0} x_{ij}^{mg} - \sum_{i,j \in N_0} x_{ij}^{mg-1} + \sum_{i \in \Gamma_{mg}} x_{ij}^{mg-1} \leq 0 \quad \forall m \in M \quad \forall g \in G \tag{3.63}
\]

\[
\sum_{i,j \in N^+_0} x_{ij}^{mg} \leq 1 \quad \forall m \in M \quad \forall g \in G \tag{3.64}
\]

\[
\sum_{k \in \Gamma_{mg+1}} y^k - \sum_{k \in \Gamma_{mg}} y^k + \sum_{k \in \Gamma_{mg}} w_0^k \cdot y^k = 0 \quad \forall m \in M \quad \forall g \in G \tag{3.65}
\]
$$\sum_{i \in \mathcal{N}_-^m} x_{ij}^{mg} - \sum_{i \in \mathcal{N}_-^m} x_{ij}^{mg+1} \geq 0 \quad \forall m \in \mathcal{M} \quad \forall g \in \mathcal{G}$$ (3.66)

$$y^k \geq 0 \quad \forall m \in \mathcal{M} \quad \forall g \in \mathcal{G}$$ (3.67)

$$0 \leq x_{ij}^{mg} \leq 1 \quad \forall i \in \mathcal{N}_-^m \quad \forall j \in \mathcal{N}_+^m$$ (3.68)

$$0 \leq f^{mg} \leq C \quad \forall m \in \mathcal{M} \quad \forall g \in \mathcal{G}$$ (3.69)

4 Algorithms

The size of the set $\mathcal{I}$ grows exponentially in the number of nodes and the sum of demands; therefore, we solve the MP by means of column generation techniques: we solve to optimality a Restricted Master Problem (RMP) involving a small set of columns (see Subsection 4.1), and we iteratively search for negative reduced cost variables solving a pricing problem (see Subsection 4.2). If no negative reduced cost variable is found, the optimal RMP solution is optimal for the MP as well, and therefore the corresponding value is retained as a valid lower bound for the SPSDVRP. If the final RMP solution is integer, then it is also optimal for the SPSDVRP, otherwise we enter a search tree by performing branching operations (see Subsection 4.3) to find a proven global optimum.

4.1 Initialization

We initialize the RMP by generating the set of columns corresponding to groups including the depot only. We remark that if a group contains the depot, then it does not contain any station. Our aim is to both simplify the pricing problem by removing a decision variable and to populate the RMP with a set of columns that are always needed to obtain a feasible solution. Therefore, for each vehicle we add an initial depot in group 1, and an ending depot in all groups $g > 1$.

In model (3.53) – (3.69), constraints (3.63) impose that a vehicle cannot visit a station after the ending depot. Indeed, if a vehicle $m$ visits the depot in group 2, that is

$$\sum_{k \in \Gamma_{m_2}} w^k_{0} \cdot y^k = 1,$$

then $m$ visits the depot in both groups 1 and 2, and therefore it does not visit any station.

For a few vehicles we can forbid such a redundant behaviour. Let us consider for example vehicle 1: due to symmetry constraints (3.66), if vehicle 1 visits the depot in group 2, then no other vehicle can leave the depot nor visit any station.

Observation 4.1. Let $g_{\text{min}}$ be the minimum number of groups required to perform all the pickups and deliveries, that is

$$g_{\text{min}} = \left\lceil \frac{\sum_{i \in \mathcal{N}} d_i}{2} \right\rceil,$$
we avoid columns with depot only in group 2 of vehicle \( m \) if
\[
(m - 1) \cdot (g_{\text{max}}^m - 2) < g_{\text{min}}.
\]

In other words, a vehicle may be left unused only if the previous ones are able to perform all the pickups and deliveries.

4.2 Pricing problem

Due to our initialization method, the pricer can neglect groups including the depot.

Let \( \pi, \lambda, \mu, \zeta, \nu^+, \nu^-, \eta, \) and \( \theta \) be respectively the dual variables of constraints (3.54), (3.55), (3.56), (3.57), (3.58), (3.59), (3.62), and (3.65). Since \( \nu^+ \) and \( \nu^- \) are referred to the two sets of nodes \( N^+ \) and \( N^- \), for each node \( i \in N \) we use instead
\[
\nu_i = \begin{cases} 
\nu^+_i & \text{if } i \in N^+ \\
\nu^-_i & \text{if } i \in N^- 
\end{cases}.
\]

For each \( k \in \Gamma_{mg} \), the reduced cost of variable \( y^k \) is computed as
\[
s^k = \sum_{i,j \in N} c_{ij} \cdot \bar{z}_{ij} - \sum_{i,j \in N} \eta_{mg} \cdot t_{ij} \cdot \bar{z}_{ij} = \sum_{i \in N} \pi_i \cdot \bar{w}_{ik} - \sum_{i \in N} \lambda_i \cdot \bar{q}_{ik} + \sum_{i \in N^+} \mu_{mg} \cdot \bar{q}_{ik}^+ - \sum_{i \in N^-} \zeta_{mg} \cdot \bar{q}_{ik}^-
\]
\[
- \sum_{i \in N^+} \nu^+_i \cdot \bar{s}_{ki} + \sum_{i \in N^-} \nu^-_i \cdot \bar{e}_{ki} - \theta_{mg} + \theta_{mg+1}.
\]

The component \( -\theta_{mg} + \theta_{mg+1} \) is a fixed prize (cost) gained (paid) for any additional column and thus can be ignored during pricing problem optimization.

After collecting the coefficients, the pricing objective function reads as follows:
\[
s^k = \sum_{i,j \in N} (c_{ij} - \eta_{mg} \cdot t_{ij}) \cdot \bar{z}_{ij} - \sum_{i \in N} \pi_i \cdot \bar{w}_{ik} - \sum_{i \in N^+} (\lambda_i + \mu_{mg} + \zeta_{mg}) \cdot \bar{q}_{ik}^+ - \sum_{i \in N^-} (\lambda_i - \mu_{mg}) \cdot \bar{q}_{ik}^-
\]
\[
- \sum_{i \in N^+} \nu^+_i \cdot \bar{s}_{ki} + \sum_{i \in N^-} \nu^-_i \cdot \bar{e}_{ki}.
\]

Let
\[
\alpha_{ij} = c_{ij} - \eta_{mg} \cdot t_{ij} \text{ be the cost of travelling on arc } (i, j);
\]
\[
\beta_{i} = \lambda_i + \mu_{mg} + \zeta_{mg} \text{ be the prize or cost of loading one unit in node } i \in N^+ \text{ and } \beta_{i} = \lambda_i - \mu_{mg} \text{ be the prize or cost of loading one unit of node } i \in N^-;
\]
\[
\gamma_{i} \text{ be the cost of starting the group visiting node } i \in N^+ \text{ and } \gamma_{i} \text{ be the cost of ending the group visiting node } i \in N^-.
\]
The objective function can be now stated as
\[
\sigma^k = \sum_{i,j \in N} \alpha_{ij}^m \cdot z_{ij}^k - \sum_{i \in N} \pi_i \cdot \bar{w}_i^k - \sum_{i \in N} \beta_i^m \cdot \bar{q}_i^k
- \sum_{i \in N^+} \gamma_i^m \cdot s_i^k - \sum_{i \in N^-} \gamma_i^m \cdot \bar{e}_i^k.
\]

