Ann Oper Res (2017) 259:327–350
DOI 10.1007/s10479-017-2520-5

ORIGINAL PAPER

An ILP-based local search procedure for the VRP with pickups and deliveries

Agustín Montero[1] · **Juan José Miranda-Bront**[1,2] · **Isabel Méndez-Díaz**[1]

Published online: 15 May 2017

Abstract In this paper we address the Vehicle Routing Problem with Pickups and Deliveries (VRPPD), an extension of the classical Vehicle Routing Problem (VRP) where we consider precedences among customers, and develop an Integer Linear Programming (ILP) based local search procedure. We consider the capacitated *one-to-one* variant, where a particular precedence must be satisfied between pairs of origin-destination customers. We extend the scheme proposed in De Franceschi et al. (Math Program 105(2–3):471–499, 2006) for the Distance-Constrained Capacitated VRP, which has been successfully applied to other variants of the VRP. Starting from an initial feasible solution, this scheme follows the destroy/repair paradigm where a set of vertices is removed from the routes and reinserted by solving heuristically an associated ILP formulation with an exponential number of variables, named *Reallocation Model*. In this research, we propose two formulations for the Reallocation Model when considering pickup and delivery constraints and compare their behavior within the framework in terms of the trade off between the quality of the solutions obtained and the computational effort required. Based on the computational experience, the proposed scheme shows good potential to be applied in practice to this kind of problems and is a good starting point to consider more complex versions of the VRPPD.

Keywords VRP with pickups and deliveries · Integer linear programming · Matheuristic

✉ Juan José Miranda-Bront
jmiranda@dc.uba.ar

Agustín Montero
aimontero@dc.uba.ar

Isabel Méndez-Díaz
imendez@dc.uba.ar

[1] Departamento de Computación, FCEyN, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, C1428EGA Ciudad Autónoma de Buenos Aires, Argentina

[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Ciudad Autónoma de Buenos Aires, Argentina

1 Introduction and literature review

Vehicle Routing Problems (VRPs) is the name given to a wide family of Combinatorial Optimization problems related to goods distribution and service provision through a transportation network. This kind of problems arise typically in the industry and the service sector, and many different variations are defined according to practical situations and operational constraints imposed by the context of the applications. In general, VRPs are easy to formulate and to understand but difficult to solve since they are $\mathcal{NP}$-Hard problems. Toth and Vigo (2014), Golden et al. (2008) provide a very complete description as well as heuristic and exact approaches for different variants of the VRP. An important family of VRPs is the so-called VRP with Pickups and Deliveries (VRPPD), where goods are transported from an origin location towards a destination. These problems appear, among others, in the courier service industry, robotics and in transportation of people (see, e.g. Cordeau and Laporte 2007). Many different formulations and definitions can be found in the related literature. Berbeglia et al. (2007) provide an extensive and complete review, including a classification, for the VRPPD. Compared with the standard VRP, the characteristics of the VRPPD naturally induce *precedence constraints* among customers that must be satisfied by every feasible solution.

In the last decade, due to the advances obtained in the resolution of Integer Linear Programming Problems (ILPs), there has been a trend aiming to formulate and solve auxiliary subproblems as ILPs. When considering algorithms for ILPs in general, this technique is usually referred as *MIPping*. For example, *Local Branching* Fischetti and Lodi (2003), Hansen et al. (2006) is a branching technique used to strategically explore the enumeration tree in order to obtain early good quality primal solutions by exploring neighborhoods of the incumbent solution. Another example is proposed in Danna et al. (2005). Following a similar approach, *Matheuristic* (Mathematical Programming Heuristics) is the name given to the (meta) heuristics developed for particular problems that include the resolution of an auxiliary ILP at some point in the algorithm. Ball (2011) provides an extensive survey of heuristic approaches using mathematical programming models in general and Archetti and Speranza (2014) perform a similar task but focusing in VRPs.

An interesting approach is proposed in De Franceschi et al. (2006) for the exploration of an exponential neighborhood of a Distance-Constrained Capacitated VRP (DCVRP). Starting from a feasible solution for the problem, following the destroy/repair paradigm, a subset of customers is extracted from the solution and an ILP is formulated in order to re-insert them, maintaining feasibility and aiming to obtain an improved solution. The ILP formulation, named *Reallocation Model* (RM), has an exponential number of variables. However, for practical purposes, only a restricted subset of variables is heuristically generated in a pricing step and an upper limit on the execution time is imposed for the solution of the resulting ILP. This procedure is iteratively executed by introducing a randomization step for the selection of the customers to be removed in order to escape from local optimum solutions. This scheme is extensively studied in a large set of benchmark DCVRP instances, obtaining remarkable results and being able to improve the best-known solution in 13 cases.

In a follow up paper, Toth and Tramontani (2008) propose an improvement for the solution of the RM by reducing (heuristically) the size of the neighborhood and exploring it by solving, again heuristically, the Column Generation Problem associated with the LP relaxation. The authors evaluate this improved scheme in 50 benchmark CVRP and DCVRP instances obtaining good quality results and being able to find 11 new best solutions. A similar approach has been considered in Salari et al. (2010) for the Open Vehicle Routing Problem (OVRP),

where the approach showed to be effective being able to obtain in most cases the best known solutions and to find new best solutions in 10 instances. Finally, Naji-Azimi et al. (2012) insert this scheme as a Local Search operator within a Variable Neighborhood Search (VNS) framework for the m-Capacitated Ring Star Problem, where it is combined with standard operators such as *swap* and an adaptation of a Lin-Kernighan operator (see, e.g., Lin and Kernighan 1973; Helsgaun 2000). As a result, the approach shows to be competitive with the most effective algorithms present in the literature for the problem.

These experiences prove the approach to be quite effective on different variants of the VRP, with potential to be used in practice, and that it is worth extending the scheme and the RM in order to incorporate new operational constraints. Indeed, De Franceschi et al. (2006) suggest to extend the RM to incorporate *precedence constraints*, including both modeling aspects as well as an experimental study. Our aim goes in this direction. We consider a variant of the VRPPD where a fleet of vehicles has to visit an even number of customers that are grouped in pairs, one of them indicating the origin and the other the destination of a commodity, imposing a precedence that must be satisfied by every feasible route.