For each vehicle \(m \in M\) and group \(g \in G\), we can formulate the pricing problem as follows:
\[
\begin{align*}
\min & \quad \sum_{i,j \in N} \alpha_{ij}^m \cdot z_{ij}^k - \sum_{i \in N} \pi_i \cdot \bar{w}_i^k - \sum_{i \in N} \beta_i^m \cdot \bar{q}_i^k \\
\quad \text{s.t.} & \quad q_i^k \leq d_i \cdot \bar{w}_i^k \quad \forall i \in N \quad (4.1) \\
& \quad q_i^k \geq w_i^k \quad \forall i \in N \quad (4.2) \\
& \quad \sum_{i \in N^+} q_i^k \leq C \quad \forall i \in N \quad (4.3) \\
& \quad \sum_{i \in N^-} q_i^k \leq C \quad \forall i \in N \quad (4.4) \\
& \quad \sum_{j \in N} z_{ij}^k = \sum_{j \in N^+} z_{ij}^k + s_i^k = \bar{w}_i^k \quad \forall i \in N^+ \\
& \quad \sum_{j \in N^-} z_{ij}^k + \bar{e}_i^k = \sum_{j \in N} z_{ji}^k = \bar{w}_i^k \quad \forall i \in N^- \quad (4.6) \\
& \quad 1 \geq \sum_{i \in N^+} \sum_{j \in N^-} z_{ij}^k \geq \bar{w}_u^k \quad \forall u \in N \quad (4.7) \\
& \quad \sum_{i \in S} \sum_{j \in S^c} (z_{ij}^k + z_{ji}^k) \geq \bar{w}_u^k \quad \forall S \subseteq N, |S| > 0 \quad u \in S \quad (4.8) \\
& \quad \sum_{i \in N^-} \bar{e}_i^k \leq 1 \quad \forall i \in N^- \quad (4.9) \\
& \quad \sum_{i \in N^+} s_i^k \leq 1 \quad \forall i \in N^+ \quad (4.10) \\
& \quad z_{ij}^k \in B \quad i,j \in N \quad (4.11) \\
& \quad \bar{w}_i^k \in B \quad \forall i \in N \quad (4.12) \\
& \quad s_i^k \in B \quad \forall i \in N^+ \quad (4.13) \\
& \quad \bar{e}_i^k \in B \quad \forall i \in N^- \quad (4.14)
\end{align*}
\]

Except for the depots, in a feasible solution of the pricing problem we may have three kinds of visited nodes: integer nodes, that are those nodes with their demands fully collected, bridge nodes, that are those with only one unit of demand collected, and fractional nodes, that are those whose demands are fractionally collected.
It may happen that nodes are visited just to collect their \( \pi \) prize, and therefore we may have more than one node in a feasible solution whose demand is not fully collected. However, we can observe that if the sequence of nodes were given, our pricing problem would reduce to a 0-1 Knapsack Problem (KP). Therefore we can exploit properties on the KP to prove that:

**Theorem 4.1.** There always exists an optimal solution of (4.1) – (4.15) in which there is at most one fractional pickup node (resp. delivery node).

**Proof.** Assume by contradiction that there exists an optimal solution in which, among the visited nodes, all demands are fully loaded on the vehicle but those of nodes \( i \) and \( j \), which have both fractional loadings \( \bar{q}_k^i \) and \( \bar{q}_k^j \). Let \( r = \bar{q}_k^i + \bar{q}_k^j \) be the space in the vehicle occupied by node \( i \) and node \( j \) loadings. Let us assume w.l.o.g. that node \( i \) is more efficient than node \( j \), that is \( \beta^{mg}_i > \beta^{mg}_j \).

We can improve the contribution to the objective value (4.1) by decreasing the space occupied by the less efficient node, and increasing the space of the most efficient. The argument iteratively extends to solutions in which the number of nodes is higher than two.

**Observation 4.2.** There always exists an optimal MP solution, in which each selected column contains at most one fractional node with more than one unit of demand collected.

In fact, each MP column can be found by solving the pricing problem, and thus by adding columns with at most one node with fragmented quantity.

Furthermore:

**Observation 4.3.** In any optimal solution of the pricing problem, the prize \( \beta_j \) of a fractional node \( j \) is less than or equal to the prize \( \beta_i \) of any integer node \( i \), and it is greater than or equal to the prize \( \beta_u \) of any bridge node \( u \), that is

\[
\beta_i \geq \beta_j \geq \beta_u.
\]

The proof directly follows the proof of Theorem 4.1.

**Pricing algorithm.** The pricing problem is a variant of the Resource Constrained Elementary Shortest Path Problem (RCESPP), that it is known to be NP-Hard. However, a recent survey [34] reviews very effective methods that solve the RCESPP by means of labelling algorithms. In [5], the authors use a discretization approach to solve the pricing problem for the SDVRP, solving an RCESPP on an extended graph with \( d_i \) nodes for each station \( i \). The drawback of such approach is that the size of the graph grows with both the number of nodes and the magnitude of demand coefficients. Another approach to solve the pricing problem consists of a nested column generation [42], in which also the pricing problem is solved by branch-and-price. In [30] the authors exploit such approach to solve the SPSDVRP on a maritime crude oil transportation problem in which the first level pricing problem is considered too complex to be solved efficiently by a dynamic programming algorithm. Instead, in [22] the author solves the pricing problem of a SDVRP with Time Windows by means of a labelling algorithm. In such a problem, the routes generated have at most one split demand, and the author devises an algorithm that generates three different labels at each visit, depending on the residual capacity of the vehicle: one label
Figure 5: Example of auxiliary graph used to solve the pricing problem: pickup nodes are unreachable from the time the vehicle visits the first delivery node.

when the node is visited but no unit of demand is collected, one with a fractional loading, and one in which the demand is fully collected. In fact, except for the handling of time windows, the pricing problem of [22] is very similar to ours.

Therefore, inspired by [22], and adapting techniques from [37], we propose a dynamic programming algorithm devised for our specific case.

Let us consider a single vehicle with a limited capacity $C$ and a particular graph $\hat{G} = (\hat{N}, \hat{A})$, where $\hat{N} = N \cup \{0^+, 0^–\}$. All nodes $i \in N$ have a demand $d_i$ but nodes $0^+$ and $0^–$. When a node $i$ is visited, its prize $\pi_i$ is collected, while a prize $\beta_i^{mg}$ is collected for each unit of demand of node $i$ loaded on the vehicle.

We denote as $\hat{A}$ the set of arcs of the graph, that is composed by

- arcs from $0^+$ to all nodes $i \in \hat{N}^+$;
- arcs from nodes $i \in \hat{N}^+$ to all nodes $j \in \hat{N}$;
- arcs from nodes $i \in \hat{N}^–$ to all nodes $j \in \hat{N}^– \cup \{0^–\}$.

An example of the graph is depicted in Figure 5.

For each arc $(i, j)$ we have a cost

$$c_{ij} = \begin{cases} 
\gamma^{mg}_{ij}, & \text{if } i = 0^+ \\
\gamma^{ing}_{ij}, & \text{if } j = 0^– \\
\alpha^{mg}_{ij}, & \text{otherwise}
\end{cases}$$

Furthermore, capacity is always replenished when travelling from a pickup node to a delivery node.

The objective of the problem is to find a minimum cost path that goes from $0^+$ to $0^–$.

The pricing problem is a variant of RCESPP in which we have a resource $r$ indicating the residual space in the partial path, and a set $V$ indicating which stations have already been visited. We stress that information on the amount of bikes loaded or unloaded on the truck at each node are not needed, as we never pass through the same node twice. Let us remark also that, because of the particular structure of the graph, each path has two distinguishing features: first, it has at least one pickup and one delivery node, and second, no pickup node is visited after a delivery node.

Therefore our label is a tuple $(i, c, r, V, f)$, where $i$ is the ending node of a partial path, $c$ is the cost of such partial path, and $f$ indicates that a fractional node has been visited.
Initialization. We initialize our algorithm with a label $\lambda$ representing a partial path starting from $0^+$ with no initial cost, full resources, and no fractional node, that is

$$\lambda = (0^+, 0, C, \{0^+\}, 0).$$

Label extension. As a selection strategy, at each iteration we select the most profitable label $\lambda' = (i, c', r', V', f')$ with minimum partial cost. We extend such label to all neighbours $j$ of $i$ such that $j \notin V'$, in three different ways:

**integer:** $j$ is selected as an integer node, fully collecting its demand. The resulting label $\lambda'' = (j, c'', r'', V'', f')$ has a partial cost $c'' = c' + c_j - \beta_j^{mg}$.\[d_j - \pi_j, r'' = r' - d_j, \text{ and } V'' = V' \cup \{j\};\]

**bridge:** $j$ is selected as a bridge node, collecting only one unit of its demand. The resulting label $\lambda'' = (j, c'', r'', V'', f')$ has a partial cost $c'' = c' + c_j - \beta_j^{mg} - \pi_j, r'' = r' - 1, \text{ and } V'' = V' \cup \{j\};$

**fractional:** only if no fractional node is selected yet, $j$ is selected as a fractional node. It is not possible to determine a priori the collected demand of $j$, and therefore the resulting label $\lambda'' = (j, c'', r'', V'', f')$ has a partial cost $c'' = c' + c_j - \pi_j, V'' = V' \cup \{j\}$. The quantity loaded in $j$ is determined only at the end of a sequence of pickup (resp. delivery) nodes.

We also remark that if $i$ and $j$ are a pickup and a delivery node, respectively, the residual capacity $r$ is fully replenished before visiting $j$.

Dominance rules. The drawback of our extension method is that we cannot collect the prize of a fractional node until all pickup (delivery) nodes are visited. Therefore, when checking the dominance of a given label with a fractional node, we have to ensure that such label is dominated for every value of fractional loading.

Let suppose that we are given two labels $\lambda' = (i', c', r', V', f')$ and $\lambda'' = (i'', c'', r'', V'', f'')$, with $i' = i''$ and $r' \leq r''$. Indeed, we may use some of the residual space of $\lambda''$ to load some units of $f''$, matching the residual space of $\lambda'$ and therefore reducing the cost of $\lambda''$. Let us consider the two extreme cases in which we load 1 and $\max_{f'} = \min\{r', d_{f'} - 1\}$ units of $f'$ demands. In the first case, we have to load $\min_{f''} = \min\{r'' - r' + 1, d_{f''} - 1\}$ units of $f''$ demand to match the residual space in $\lambda'$, while in the second case $\max_{f''} = \min\{r'' - r' + \max_{f'} + d_{f''} - 1\}$. Then, we say that label $\lambda'$ is dominated if it exists a label $\lambda''$ such that

$$\begin{cases}i' = i'' \\ r' \leq r'' \\ V'' \subseteq V' \\ c' + \beta_{f'} \geq c'' + \beta_{f''} \cdot \min_{f''} \\ c' + \beta_{f'} \cdot \max_{f'} \geq c'' + \beta_{f''} \cdot \max_{f''}
\end{cases}$$

and at least one of the inequalities is strict. Dominance rules ensure that a label is dominated only if it exists another label of a partial path ending in the same node, that has more residual capacity, has visited less nodes, and costs less for each quantity of $f'$ loaded.
Reduction and lookahead dominance. During the process we check several condition to reduce the number of extensions. In fact, due to Observation 4.3 we know that an optimal solution always exists, in which no fractional node has higher prize than an integer node, nor smaller prize than a bridge node. Therefore, let us suppose that \( \lambda \) is the label to extend, \( \beta_j \) is the prize of the destination node \( j \), and \( \beta_{\text{integer}}, \beta_{\text{fractional}}, \) and \( \beta_{\text{bridge}} \) are respectively the minimum prize among all integer nodes in \( \lambda \), the prize of the fractional node in \( \lambda \), and the maximum prize among all bridge nodes in \( \lambda \).

- integer extension is forbidden when \( \beta_j < \beta_{\text{fractional}} \) or \( \beta_j < \beta_{\text{bridge}} \), and in general when \( \beta_j \leq 0 \);
- bridge extension is forbidden when \( \beta_j > \beta_{\text{fractional}} \) or \( \beta_j > \beta_{\text{integer}} \);
- fractional extension is forbidden when \( \beta_j > \beta_{\text{integer}} \) or \( \beta_j < \beta_{\text{bridge}} \), and in general when \( \beta_j \leq 0 \).

Furthermore, let suppose that we are given two labels \( \lambda' \) and \( \lambda'' \), both identical but for two nodes \( i \) and \( j \). Let us suppose that \( i \) is a bridge node in \( \lambda' \) and a fractional node \( \lambda'' \), while vice versa, \( j \) is a fractional node in \( \lambda' \) and a bridge node in \( \lambda'' \). Let us suppose w.l.o.g. that \( \beta_i \geq \beta_j \), then \( \lambda'' \) dominates \( \lambda' \). Therefore, we aggregate fractional and bridge extension when we visit a node \( j \) with a positive prize \( \beta_j \): given a label with fractional node \( f \), if \( \beta_j \geq \beta_f \) then a unit of \( f \) is collected, cost and residual capacity of the label are updated, \( j \) becomes the new fractional node of the label, and \( f \) becomes a bridge node. Otherwise we perform a bridge extension.

Pricer execution and insertion policy. In a strict column generation approach, this procedure should be computed for each pair of group \( g \in G \) and vehicle \( m \in M \), in order to find the column with minimum reduced cost. Instead, we found profitable from a computational point of view, to perform partial pricing, and stop the pricing procedure after we find the first pair \( g \) and \( m \) yielding a negative reduced cost column. We then insert such a column in all \( \Gamma_{mg} \) sets.