The contribution of the paper is threefold. Firstly, from a theoretical point of view, we extend the RM and the overall scheme for the case where the precedences mentioned above are present. Compared with the original scheme, the overall framework has to be reconsidered and it is necessary to redefine and extend some basic notation, definitions and procedures. In order to guarantee the feasibility of the solutions, the algorithm is modified to handle these particular constraints. For instance, if from a pair of pickup and delivery only one of them is removed, then it can only be reinserted within the same route due to the precedence constraints. Secondly, we propose two different ILP formulations to adapt the RM to this context. The modeling of the precedence constraints has a direct impact on the pricing step of the algorithm. In addition, we also slightly modify the pricing step of the scheme by introducing a dynamic criterion for the selection of candidate variables to be included in the final RM. Finally, from a computational standpoint, we implement the framework and compare the behavior of these two models experimentally over a large set of instances having more than 400 customers. The results obtained show that the overall approach has potential to be used in practice and also gives a strong evidence towards one of the ILPs, which obtains consistently better results than the other.

The remainder of the paper is organized as follows. In Sect. 2 we present the definition of the VRPPD considered as well as some basic notation used along the paper and a general overview of the scheme. In Sect. 3 we formulate the extensions of the RM for our problem and the adaptations required to the scheme. In Sect. 4 we show and discuss the computational results and finally, in Sect. 5 we draw some conclusions and discuss future research alternatives.

2 Basic definitions and the SERR algorithm

As mentioned in the introduction, we address the Vehicle Routing Problem with pickups and deliveries (VRPPD). Formally, let $G = (V, E)$ be an undirected complete graph, with $V = \{v_0, v_1, \ldots, v_{2n}\}$ the set of vertices and E the set of edges, where v_0 represents the depot. We consider an homogeneous fleet with m vehicles, each of them having capacity Q. Each edge $(v_i, v_j) \in E$ has an associated cost $c_{v_i v_j} \geq 0$ and the objective is to transport n different commodities. Each commodity k ($k = 1, \ldots, n$) has to be transported from v_k to v_{n+k}. Therefore, a precedence relation between vertices v_k, v_{n+k} is established, for

$k = 1, \ldots, n$, where v_k is called the pickup vertex and v_{n+k} the delivery vertex. Such a commodity requires q_{v_k} units of capacity in a vehicle. We further define $P = \{v_1, \ldots, v_n\}$ and $D = \{v_{n+1}, \ldots, v_{2n}\}$ the sets of pickups and deliveries respectively, where $V = \{v_0\} \cup P \cup D$. The demand for vertex $v_k \in V$ is denoted by d_{v_k}, where $d_{v_k} = q_{v_k}$, $v_k \in P$, and the demand for vertex $v_{n+k} \in D$ is $d_{v_{n+k}} = -q_{v_k}$, i.e., indicating that the commodity is collected at the pickup vertex v_k and occupies space in the vehicle until it is delivered at vertex v_{n+k}. We assume that the demand for vertex 0 is $d_0 = 0$. The objective is to find exactly m routes covering each vertex exactly once, satisfying the precedences between v_k and v_{n+k}, for $k = 1, \ldots, n$, without exceeding the capacity of the vehicles and at minimum total cost.

Following the three-field notation proposed in Berbeglia et al. (2007), the problem can be classified as a capacitated $1 - 1|P/D|m$, meaning that the distribution is *one-to-one*, i.e., each commodity has to be distributed from a unique origin to a unique destination; each vertex is either a pickup or a delivery, but not both; and that (exactly) m vehicles have to be used. A similar version of the problem has been considered in Hernández-Pérez and Salazar-González (2009) with an exact approach and in Rodríguez-Martín and Salazar-González (2012) in a heuristic framework. The main differences with respect to our case are that they consider the single-vehicle case, i.e. $m = 1$, and that they allow a vertex to be a pickup and a delivery at the same time. In the related literature, in many cases the VRPPD involves also *time windows*, in which customers must be visited (see, e.g., Toth and Vigo 2014).

We begin by describing the main idea behind the general scheme suggested by De Franceschi et al. (2006), called SERR (Selection, Extraction, Recombination and Reallocation), which aims to consider an exponential neighborhood of a feasible solution. The sketch of the procedure is shown in Algorithm 1. Following their notation, let $\mathcal{Z}$ be the set of all feasible solutions for the DCVRP, and consider $z_0 \in \mathcal{Z}$ an initial (feasible) solution. By applying the destroy/repair paradigm, heuristically select a set of vertices, named $\mathcal{F}$, remove these vertices from z_0, and generate a restricted solution $z_0(\mathcal{F})$ by linking consecutive vertices in z_0 after the extraction. There may be more than one removal criterion, and the strategy to combine them is reflected in Step 2. Each arc in $z_0(\mathcal{F})$ is called an insertion point, and we denote as $\mathcal{I} = \mathcal{I}(z_0, \mathcal{F})$ the set of all insertion points in $z_0(\mathcal{F})$. The neighborhood is denoted by $N(z_0, \mathcal{F})$ and is defined as follows. Let $\mathcal{S} = \mathcal{S}(\mathcal{F})$ be the set of all possible sequences, without repetitions and of any possible length, obtained by recombinations of the vertices in $\mathcal{F}$. Each $s \in \mathcal{S}$ can be assigned to one of the insertion points in $\mathcal{I}$, and to each insertion point at most one sequence can be assigned. The neighborhood $N(z_0, \mathcal{F})$ considers all feasible solutions that can be obtained by assigning sequences in $\mathcal{S}$ to insertion points $\mathcal{I}$, and the aim is to obtain an improved solution by exploring such neighborhood by solving an ILP. Clearly, the number of variables in the formulation could be extremely large and therefore only a subset of them are heuristically generated. The ILP is also heuristically solved using a general purpose ILP solver.