Bidirectional algorithm. In order to further improve the performance of our pricing algorithm, we exploit bidirectional search. Since each solution is composed of a sequence of pickup nodes, and a sequence of delivery nodes that can be treated independently, we run the algorithm on two different graphs, one composed by pickup nodes only, and one by delivery nodes only. At the end, we obtain full solutions by joining the best labels of each pair of pickup and delivery nodes.

Heuristic pricing. To speed up the column generation process, we also implemented two heuristic variants of the pricing algorithm.

First we run the exact algorithm on a reduced graph, obtained by removing for each node, the set of \( k \) arcs with highest travelling cost \( \tilde{c}_{ij} \), with \( k \) fixed a-priori.

Second, if the first heuristic does not find any column with negative reduced cost, we run the pricing algorithm considering columns with integer loading only. In such a way, at each extension we generate only one label in the destination node, reducing the overall number of labels.
If none of the two heuristic algorithms generate a column with negative reduced cost, we run the exact pricing algorithm.

4.3 Branching rules

When the optimal MP solution is fractional, and upper and lower bounds do not match, we check which integrality constraints in the original formulation are not satisfied and enforce them by exploring a search tree through branching rules. In our case, branching is particularly involved, as the MP is prone to symmetries.

We devised the following binary branching rule in which nodes are progressively fixed in vehicle groups. Let $\tilde{y}^k$ be the value of a variable $y^k$ in the fractional solution of the MP, and let

$$\tilde{w}_i^{mg} = \sum_{k \in \Gamma_{mg}} \tilde{w}_i^k \cdot \tilde{y}_i^k$$

be the fractional assignment of each node $i \in N_0$ to a group $g \in G$ of a vehicle $m \in M$. We search for a tuple $(\hat{i}, \hat{g}, \hat{m})$ that corresponds to the maximum fractional assignment in the current fractional solution, that is

$$(\hat{i}, \hat{g}, \hat{m}) \in \arg\min_{\hat{i} \in N_0, \hat{g} \in G, \hat{m} \in M} \|\tilde{w}_i^{mg} - 1/2\|.$$ 

If $\tilde{w}_i^{\hat{g}\hat{m}}$ is fractional, then we perform binary branching: in one branch we enforce $\hat{i}$ to be always visited by vehicle $\hat{m}$ in group $\hat{g}$. In the other branch, we preclude the visit of $\hat{i}$ by vehicle $\hat{m}$ in group $\hat{g}$. Let us recall that forcing $\hat{i}$ to be visited by vehicle $\hat{m}$ in group $\hat{g}$ does not preclude the visit of $\hat{i}$ by another vehicle or in different groups.

If no fractional $\tilde{w}_i^{\hat{g}\hat{m}}$ is found, we can stop branching, as an integer SDSPVRP solution can be directly found. In fact, the following holds.

**Observation 4.4.** When no assignment to groups is fractional, there is at least one route for each vehicle that is integer.

**Proof.** Let us suppose that we are given an optimal solution of the MP in a certain branching node. If no fractional $\tilde{w}_i^{\hat{g}\hat{m}}$ is found, it means that for each group of each vehicle, the (potentially fractional) selected columns describe groups that are permutations of the same set of stations. Therefore, among all the permutations, only the best permutation can be selected, finding an equivalent, but integer, optimal MP solution.

This means that for each vehicle we can arbitrary select one route among all the fractionally selected ones for that vehicle, in order to obtain a full SDSPVRP solution.

A solution obtained in such a way could still be non-integer due to loading quantities in each node, but exploiting Observation 3.6 we can then optimally assign such values in polynomial time, obtaining a full feasible integer solution.
4.4 Branching implementation.
First, let us remark that depot 0, is not involved in the pricing problem, and therefore if $\hat{i} = 0$ we branch by adding in the MP a constraint

$$\sum_{k \in \Gamma g_m} \hat{w}_k \leq 0$$  \hspace{1cm} (4.16)

in one branch, excluding the depot from a group, and

$$\sum_{k \in \Gamma g_m} \hat{w}_k \geq 1$$  \hspace{1cm} (4.17)

in the other branch, fixing it into the group.

Instead, if $\hat{i}$ is not 0, but a pickup or delivery node, we first exclude in one branch all columns that contains node $\hat{i}$ in group $\hat{g}$ of vehicle $\hat{m}$, and we modify the pricing graph by removing incoming arcs in $\hat{i}$. This ensures that no further column is generated including such a node, and also improves the performances of the pricing algorithm.

In the other branch, we fix $\hat{i}$ into group $\hat{g}$ of vehicle $\hat{m}$ and, to ensure that the MP selects columns with such a feature, we add the constraint

$$\sum_{k \in \Gamma g_m} \hat{w}_k \geq 1.$$  \hspace{1cm} (4.18)

This constraint, in turn, introduces a new dual variable that must be considered in the pricing problem. Therefore, when we visit node $\hat{i}$ during dynamic programming extension, we collect a prize corresponding to that dual variable. Let us remark that such a detail does not change the structure of the pricer.

4.5 Additional inequalities
According to literature [13]:

Theorem 4.2. For any optimal solution, given two pickup (or equiv. two delivery) nodes $i$ and $j$, the number of times arc $(i,j)$ is used plus the number of times arc $(j,i)$ is used is less than or equal to 1.

We extend this theorem and prove the following:

Theorem 4.3. For any optimal solution, given two pickup (or equiv. two delivery) nodes $i$ and $j$, the number of times $i$ and $j$ are in the same group is less than or equal to 1.

Proof. Let us consider w.l.o.g. an optimal solution in which two vehicles visit nodes $i$ and $j$ in the same group. Let $a_i$ and $a_j$ be the number of bikes loaded (unloaded) by the first vehicle on nodes $i$ and $j$, and let $b_i$ and $b_j$ be the number of bikes loaded (unloaded) by the second vehicle. It may happen that:

- $a_i \geq b_j$: in this case we can set new loading values $a_i' = a_i - b_j$, $a_j' = a_j + b_j$, $b_j' = 0$ and $b_i' = b_i + b_j$. The quantity previously loaded from $b_j$ is then loaded from $a_j$, while the overall load of each vehicle does not change since $a_i$ is decreased by $b_j$ and $b_i$ is increased by $b_j$;
• $a_i < b_j$: in this case $a_i' = 0$, $a_j' = a_j + a_i$, $b_i' = b_i - a_i$ and $b_j' = b_i + a_i$.

The quantity previously loaded from $a_i$ is then loaded from $b_i'$, while the overall load of each vehicle does not change since $b_i$ is decreased by $a_i$ and $a_j$ is increased by $a_i$.

In both cases one vehicle visits a node without any operation and therefore the corresponding arc is removed.

We remark that the theorem holds also when nodes are visited by the same vehicle.