Figure 1 illustrates an example of the VRPPD and how the SERR could be applied in our context. We consider an instance with $V = \{0\} \cup P \cup D$, where $P = \{p_1, \ldots, p_6\}$ and $D = \{d_1, \ldots, d_6\}$, and vertex p_i denotes the pickup vertex of the delivery d_i, $i = 1, \ldots, 6$. For simplicity, we assume that the capacity is not restrictive and therefore the demands are omitted. Figure 1a shows a starting feasible solution, and in Fig. 1b the selected vertices to be extracted are marked. In addition, dashed arcs denote arcs which are also removed from the solution, due to the selected vertices. In this case, the set $\mathcal{F} = \{p_2, p_4, p_5, d_2, d_3, d_4, d_5\}$. Figure 1c shows how the restricted solution $z_0(\mathcal{F})$ is constructed, where $\mathcal{I} = \{i_1 = (0, p_1), i_2 = (p_1, d_1), i_3 = (d_1, 0), i_4 = (0, p_3), i_5 = (p_3, 0), i_6 = (0, p_6), i_7 = (p_6, d_6), i_8 = (d_6, 0)\}$. Finally, Fig. 1d shows the new solution,

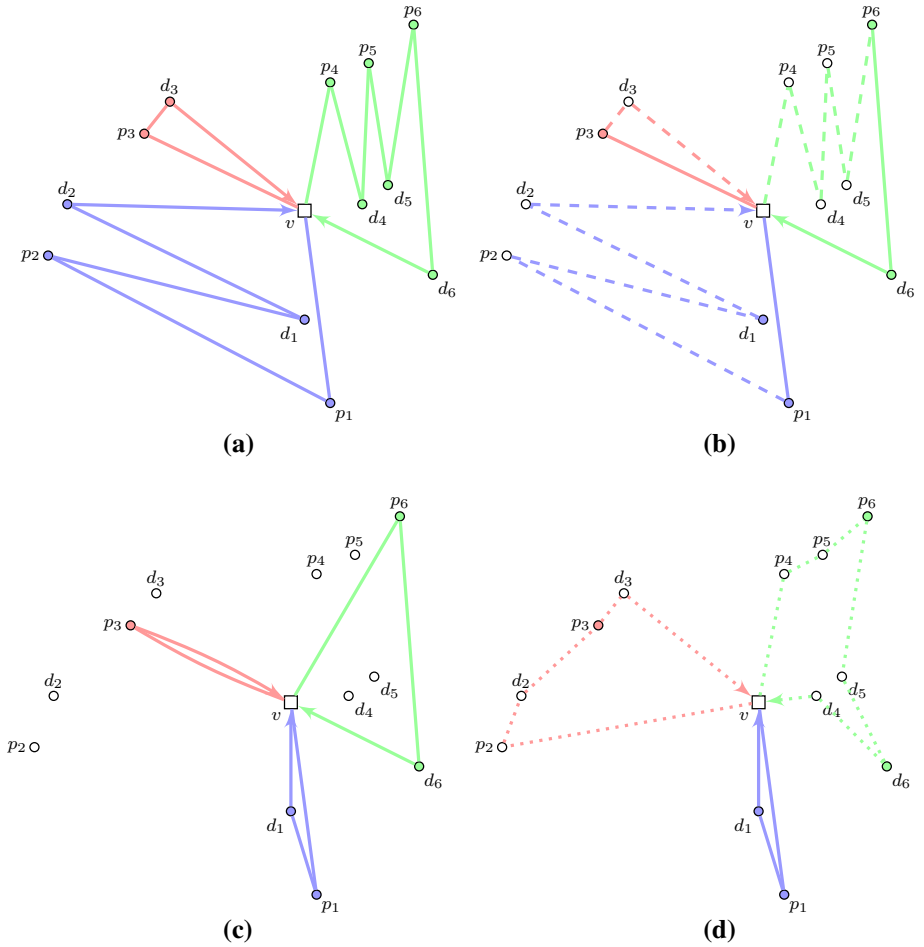

Fig. 1 Example of extraction and reallocation of vertices **a** initial solution **b** selected vertices **c** restricted solution (infeasible) **d** new solution

which is the result of assigning sequence $s_1 =< p_2, d_2 >$ to insertion point i_4, $s_2 =< d_3 >$ to i_5, $s_3 =< p_4, p_5 >$ to i_6, $s_4 =< d_5 >$ to i_7 and $s_5 =< d_4 >$ to i_8.

Finally, we note that the presence of precedence constraints in the VRPPD introduces several limitations to the framework when compared to the DCVRP or the CVRP. Clearly, the ILP formulated for the reallocation step must account for precedences, in order to generate feasible solutions. Moreover, some other steps which are quite general for the CVRP and the DCVRP must be adapted as well. In the next section we study and describe in detail each step of the scheme adapted for the VRPPD.

3 Extension to the VRP with pickups and deliveries

In this section we describe the adaptation of the SERR to the VRPPD. Due to the inclusion of precedences between pairs of vertices, the adaptation is not limited to the ILP formulation of

Algorithm 1 SERR: *Selection, Extraction, Recombination and Reallocation* (De Franceschi et al. 2006)

1. (Initialization) Apply an initial heuristic to find a feasible solution z_0.
2. (Selection) Apply one of the criteria to select the vertices to be extracted.
3. (Extraction) Extract the vertices selected in Step 2 and construct the *restricted solution* by short-cutting the consecutive extracted vertices. The set $\mathcal{I}$ contains all the edges in the restricted solution, called insertion points.
4. (Recombination) The sequences of consecutive vertices extracted in the previous step, called *basic sequences*, are stored in a *sequence pool*. Then, heuristically generate new sequences using the vertices in $\mathcal{F}$ and add them to the sequence pool. This procedure is repeated iteratively and the dual information provided by the LP relaxation of the Reallocation Model RM (see Step 5) is used to determine which sequences to keep in the pool. In addition, each sequence in the pool is heuristically associated with a subset $\mathcal{I}_s \subseteq \mathcal{I}$ of insertion points, indicating that sequences s can be inserted in insertion point $i \in \mathcal{I}_s$. In particular, a basic sequence s contains in $\mathcal{I}_s$ the original insertion point from which it has been removed, called a pivot. In this fashion, the current solution can be retrieved.
5. (Reallocation) Decide a feasible assignment of sequences in the pool to insertion points by formulating and heuristically solving an ILP (Reallocation Model, RM) using a general purpose solver. Define binary variables x_{si} to take value 1 if sequence $s \in \mathcal{S}$ is assigned to insertion point $i \in \mathcal{I}_s$. Impose an upper limit on the execution time for the solver and obtain the best solution found.
6. (Termination) If an improved solution has been found in the last K iterations, repeat from Step 2. Otherwise, return the best solution found.

the reallocation step. For instance, the vertex selection criteria and the sequence generation stage that remain unchanged for other VRPs, must be reconsidered regarding the VRPPD and further decisions are required.

3.1 Initial solution

In order to evaluate the behavior of the RM, we implement an ad-hoc heuristic procedure consisting of a greedy construction phase followed by a *Variable Neighborhood Descent* (VND) procedure. We remark, as stated in previous research regarding the RM and validated in our preliminary computational results, that the initial solution has an impact on the final solution and on the overall behavior of the scheme. Therefore, we initially consider a standard constructive approach, followed by a local search procedure. The approach is then generalized in a randomized fashion with some minor modifications. These two alternatives are used in different experiments to assess the quality of the approach and to give some insights regarding its behavior, which are shown in Sect. 4.

We begin by pointing out that the problem of finding a feasible solution for the VRPPD considered in this paper can be easily solved by grouping pairs of pickup and deliveries and generating exactly m routes -recall that we are imposing to use exactly m vehicles. In this way, the capacity of a vehicle is never exceeded since each commodity travels alone in the vehicle. Indeed, the construction phase implements this idea in a greedy fashion. Initially, for each request we compute the cost of using a route to satisfy it, and select the m requests with smaller cost. Once the m routes are generated, we iteratively add one unattended request at a time by selecting the request and the route that, when extended by adding the request at the end, generates the smallest increment in the cost of the current solution.

The second step is to apply a VND procedure to improve the current solution. For this purpose, we consider an adaptation of three local search operators proposed in Nanry and Barnes (2000) which are applied iteratively with a best move criterion, until no improvements can be achieved. These operators are:

- **Single paired insertion (SPI)**: For every request (v_k, v_{n+k}), $k = 1, \ldots, n$ in route r, remove vertices v_k and v_{n+k} and evaluate every feasible insertion in every other route $r' \neq r$. Select the request and insertion generating the least cost solution.
- **Swapping pairs between routes (SBR)**: Given two requests (v_k, v_{n+k}) and (v_l, v_{n+l}) allocated to different routes r and r', respectively, evaluate swapping (v_k, v_{n+k}) to r' and (v_l, v_{n+l}) to r, considering also the best feasible insertion within each route. Select the pair of requests generating the least cost solution.
- **Within route insertion (WRI)**: This is a route improvement operator. For every route and, for every request in it, evaluate every feasible allocation of both the pickup and the delivery and select the one generating the least cost route.

Within the VND procedure, the operators are applied in the order they are presented and we consider the next operator once no improvements can be found with the actual one. Finally, define z_0 as the solution found by this procedure when no improvement can be found by any of the operators.

As mentioned before, we consider as well a randomized version of this algorithm in order to provide some variability on the initial solution. The randomized step is rather straightforward, and consists of applying the same procedure described in the greedy step but the order of the requests is randomly determined beforehand. A predefined number of orderings is considered, obtaining as a result a basic multistart scheme. For each of these initial solutions, the VND procedure is applied with some minor modifications. The SBR operator, which is the most time consuming, is not considered. We also noted that the quality of the results is not considerably affected. In addition, the remaining operators are applied in a first-improvement fashion. These two modifications result in a faster approach than the original one.

3.2 Vertex selection strategy

The precedence between vertices imposed by requests has an impact on the vertex extraction step, and the decisions made at this point affect the definition of the neighborhood $N(z_0, \mathcal{F})$ as well as the subsequent stages of the procedure. In particular, given that precedences are established between pairs of vertices, we observe two possible scenarios regarding the extracted vertices in $\mathcal{F}$. Suppose that at least one vertex of the request (v_k, v_{n+k}) is in $\mathcal{F}$, for some v_k. Then: (i) either for some $v_k \in \mathcal{F}$ or $v_{n+k} \in \mathcal{F}$, but not both; (ii) both vertices are extracted, i.e., $v_k, v_{n+k} \in \mathcal{F}$. In the former, the reallocation of the extracted vertex is limited to its original route, which is somehow restrictive. On the contrary, the latter is more general and allows the request to be eventually reallocated in another route. Moreover, when considering a sequence $s \in \mathcal{S}$ containing both v_k and v_{n+k} we must guarantee that they appear in the correct order. In addition, removing the complete request also guarantees that the non-empty routes left in the restricted solution are always feasible, which is not necessarily true in the other case. This fact may have an impact regarding the feasibility of the ILP formulation considered and the overall scheme described in the following sections.

Recall the example in Fig. 1, in particular the extraction of vertex d_3. Since the pickup vertex p_3 is left in the restricted solution, d_3 can only be assigned to the route where p_3 is present and, furthermore, it must be assigned to an insertion point that appears after p_3, otherwise the solution would be infeasible. Regarding the second situation, observe that sequence $s_1 =< p_2, d_2 >$ satisfies the precedence imposed by the request. This intuition is formalized in the following result.