**Corollary 4.4.** Let $v^{m g}_{i j}$ be the binary variable that is 1 if pickup (delivery) node $i$ is visited together with pickup (delivery) node $j$ in group $g$ of vehicle $m$; then constraints

\[
\sum_{m \in M} \sum_{g \in G} v^{m g}_{i j} \leq 1 \tag{4.19}
\]

are valid inequalities for model (3.1) – (3.30).

In the MP, Constraints (4.19) become

\[
\sum_{m \in M} \sum_{g \in G} \sum_{k \in \Gamma_{m g}} v^{k}_{i j} \cdot y^k \leq 1 \tag{4.20}
\]

and their corresponding dual variables must be taken into account into the pricing problem. In fact, let $\xi_{i j}$ be the dual variables of Constraints (4.20), we first add the terms

\[
\sum_{i,j \in N^+} \xi_{i j} \cdot \bar{v}^{k}_{i j} + \sum_{i,j \in N^-} \xi_{i j} \cdot \bar{v}^{k}_{i j}
\]

to the objective function (4.1). Then, we also add constraint

\[
w_i + w_j \leq v_{i j} + 1
\]

to the pricing problem for each pair of pickup (delivery) nodes such that $i < j$.

For what concerns the implementation of the pricing algorithm, we modify the extension in such a way that when we extend to a node $j$, we add to the new label the sum of the costs associated to the node $j$, that is

\[
\sum_{i \in N^+} \xi_{i j} \quad \text{if } j \text{ is a pickup node, and}
\]

\[
\sum_{i \in N^-} \xi_{i j} \quad \text{if } j \text{ is a delivery node.}
\]

Let us remark that such inequalities do not change the structure of the pricing algorithm. In fact, a label can be dominated only by a label that visited a subset of its nodes. Therefore, all the prices collected by the latter, are also collected by the former.
4.6 Infeasibility detection

During the exploration of the branching tree, we may run into nodes corresponding to infeasible partial solutions. Indeed, solving the MP relaxation of such a node would reveal the infeasibility, but this may require several iterations of column generation. Therefore we perform a first feasibility check on the node, in order to detect its infeasibility before computing its continuous relaxation. If the test succeeds, then the current node may lead to a feasible solution, and therefore the relaxation is computed. Otherwise, the node is simply discarded.

In details, let us suppose to have a partial solution obtained through branching, and let suppose that for each group \( g \in G \) and vehicle \( m \in M \), we have a set \( F_{mg} \) of nodes that are fixed into such a group, and a set \( E_{mg} \) of nodes that are excluded. All nodes that do not belong to any of the two sets are free nodes that may or may not be visited in such a group.

We then build a graph similar to the one in Subsection 3.2, in which we have a source node \( s \), a depot node \( t \), a node \( f^+_{i} \) for each pickup node \( i \), a node \( f^-_{i} \) for each delivery node \( i \), and two nodes \( p_{mg} \) and \( d_{mg} \) for each available group of each vehicle.

Now, let us define \( \tilde{d}_i \) as the maximum quantity of demand of node \( i \) that is not fixed into any group, that is if a node is fixed in one group \( \tilde{d}_i = d_i - 1 \), and in general

\[
\tilde{d}_i = d_i - \sum_{g \in G \atop m \in M} |F_{mg} \cap \{i\}|
\]

We add arcs from \( s \) to \( f^+_{i} \) and from \( f^-_{i} \) to \( t \) with capacity \( \tilde{d}_i \). These arcs limit the quantity of demand we are free to distribute for nodes that have been fixed in groups.

Then, for each pickup node \( i \), group \( g \) and vehicle \( m \) add an arc from \( f^+_{i} \) to \( p_{mg} \) if \( i \notin E_{mg} \), and respectively for each delivery node \( i \), group \( g \) and vehicle \( m \) add an arc from \( d_{mg} \) to \( f^-_{i} \) if \( i \notin E_{mg} \). Also, for each pickup node \( i \in F_{mg} \), add an arc with capacity 1 from \( s \) to node \( p_{mg} \), and respectively for each delivery node \( i \in F_{mg} \), add an arc with capacity 1 from \( d_{mg} \) to node \( t \). These arcs impose that to cover the demand of a station, some flow must pass through the groups in which such station has been fixed.

Finally, from each node \( p_{mg} \) add an arc to \( d_{mg} \) with capacity \( C \), and from each node \( d_{mg} \) add an arc to \( p_{mg+1} \), with infinite capacity.

An example of the graph is show in Figure 6, representing the partial solution in which we have at most 2 groups for each vehicle, and

- node 1 is fixed in group 1 of vehicle 1;
- node 5 is fixed in group 2 of vehicle 2;
- node 2 is fixed in group 2 of vehicle 1;
- node 10 is fixed in group 2 of vehicle 2.

Furthermore, node 1 and 4 are excluded respectively from groups 1 and 2 of vehicle 2, while 7 and 9 can be visited only by vehicle 2. Node 2 can be visited only by vehicle 1, while node 3 and 10 care excluded respectively from group 2 of vehicle 2 and group 2 of vehicle 1.
Figure 6: Example of graph for feasibility check (infinite capacities are omitted).
Observation 4.5. If the maximum flow going from \( s \) to \( t \) is less than the overall pickup (delivery) demand, the partial solution of the branching node is infeasible.

In fact, if the maximum flow is less than the overall demand, it means that the demand of at least one node has not been satisfied.

The contrary does not necessarily hold: in fact, if a partial solution is infeasible, it may still pass the feasibility check, because it is not possible to ensure that for each vehicle it exists a route that does not exceed resource \( T \).

5 Experimental analysis

We implemented our algorithms in C++, using the SCIP framework \([1]\) version 3.0.2. The LP subproblems were solved using the simplex algorithm implemented in CPLEX 12.4 \([23]\): the framework automatically switches between primal and dual methods. To obtain good upper bounds we also included a generic rounding heuristic from SCIP.

Unfortunately a real instance would consist of hundreds of stations, and would be out of reach for our methodology. Therefore, as a benchmark we considered the set of instances used in \([13]\) for the SPSDVRP with 10 nodes. Each instance describes a randomly generated network with nodes located in the two-dimensional space \([-500, 500] \times [-500, 500]\), with the depot located at \((0, 0)\). Travelling costs \( c_{ij} \) are computed as the Euclidean distance between nodes \( i \) and \( j \). Each station has a demand randomly generated between \([-10, 10]\), where positive values define pickup nodes, and negative values define delivery nodes, and with the sum of the pickup demands is equal to the sum of delivery demands. The vehicle capacity \( C \) is set to 10, and the number of available vehicles is 5.

The time limit resource was undefined in the original instances. To obtain a fair comparison with previous approaches from the literature we set \( T = 10 \), and each \( t_{ij} = 1 \). This means that each vehicle can visit at most 9 stations before going back to the depot.

We compared our results with the exact algorithm described in \([13]\), and sketched in the Introduction, that is able to find optimal solutions for all instances with 10 nodes. We compiled the original C++ source code with additional optimization flags, linking it to CPLEX version 12.4 libraries instead of version 12.0. A few additional coding tweaks were needed to ensure correct runs. Both algorithms have then been executed on a machine with Intel(R) Core i7-2640M at 2.80 GHz and 8 GB of memory in single thread mode.

In the remainder we refer to our exact branch-and-price algorithm as BPA, while we refer to the algorithm in \([13]\) as MH.

5.1 Column generation profiling

In a first round of experiments we performed a profiling of our algorithm in order to detail how the time is spent during the computation of a lower bound.

In Table 1 we report the results obtained at root node. For each instance we report the total time spent solving the LP during column generation, and for each type of pricer we report the number of calls, the number of generated variables, and the total running time. Results show that most of the time is
spent in solving the LPs. Heuristic pricers are effective, reducing the number of columns that must be generated by the exact pricer.

In Table 2 we report the results obtained while solving the problem to proven optimality. Still, solving the LP is usually the most time consuming operation. In the overall process, the heuristic pricing it is not effective, while the heuristic integer pricing still manages to reduce the number of calls to the exact pricer.

5.2 Root lower bound

In the second round of experiments we compared both the quality of the lower bound and the efforts required to obtain it.

In Table 3 we report for each instance the gap and the time needed to MH to compute a lower bound, the lower bound given by BPA at root node, and the lower bound and time needed to BPA to obtain the same bound of MH. Technically speaking, after setting such a target, it often happens that BPA even improves the bound of MH. The results show that BPA computation times at the root node are orders of magnitude smaller than the MH ones. However, the average lower bound given by the BPA at the root node is worse than that of MH. However, during the exploration of the branching tree, our algorithm provides on average a better lower bound than MH four times faster.

5.3 Upper bound

Third, we compared the quality and the time needed to compute a good upper bound for the problem. Let us remark that our algorithm is not intended to be used as a heuristic, and in fact we did not implement a specific heuristic for such a problem, but we rather used an off-the-shelf generic rounding heuristic available in the SCIP framework. Such a heuristic iteratively rounds fractional variables trying to recover from infeasibility whenever a constraint is violated.

For what concern MH, the algorithm runs its meta-heuristic before and after computing a lower bound to the problem. In our tests we considered the solution given by the algorithm after finishing the first meta-heuristic round. Instead, for what concern BPA we perform two different analyses: first we stopped the algorithm when a first solution is found. Second, we ran the algorithm until the gap between the upper bound and the known optimal solution is zero.

In Table 4 we report for each instance class and for each method, the gap between the corresponding upper bound and the optimal solution, and the time required for the computation. For what concern the BPA we report the number of nodes analysed before reaching its best UB, and the number of nodes and computation time needed to find the optimal solution. The results show that a first solution is found quickly, is usually good, and sometimes even optimal. Furthermore, the optimal solution is usually found in the early phases of the branching tree exploration, when only few nodes have been analysed. Indeed, BPA is able to retrieve an optimal solution within the same amount of time required by MH when used as a heuristic.
<table>
<thead>
<tr>
<th>Instance</th>
<th>RMP</th>
<th>Heuristic pricer</th>
<th>Integer pricer</th>
<th>Exact pricer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d</td>
<td>calls</td>
<td>vars</td>
<td>t (s)</td>
</tr>
<tr>
<td>n10q10a</td>
<td>52</td>
<td>23</td>
<td>1028</td>
<td>0.03</td>
</tr>
<tr>
<td>n10q10A</td>
<td>48</td>
<td>15</td>
<td>778</td>
<td>0.02</td>
</tr>
<tr>
<td>n10q10B</td>
<td>60</td>
<td>18</td>
<td>871</td>
<td>0.03</td>
</tr>
<tr>
<td>n10q10C</td>
<td>32</td>
<td>18</td>
<td>903</td>
<td>0.01</td>
</tr>
<tr>
<td>n10q10D</td>
<td>64</td>
<td>20</td>
<td>701</td>
<td>0.05</td>
</tr>
<tr>
<td>n10q10E</td>
<td>44</td>
<td>8</td>
<td>778</td>
<td>0.02</td>
</tr>
<tr>
<td>n10q10F</td>
<td>54</td>
<td>22</td>
<td>725</td>
<td>0.01</td>
</tr>
<tr>
<td>n10q10G</td>
<td>42</td>
<td>12</td>
<td>1228</td>
<td>0.02</td>
</tr>
<tr>
<td>n10q10H</td>
<td>58</td>
<td>14</td>
<td>928</td>
<td>0.01</td>
</tr>
<tr>
<td>n10q10I</td>
<td>38</td>
<td>17</td>
<td>728</td>
<td>0.02</td>
</tr>
<tr>
<td>n10q10J</td>
<td>50</td>
<td>12</td>
<td>1323</td>
<td>0.03</td>
</tr>
<tr>
<td>n10q10K</td>
<td>36</td>
<td>16</td>
<td>1153</td>
<td>0.01</td>
</tr>
<tr>
<td>n10q10L</td>
<td>52</td>
<td>24</td>
<td>897</td>
<td>0.06</td>
</tr>
<tr>
<td>n10q10M</td>
<td>34</td>
<td>12</td>
<td>1203</td>
<td>0.02</td>
</tr>
<tr>
<td>n10q10N</td>
<td>48</td>
<td>13</td>
<td>853</td>
<td>0</td>
</tr>
<tr>
<td>n10q10O</td>
<td>62</td>
<td>17</td>
<td>1303</td>
<td>0.03</td>
</tr>
<tr>
<td>n10q10P</td>
<td>44</td>
<td>18</td>
<td>1253</td>
<td>0.03</td>
</tr>
<tr>
<td>n10q10Q</td>
<td>60</td>
<td>13</td>
<td>776</td>
<td>0.02</td>
</tr>
<tr>
<td>n10q10R</td>
<td>56</td>
<td>29</td>
<td>900</td>
<td>0.04</td>
</tr>
<tr>
<td>n10q10S</td>
<td>42</td>
<td>17</td>
<td>1128</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 1: Time spent during the computation of a lower bound at root node
<table>
<thead>
<tr>
<th>Instance</th>
<th>MP graph</th>
<th>MP calls</th>
<th>MP vars</th>
<th>MP t (s)</th>
<th>Heuristic pricer calls</th>
<th>Heuristic pricer vars</th>
<th>Heuristic pricer t (s)</th>
<th>Integer pricer calls</th>
<th>Integer pricer vars</th>
<th>Integer pricer t (s)</th>
<th>Exact pricer calls</th>
<th>Exact pricer vars</th>
<th>Exact pricer t (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n10q10a</td>
<td>n10q10A</td>
<td>52</td>
<td>454</td>
<td>1628</td>
<td>0.22</td>
<td>436</td>
<td>12775</td>
<td>1.61</td>
<td>296</td>
<td>5550</td>
<td>4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10b</td>
<td>n10q10B</td>
<td>60</td>
<td>286</td>
<td>1246</td>
<td>0.14</td>
<td>269</td>
<td>9975</td>
<td>1.2</td>
<td>140</td>
<td>1375</td>
<td>1.93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10c</td>
<td>n10q10C</td>
<td>32</td>
<td>284</td>
<td>1503</td>
<td>0.13</td>
<td>258</td>
<td>3775</td>
<td>0.48</td>
<td>206</td>
<td>9775</td>
<td>2.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10d</td>
<td>n10q10D</td>
<td>64</td>
<td>49</td>
<td>826</td>
<td>0.03</td>
<td>43</td>
<td>1950</td>
<td>0.11</td>
<td>27</td>
<td>1600</td>
<td>0.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10e</td>
<td>n10q10E</td>
<td>44</td>
<td>1226</td>
<td>2278</td>
<td>0.71</td>
<td>1184</td>
<td>1200</td>
<td>3.19</td>
<td>994</td>
<td>20350</td>
<td>13.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10f</td>
<td>n10q10F</td>
<td>54</td>
<td>1988</td>
<td>1850</td>
<td>0.92</td>
<td>1960</td>
<td>50125</td>
<td>23.46</td>
<td>1337</td>
<td>41275</td>
<td>38.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10g</td>
<td>n10q10G</td>
<td>42</td>
<td>2469</td>
<td>2003</td>
<td>1.5</td>
<td>2441</td>
<td>14850</td>
<td>11.76</td>
<td>2149</td>
<td>10800</td>
<td>78.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10h</td>
<td>n10q10H</td>
<td>58</td>
<td>1669</td>
<td>1853</td>
<td>0.87</td>
<td>1669</td>
<td>10500</td>
<td>3.62</td>
<td>1473</td>
<td>25625</td>
<td>17.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10i</td>
<td>n10q10I</td>
<td>56</td>
<td>1063</td>
<td>1853</td>
<td>0.52</td>
<td>1030</td>
<td>8850</td>
<td>2.13</td>
<td>876</td>
<td>21200</td>
<td>10.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10j</td>
<td>n10q10J</td>
<td>50</td>
<td>1108</td>
<td>1973</td>
<td>0.64</td>
<td>1083</td>
<td>29625</td>
<td>14.22</td>
<td>683</td>
<td>8500</td>
<td>30.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10k</td>
<td>n10q10K</td>
<td>36</td>
<td>134</td>
<td>1428</td>
<td>0.06</td>
<td>127</td>
<td>3650</td>
<td>0.2</td>
<td>87</td>
<td>2600</td>
<td>2.38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10l</td>
<td>n10q10L</td>
<td>52</td>
<td>304</td>
<td>1447</td>
<td>0.21</td>
<td>280</td>
<td>8450</td>
<td>1.12</td>
<td>151</td>
<td>1500</td>
<td>2.72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10m</td>
<td>n10q10M</td>
<td>34</td>
<td>135</td>
<td>1428</td>
<td>0.07</td>
<td>124</td>
<td>2150</td>
<td>0.3</td>
<td>91</td>
<td>3700</td>
<td>3.49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10n</td>
<td>n10q10N</td>
<td>48</td>
<td>410</td>
<td>1733</td>
<td>0.26</td>
<td>377</td>
<td>4825</td>
<td>0.78</td>
<td>306</td>
<td>4075</td>
<td>4.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10o</td>
<td>n10q10O</td>
<td>62</td>
<td>607</td>
<td>1778</td>
<td>0.33</td>
<td>591</td>
<td>9075</td>
<td>2.1</td>
<td>441</td>
<td>6725</td>
<td>12.91</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10p</td>
<td>n10q10P</td>
<td>44</td>
<td>678</td>
<td>1928</td>
<td>0.42</td>
<td>657</td>
<td>7675</td>
<td>2.1</td>
<td>533</td>
<td>11200</td>
<td>23.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10q</td>
<td>n10q10Q</td>
<td>60</td>
<td>506</td>
<td>1626</td>
<td>0.3</td>
<td>474</td>
<td>10250</td>
<td>1.51</td>
<td>316</td>
<td>4775</td>
<td>4.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10r</td>
<td>n10q10R</td>
<td>56</td>
<td>1253</td>
<td>1775</td>
<td>0.49</td>
<td>1223</td>
<td>22425</td>
<td>5.62</td>
<td>891</td>
<td>14550</td>
<td>11.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n10q10s</td>
<td>n10q10S</td>
<td>42</td>
<td>417</td>
<td>1603</td>
<td>0.2</td>
<td>401</td>
<td>3150</td>
<td>0.39</td>
<td>350</td>
<td>6075</td>
<td>3.24</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Time spent during column generation procedures
### Table 3: Lower bounds on instances with 10 stations and \(d_i \leq 10\)