Proposition 1 *Let $z_0 \in \mathcal{Z}$, $\mathcal{G} \subseteq \mathcal{F} \subseteq P \cup D$. Then, $N(z_0, \mathcal{G}) \subseteq N(z_0, \mathcal{F})$.*

Proof We show that for any feasible solution in $N(z_0, \mathcal{G})$ we can construct an equivalent one in $N(z_0, \mathcal{F})$. Consider an initial solution z_0 and a feasible solution $z_1 \in N(z_0, \mathcal{G})$. For each insertion point $i = (a, b) \in \mathcal{I}(z_0, \mathcal{F})$, given that $\mathcal{G} \subseteq \mathcal{F}$, either one of the following holds: (i) $i = (a, b) \in \mathcal{I}(z_0, \mathcal{G})$, or (ii) a sequence of consecutive insertion points $i_0 = (a, u_0), i_1 = (u_0, u_1), \ldots, i_k = (u_{k-1}, b) \in \mathcal{I}(z_0, \mathcal{G})$, with $u_1, \ldots, u_k \in \mathcal{F}$ since in both cases we are starting from the same initial solution z_0. To construct z_1, starting from $z_0(\mathcal{F})$, we proceed as follows. Consider each insertion point $i = (a, b) \in \mathcal{I}(z_0, \mathcal{F})$ separately. In case condition (i) holds, then assign exactly the same sequence as in z_1, if any. Otherwise, condition (ii) holds and let, w.l.o.g., $\bar{s}_0, \ldots, \bar{s}_k$ be the sequences assigned to $i_0, \ldots, i_k$, respectively. Eventually, some of them (or even all) can be the empty sequence, indicating that no assignment has been made in z_1 to the corresponding insertion point. Then, construct the sequence $\bar{s}$ as the concatenation among the intermediate vertices and the assigned sequences, i.e., $\bar{s} =< \bar{s}_0, u_0, \bar{s}_1, u_1, \ldots, u_{k-1}, \bar{s}_k >$, where we abuse notation and allow the concatenation of sequences with simple elements. Then, the sequence of vertices between a and b is the same as in z_1. Applying this same reasoning for all insertion points we obtain z_1 the resulting feasible solution. □

Based on this discussion we decided to remove requests instead of vertices, obtaining as a result a more general neighborhood.

De Franceschi et al. (2006) suggest three different schemes for (randomly) selecting the vertices to be included in $\mathcal{F}$ for the DCVRP, which are also considered with minor modifications in Toth and Tramontani (2008), Salari et al. (2010):

- RANDOM- ALTERNATE: randomly selects some routes and, also randomly, removes vertices depending on the parity of the position in the solution;
- SCATTERED(p): each vertex can be removed from the solution with uniform probability p;
- NEIGHBORHOOD: randomly select some vertices $v \in V$ and consider a small neighborhood for each of them. Remove some of the vertices in such neighborhood according to a pre-defined criterion (we refer the reader to De Franceschi et al. (2006) for a detailed explanation).

We adapt these schemes to the VRPPD in the following fashion: execute the selection scheme over the vertices in P and, whenever a pickup vertex v_k is selected, extract also the corresponding delivery v_{n+k} defining the request. All the three schemes have been implemented and tested, including several combinations among them.

3.3 Reallocation models

In this section we formalize some of the ideas mentioned in the previous sections and propose two ILPs to explore $N(z_0, \mathcal{F})$. Recall that we denote by $\mathcal{Z}$ the set of all feasible solutions for the VRPPD, $z_0 \in \mathcal{Z}$ an initial feasible solution, $\mathcal{F}$ the set of extracted vertices from z_0 and $z_0(\mathcal{F})$ the resulting restricted solution having the set of insertion points $\mathcal{I}$, and $\mathcal{S}$ the set of all feasible sequences composed by vertices in $\mathcal{F}$. We further denote with $\mathcal{R}$ to the set of routes in a (restricted) solution.

Given an insertion point $i = (a, b) \in \mathcal{I}$, we say that i allocates vertices $\{v_j \in \mathcal{F} : j = 1, \ldots, h\}$ through sequence $s = (v_1, \ldots, v_h) \in \mathcal{S}$ if arc (a, b) in the restricted solution is replaced by arcs $(a, v_1), (v_1, v_2), \ldots, (v_h, b)$ in the new solution. In order to obtain a new feasible solution, sequences $s \in \mathcal{S}$ considered for reallocation have to satisfy some basic properties. In the VRPPD, precedences clearly have an impact in the order in which vertices must appear in the sequence as well as capacities, since visiting a delivery vertex increases

the remaining capacity in the vehicle. For a sequence $s = (v_1, \ldots, v_h)$, let $V(s) \subseteq \mathcal{F}$ be the set of vertices in s and we define the *total demand of* s as

$$d(s) = \sum_{j=1}^{h} d_{v_j} \tag{1}$$

and *maximum demand consumption of* s as

$$\bar{d}(s) = \max\left\{0, \max_{l=1,\ldots,h} \sum_{j=1}^{l} d_{v_j}\right\}. \tag{2}$$

Therefore, a sequence of vertices s is feasible if the following conditions are satisfied:

1. for every pair of vertices defining a request $(v_k, v_{n+k}) \in V(s)$, for some $k = 1, \ldots, n$, v_k must appear before v_{n+k}, and
2. it does not exceed the capacity of the vehicle at any point in the sequence, i.e., $\bar{d}(s) \leq Q$.

In addition to feasibility, we must account for the incurred cost when allocating a sequence $s \in \mathcal{S}$ into insertion point $i = (a, b) \in \mathcal{I}$. For $s = (v_1, \ldots, v_h)$, let

$$c(s) = \sum_{j=1}^{h-1} c_{v_j v_{j+1}}$$

denote the cost of sequence s and $\gamma_{si} = c_{av_1} + c(s) + c_{v_h b} - c_{ab}$ denote the cost incurred when replacing arc (a, b) by sequence s, including the cost of the linking arcs with the endpoints of i. This definition differs from the previous research regarding the CVRP and DCVRP, where γ_{si} considers the cost of the best insertion between s and the reverse of s, reducing in this way the number of sequences. In the VRPPD, however, if s includes among its vertices a complete request, then by the condition established in 1 the reverse of s is not feasible. We further denote with $\mathcal{S}(v) \subseteq \mathcal{S}$ for $v \in \mathcal{F}$ as the set of sequences containing vertex v and $\mathcal{S}_E(v)$ to the set of sequences containing vertex v but not its corresponding delivery if v is a pickup, or its corresponding pickup if v is a delivery. For $s \in \mathcal{S}$, we define the set $V_P^E(s) \subseteq P \cap V(s)$ as the set of pick-up vertices appearing in s that are *exclusive*, that is, vertices v_k such that the corresponding delivery $v_{n+k} \notin V(s)$. Similarly, we define $V_D^E(s) \subseteq D \cap V(s)$ for the deliveries.

Regarding the restricted solution $z_0(\mathcal{F})$, we also need to introduce some notation. A route $r \in \mathcal{R}$ has an orientation and we define $\mathcal{I}(r)$ to the set of insertion points in r, where we also assume that insertion points in $\mathcal{I}(r)$ are numbered according to their relative position in r. Therefore, given $i, j \in \mathcal{I}(r)$, $i \neq j$, either $i < j$ or $j < i$. This is consistent with the previous remark regarding the orientation of sequences, since otherwise we cannot guarantee the new solution to be feasible. Conversely, given $i \in \mathcal{I}(r)$ we define $r_i = r$, i.e., r_i is the unique route containing insertion point $i \in \mathcal{I}$.

To account for the capacity constraint, given an insertion point $i \in \mathcal{I}(r)$ we define the *accumulated restricted demand for* i as

$$d_a(i) = \sum_{\substack{j=(a_j,b_j)\in\mathcal{I}(r)\\ j\leq i}} d_{a_j}, \tag{3}$$

which accounts for the accumulated demand of the restricted solution if no sequence is assigned before i. Finally, we consider the case where all customers in a route are removed. In this case, a unique insertion point $i = (0, 0)$ is considered. We abuse notation and denote

$r = \{0\}$ to indicate that the route has only the depot, and $i = (0, 0)$ is the unique insertion point in the route.

Let binary variable $x_{si}, s \in \mathcal{S}$ and $i \in \mathcal{I}$, take value 1 iff sequence s is allocated at insertion point i. We proceed to show a first ILP formulation to explore $N(z_0, \mathcal{F})$ in the case of the VRPPD that we name RMPD.

$$\min \sum_{s\in\mathcal{S}} \sum_{i\in\mathcal{I}} \gamma_{si} x_{si} \tag{4}$$

$$\text{s.t.} \sum_{s\in\mathcal{S}(v)} \sum_{i\in\mathcal{I}} x_{si} = 1 \quad v \in \mathcal{F} \tag{5}$$

$$\sum_{s\in\mathcal{S}} x_{si} \leq 1 \quad i \in \mathcal{I} \tag{6}$$

$$\sum_{s\in\mathcal{S}} \bar{d}(s)\, x_{si} \leq Q - d_a(i) - \sum_{\substack{j<i \\ j\in\mathcal{I}(r_i)}} \sum_{s\in\mathcal{S}} d(s)\, x_{sj} \quad i \in \mathcal{I} \tag{7}$$

$$1 \leq \sum_{s\in\mathcal{S}} x_{si} \quad r \in \mathcal{R}, r = \{0\}$$

$$i \in r \tag{8}$$

$$x_{s_1,i} + x_{s_2,j} \leq 1 \quad \exists\, v_k \in P \cap \mathcal{F},\ s_1 \in \mathcal{S}(v_k),\ ;\ s_2 \in \mathcal{S}(v_{n+k}), r_i \neq r_j,\ i, j \in \mathcal{I} \tag{9}$$

$$x_{s_2,i} + x_{s_1,j} \leq 1 \quad \exists\, v_k \in P \cap \mathcal{F},\ s_1 \in \mathcal{S}(v_k),\ s_2 \in \mathcal{S}(v_{n+k}),\ r_i = r_j,\ i < j,\ i, j \in \mathcal{I} \tag{10}$$

$$x_{si} \in \{0, 1\} \quad s \in \mathcal{S}, i \in \mathcal{I} \tag{11}$$

The objective function (4) minimizes the total cost obtained by reallocating sequences of vertices, aiming to obtain the best feasible reinsertion. Constraint (5) force each removed vertex to be covered by exactly one sequence, and therefore visited exactly once in the new solution. Constraint (6) establish that at most one sequence is assigned to each insertion point, and thus guaranteeing the correct computation of the cost of the new solution. Restrictions (7) force the assignment of a sequence to an insertion point to not exceed the capacity of the vehicle, considering also the (possible) assignments made to insertion points appearing before in the route. Constraint (8) guarantee that exactly m routes are used in the solution, by forcing at least one assignment if all vertices have been removed from a route. Finally, constraints (9) and (10) account for the precedences imposed by requests. Given a request (v_k, v_{n+k}), the former are referred as *inter route precedence constraints* and establish that two sequences containing the pickup and the delivery, respectively, cannot be assigned to different routes. The latter are referred as *intra route precedence constraints* and establish that, within a route, these two sequences must be assigned to insertion points satisfying the corresponding precedence. Finally, constraint (11) define the decision variables as binaries.

The first two sets of constraints are exactly the same as in the formulation proposed by De Franceschi et al. (2006). Constraint (7) have been adapted to capture the particular characteristics of the VRPPD regarding the load of the vehicle. Constraints (8), (9) and (10) are new with respect to the previous research. Although they are an intuitive and natural way of modeling precedences, constraints (9) and (10) present a few drawbacks from a computational standpoint. Firstly, we note that the number of constraints to be included in the ILP formulation can increase considerably since constraints are defined by pairs of sequences

and pairs of insertion points. This require a significant computation time to generate as well as to solve the LP relaxation.