<table>
<thead>
<tr>
<th>Instance</th>
<th>MH</th>
<th>BPA</th>
<th>gap limited</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>graph</td>
<td>(d)</td>
<td>root node</td>
</tr>
<tr>
<td></td>
<td>Gap (%)</td>
<td>t (s)</td>
<td>Gap (%)</td>
</tr>
<tr>
<td>n10q10a</td>
<td>52</td>
<td>2.98</td>
<td>33.8</td>
</tr>
<tr>
<td>n10q10A</td>
<td>48</td>
<td>0.00</td>
<td>32.04</td>
</tr>
<tr>
<td>n10q10b</td>
<td>60</td>
<td>2.49</td>
<td>39.31</td>
</tr>
<tr>
<td>n10q10B</td>
<td>32</td>
<td>1.98</td>
<td>56.53</td>
</tr>
<tr>
<td>n10q10c</td>
<td>64</td>
<td>0.00</td>
<td>30.35</td>
</tr>
<tr>
<td>n10q10C</td>
<td>44</td>
<td>4.11</td>
<td>297.73</td>
</tr>
<tr>
<td>n10q10d</td>
<td>54</td>
<td>6.90</td>
<td>31.82</td>
</tr>
<tr>
<td>n10q10D</td>
<td>42</td>
<td>1.64</td>
<td>63.92</td>
</tr>
<tr>
<td>n10q10e</td>
<td>58</td>
<td>4.01</td>
<td>103.65</td>
</tr>
<tr>
<td>n10q10E</td>
<td>58</td>
<td>0.39</td>
<td>73.12</td>
</tr>
<tr>
<td>n10q10f</td>
<td>50</td>
<td>5.62</td>
<td>24.71</td>
</tr>
<tr>
<td>n10q10F</td>
<td>36</td>
<td>0.03</td>
<td>42.65</td>
</tr>
<tr>
<td>n10q10g</td>
<td>52</td>
<td>2.55</td>
<td>29.39</td>
</tr>
<tr>
<td>n10q10G</td>
<td>34</td>
<td>6.35</td>
<td>65.21</td>
</tr>
<tr>
<td>n10q10h</td>
<td>48</td>
<td>3.07</td>
<td>109.46</td>
</tr>
<tr>
<td>n10q10H</td>
<td>62</td>
<td>4.96</td>
<td>92.53</td>
</tr>
<tr>
<td>n10q10i</td>
<td>44</td>
<td>0.47</td>
<td>442.69</td>
</tr>
<tr>
<td>n10q10I</td>
<td>60</td>
<td>6.78</td>
<td>38.54</td>
</tr>
<tr>
<td>n10q10j</td>
<td>56</td>
<td>1.79</td>
<td>38.58</td>
</tr>
<tr>
<td>n10q10J</td>
<td>42</td>
<td>0.00</td>
<td>392.1</td>
</tr>
<tr>
<td>Average</td>
<td>2.81</td>
<td>101.91</td>
<td>15.27</td>
</tr>
</tbody>
</table>