The second drawback concerns the heuristic resolution of the RMPD where, as proposed in De Franceschi et al. (2006), the ILP is not solved to optimality but instead a restricted set of variables are generated, added to the formulation and then a general purpose ILP solver is used. To generate the variables, De Franceschi et al. (2006) suggest to heuristically generate sequences and, based on the dual information, compute the reduced costs to decide which variables to include in the ILP formulation. Toth and Tramontani (2008) suggest for the DCVRP to use column generation techniques by heuristically solving the associated column generation subproblem. Regarding the RMPD, these two approaches need to be carefully reconsidered. Both the pricing step as well as the column generation mentioned before require the dual information to be available. If the RMPD has been populated only with a restricted set of variables, the constraints (9) and (10) present in the current restricted formulation will refer only to variables in this set. On the contrary, constraints (9) and (10) concerning variables which have not been included so far will not be part of the formulation. Therefore, the reduced cost of a candidate variable to be included in the restricted set cannot be computed using the current present dual information in the restricted ILP formulation. This type of problems are known as *Column-Dependent Rows* (CDR) and a possible approach could be to adapt some of the strategies proposed in, e.g., Muter et al. (2012), Muter et al. (2012) to formulate the RMPD. However, we consider that such an approach would be too complex and very time consuming when in the end the resulting ILP is solved heuristically.

In order to adapt the pricing scheme proposed in De Franceschi et al. (2006), which is described in the next section, we consider approximating the reduced cost heuristically using the information available at the time. Consider a *reduced* RMPD having only a subset of variables and constraints and that the corresponding LP relaxation has been solved. Let also $(\tilde{\pi}^1, \tilde{\pi}^2, \tilde{\pi}^3, \tilde{\pi}^4)$ be the vector of dual variables associated with constraints (5), (6), (7) and (8), respectively. Let $s \in \mathcal{S}$ and $i \in \mathcal{I}$ such that x_{si} has not been already added to the reduced LP, then the *truncated reduced cost* of variable x_{si}, $\hat{rc}_{si}$, can be computed as:

$$\hat{rc}_{si} := \begin{cases} \gamma_{si} - \sum\limits_{v \in V(s)} \tilde{\pi}_v^1 - \tilde{\pi}_i^2 - \bar{d}(s)\tilde{\pi}_i^3 + \tilde{\pi}_{r_i}^4 & \text{if } r_i = \{0\} \\ \gamma_{si} - \sum\limits_{v \in V(s)} \tilde{\pi}_v^1 - \tilde{\pi}_i^2 - \sum\limits_{\substack{i<k \\ k \in \mathcal{I}(r_i)}} d(s)\tilde{\pi}_k^3 - \bar{d}(s)\tilde{\pi}_i^3 & \text{otherwise.} \end{cases} \tag{12}$$

To overcome these issues, we strengthen the ILP and substitute the precedence constraints (9) and (10) by a new set of constraints. Indeed, the idea relies in the fact that for a request (v_k, v_{n+k}), by constraint (5) there will be exactly one sequence containing pickup vertex v_k and exactly one sequence containing delivery vertex v_{n+k}. As a result, both inter and intra route precedences can be defined by including one constraint for each combination of a request and an insertion point. This reduces the size of the formulation and, with a proper initialization of the ILP formulation, allows the exact computation of the reduced cost of a variable. The resulting formulation, named *Strengthened RMPD* (S-RMPD), is shown below.

$$\begin{aligned} &\min \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{I}} \gamma_{si} x_{si} \\ &\text{s.t.} (5) - (8) \\ &\sum_{\substack{j \in \mathcal{I} \\ r_j \neq r}} \sum_{s \in \mathcal{S}_E(v_{n+k})} x_{sj} + \sum_{i \in \mathcal{I}(r)} \sum_{s \in \mathcal{S}_E(v_k)} x_{si} \leq 1 \qquad v_k \in P \cap \mathcal{F},\ r \in \mathcal{R} \end{aligned} \tag{13}$$

$$\sum_{s\in\mathcal{S}_E(v_k)} x_{si} \leq \sum_{\substack{j>i \\ j\in\mathcal{I}(r_i)}} \sum_{s\in\mathcal{S}_E(v_{n+k})} x_{sj} \quad v_k \in P\cap\mathcal{F},\ i\in\mathcal{I}$$

$$x_{si}\in\{0,1\} \qquad s\in\mathcal{S}, i\in\mathcal{I} \tag{14}$$

Consider again $(\tilde{\pi}^1,\tilde{\pi}^2,\tilde{\pi}^3,\tilde{\pi}^4)$ the vector of dual variables associated with constraints (5), (6), (7) and (8), respectively. Let also (ρ^1,ρ^2) be the dual variables associated with constraints (13) and (14), respectively. Therefore, the reduced cost $\bar{rc}_{si}$, for $s\in\mathcal{S}$ and $i\in\mathcal{I}$, assuming $|r_i|>0$, is

$$\begin{aligned}\bar{rc}_{si} = \gamma_{si} &- \sum_{v\in V(s)} \tilde{\pi}^1_v - \tilde{\pi}^2_i - \sum_{\substack{i<k \\ k\in\mathcal{I}(r_i)}} d(s)\tilde{\pi}^3_k - \bar{d}(s)\tilde{\pi}^3_i + \tilde{\pi}^4_{r_i} \\ &- \sum_{v_k\in V^E_P(s)} \rho^1_{v_k r_i} - \sum_{\substack{r_j\in\mathcal{R} \\ r_j\neq r_i}} \sum_{\substack{v_k: \\ v_{n+k}\in V^E_D(s)}} \rho^1_{v_k r_j} \\ &- \sum_{v_k\in V^E_P(s)} \rho^2_{v_k i} + \sum_{\substack{i>l \\ l\in\mathcal{I}(r_i)}} \sum_{\substack{v_k: \\ v_{n+k}\in V^E_D(s)}} \rho^2_{v_k l}\end{aligned} \tag{15}$$

where for the sake of notation the case having $r_i\neq\{0\}$ is omitted and can be easily computed similarly to (12).