### Table 4: Upper bounds on instances with 10 stations and \(d_i \leq 10\)

<table>
<thead>
<tr>
<th>Instance</th>
<th>MH</th>
<th>BPA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>optimal sol.</td>
<td>first sol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nodes</td>
<td>t (s)</td>
<td>Gap (%)</td>
</tr>
<tr>
<td>n10q10a</td>
<td>52</td>
<td>91</td>
<td>9.33</td>
</tr>
<tr>
<td>n10q10A</td>
<td>48</td>
<td>15</td>
<td>1.92</td>
</tr>
<tr>
<td>n10q10b</td>
<td>60</td>
<td>110</td>
<td>10.81</td>
</tr>
<tr>
<td>n10q10B</td>
<td>32</td>
<td>115</td>
<td>12.10</td>
</tr>
<tr>
<td>n10q10c</td>
<td>64</td>
<td>3</td>
<td>0.93</td>
</tr>
<tr>
<td>n10q10C</td>
<td>44</td>
<td>86</td>
<td>7.87</td>
</tr>
<tr>
<td>n10q10d</td>
<td>54</td>
<td>67</td>
<td>12.65</td>
</tr>
<tr>
<td>n10q10D</td>
<td>42</td>
<td>1465</td>
<td>144.00</td>
</tr>
<tr>
<td>n10q10e</td>
<td>58</td>
<td>531</td>
<td>51.81</td>
</tr>
<tr>
<td>n10q10E</td>
<td>58</td>
<td>227</td>
<td>28.85</td>
</tr>
<tr>
<td>n10q10f</td>
<td>50</td>
<td>307</td>
<td>52.64</td>
</tr>
<tr>
<td>n10q10F</td>
<td>36</td>
<td>20</td>
<td>3.92</td>
</tr>
<tr>
<td>n10q10g</td>
<td>52</td>
<td>115</td>
<td>10.16</td>
</tr>
<tr>
<td>n10q10G</td>
<td>34</td>
<td>40</td>
<td>6.41</td>
</tr>
<tr>
<td>n10q10h</td>
<td>48</td>
<td>127</td>
<td>9.90</td>
</tr>
<tr>
<td>n10q10H</td>
<td>62</td>
<td>272</td>
<td>25.45</td>
</tr>
<tr>
<td>n10q10i</td>
<td>44</td>
<td>374</td>
<td>42.57</td>
</tr>
<tr>
<td>n10q10I</td>
<td>60</td>
<td>217</td>
<td>18.15</td>
</tr>
<tr>
<td>n10q10j</td>
<td>56</td>
<td>426</td>
<td>51.65</td>
</tr>
<tr>
<td>n10q10J</td>
<td>42</td>
<td>42</td>
<td>2.70</td>
</tr>
<tr>
<td>Average</td>
<td>25.19</td>
<td>10.58</td>
<td>5.96</td>
</tr>
</tbody>
</table>

32
5.4 Solving instances to proven optimality

Finally, we compared MH and BPA in solving instances of SPSDVRP to proven optimality. In table 5 we report our results. For each instance we report the sum of the demands $d$, the computation time for both algorithms and the number of nodes explored by our BPA.

As we can see, BPA is on average much faster than MH. In fact it is slower only for three instances, and requires more than 5 minutes of computation only for one instance.

<table>
<thead>
<tr>
<th>Instance graph</th>
<th>MH $t$ (s)</th>
<th>BPA $t$ (s)</th>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>n10q10a</td>
<td>112.53</td>
<td>20.21</td>
<td>189</td>
</tr>
<tr>
<td>n10q10b</td>
<td>43.87</td>
<td>24.69</td>
<td>110</td>
</tr>
<tr>
<td>n10q10c</td>
<td>115.24</td>
<td>12.11</td>
<td>115</td>
</tr>
<tr>
<td>n10q10d</td>
<td>45.67</td>
<td>7.09</td>
<td>7</td>
</tr>
<tr>
<td>n10q10e</td>
<td>1214.69</td>
<td>79.39</td>
<td>652</td>
</tr>
<tr>
<td>n10q10f</td>
<td>54.11</td>
<td>711.49</td>
<td>724</td>
</tr>
<tr>
<td>n10q10g</td>
<td>583.03</td>
<td>187.05</td>
<td>1886</td>
</tr>
<tr>
<td>n10q10h</td>
<td>343.51</td>
<td>113.29</td>
<td>989</td>
</tr>
<tr>
<td>n10q10i</td>
<td>211.41</td>
<td>59.69</td>
<td>537</td>
</tr>
<tr>
<td>n10q10j</td>
<td>93.56</td>
<td>106.36</td>
<td>551</td>
</tr>
<tr>
<td>n10q10k</td>
<td>60.26</td>
<td>6.79</td>
<td>43</td>
</tr>
<tr>
<td>n10q10l</td>
<td>99.01</td>
<td>10.45</td>
<td>119</td>
</tr>
<tr>
<td>n10q10m</td>
<td>292.13</td>
<td>6.70</td>
<td>44</td>
</tr>
<tr>
<td>n10q10n</td>
<td>2732.18</td>
<td>14.77</td>
<td>201</td>
</tr>
<tr>
<td>n10q10o</td>
<td>909.23</td>
<td>29.09</td>
<td>313</td>
</tr>
<tr>
<td>n10q10p</td>
<td>3596.12</td>
<td>44.04</td>
<td>388</td>
</tr>
<tr>
<td>n10q10q</td>
<td>144.44</td>
<td>18.17</td>
<td>217</td>
</tr>
<tr>
<td>n10q10r</td>
<td>57.98</td>
<td>75.25</td>
<td>616</td>
</tr>
<tr>
<td>n10q10s</td>
<td>393.80</td>
<td>12.22</td>
<td>235</td>
</tr>
</tbody>
</table>

Table 5: Results on the instances with 10 stations and $d_i \leq 10$

6 Conclusions

We proposed an exact method to tackle a SPSDVRP arising on a bike-sharing system. That corresponds to the problem of balancing bikes on a network using a fleet of homogeneous vehicles that may split the demands of each customer on the network.

In order to reduce its complexity, we modelled the problem by decomposing routes into substructures called groups. Such groups help to discard sub-optimal configurations and to limit problem symmetries. They favour also a decomposition approach from an algorithmic point of view.

To improve the lower bound given by our model, we produced an extended formulation by using Dantzig-Wolfe decomposition. We solved the linear relaxation of the extended formulation using column generation techniques, and integrated such a procedure into a branch-and-price framework. Our ad-hoc pricing algorithms, branching rules, feasibility detection routines and addi-
tional cuts proved to be computationally useful. As an overall assessment, our approach turns out to be faster and more flexible than competitors.

References