Note that, as opposed to the RMPD, Eq. (15) computes the reduced cost of a variable x_{si}. In addition, the number of constraints in the formulation is known a priori and does not depend on the number of sequences generated. This is particularly important because it has a direct impact on the quality of the heuristic set of variables considered, as explained in the pricing stage in the next section, as well as in the size of the resulting ILP formulation.

3.4 Vertex recombinations and construction of the RM

We finally present the details involved in the resolution of the RM, starting with the construction of the ILP formulation. As mentioned in the introduction, only a subset of the variables x_{si} is considered, and such subset is constructed in a heuristic fashion. We remark that the procedure is similar for the two ILP formulations presented in the previous section, RMPD and S-RMPD, and the difference relies in the computation of the reduced costs in each case, as expressed in Eqs. (12) and (15), respectively.

Initially, we associate each basic sequence to its pivot position, in order to guarantee that the original solution can be reconstructed, and that any solution obtained will be at least as good as the starting solution. In addition, for implementation purposes, we add an arbitrary set of variables to ensure that all constraints have at least one variable with a nonzero coefficient and that the ILP is feasible. For each insertion point $i\in\mathcal{I}$ in the restricted solution, we consider unitary sequences $s=<v>$, $v\in\mathcal{F}$, and add the variables corresponding to the 25% sequences with the smallest insertion cost.

Having these variables in the model, as well as their corresponding constraints, solve the LP formulation and compute the dual information in order to heuristically generate candidate variables and, based on their reduced cost, select the best ones in a *pricing step*. Once a new subset of variables has been included, the restricted LP is re-optimized in order to update the dual information available. This procedure is repeated until a particular stopping criterion is met.

The pricing step considers each insertion point independently. For each $i \in \mathcal{I}$, sequences $s \in \mathcal{S}$ of at most length L_{max} are generated and the reduced cost of the corresponding variable x_{si}, rc_{si}, is used to determine whether the variable is a candidate one or not. Limiting the size of the sequences has an impact both on the computation time required for building the model, which cannot be neglected, and also on the behavior of the ILP formulation. For example, considering large sequences may produce many incompatibilities among them due to precedence violations. The sequences are constructed by iteratively increasing their length and, in each iteration, the best N_{min} sequences -in terms of the reduced costs- are added to the formulation and used in the next iteration. From the remaining sequences, those having the corresponding reduced cost below a threshold value RC are considered as well, selecting at most N_{max} of them. The parameters L_{max}, N_{min}, N_{max} and value RC are determined experimentally.

For the parameter RC, we propose a modification regarding the procedure developed by De Franceschi et al. (2006). Instead of considering a fixed value, we suggest to dynamically update this threshold using the average of the candidate variables selected so far, and to reset this value whenever new dual information becomes available. On preliminary computational results, this modification produced very good results compared to the static version with different values. The motivation behind this change is that, as opposed to the other parameters, RC is very sensitive to the characteristics of the instance and defining a general value seems to be difficult to adjust experimentally. To avoid confusions, we redefine this value and denote it as rc^{dyn}. Before starting, $rc^{\text{dyn}} = \infty$ and initially it is computed as the average of the first $N_{min} + N_{max}$ variables with the smallest reduced cost.

The sketch of the pricing step for a particular insertion point $i \in \mathcal{I}$ is shown in Algorithm 2. We remark that the LP relaxation is re-optimized after the pricing step has been applied for all insertion points, and we set $rc^{\text{dyn}} = \infty$ and start the pricing procedure again. Intuitively, the value of rc^{dyn} is recomputed from scratch whenever the dual information is updated. The procedure is applied iteratively until a limit of 5 consecutive LP re-optimizations without improvements is reached, or a maximum of 200 iterations overall. Then, the resulting ILP formulation is constructed and solved by a general purpose solver. To avoid generating duplicated variables, hashing techniques were used in the implementation of the algorithm.

Algorithm 2 VRPPD PRICING STEP

Input: insertion point i, $\tilde{\pi}$ optimal dual variables for the current LP, N_{min}, N_{max}, L_{max}.

1. (Initialization) Set $L := 0$, $S := \{<>\}$
2. $L = L + 1$ represents the actual size of the generated sequences.
3. (Generation) For each $s \in S$, $v \in \mathcal{F}$ such that $v \notin s$, generate all sequences of size L obtained by feasible insertions of v into s, that is, without violating the maximum capacity and the precedences.
4. (Evaluation) For each s generated in Step 3, consider the variable x_{si} and evaluate its corresponding reduced cost $rc_{si}(\tilde{\pi})$. Set $S = \emptyset$.
5. (Selection) Sort sequences increasingly according to their reduced cost rc_{si}. Insert in S the first N_{min} sequences, and at most N_{max} from the remaining such that $rc_{si}(\tilde{\pi}) \leq rc^{\text{dyn}}$. Insert in the RM the variables x_{si} corresponding to the selected variables. Update rc^{dyn} using the reduced costs of the candidate variables.
6. (Termination) If $L < L_{max}$, start again from Step 2. Otherwise, terminate the procedure.

4 Computational experiments

The method described in the previous section has been implemented in C++, using g++ 4.8.2, CPLEX 12.4 as a general purpose LP and ILP solver, and a CentOS 6.4 as operating system. The experiments are run on an Intel Core i7 3.40 GHz with 16Gb of RAM.

Regarding the instances, we report over four different sets from related problems, previously adapted to the version of the VRPPD considered in this paper:

- *Set 1:* Instances for the DCVRP from the VRPLIB [1], where the pairs defining each request are randomly generated. The number of vehicles is fixed to the maximum established in each instance.
- *Set 2:* Homberger instances[2], adapted by discarding time windows and randomly generating the pairs defining each request, similarly to the previous case. We consider all instances with $n = 200$, and a subset of the instances with $n = 400$. We set $m = 20$ for the instances having $n = 200$, and $m = 10$ when $n = 400$. This decision relies on the fact that, on preliminary experiments, we observed that the solutions tend to be mainly small routes satisfying each of them a few requests when having many routes. Therefore, in order to allow more flexibility to the solution set, we decided to reduce the number of routes.
- *Set 3:* A subset of the instances for the single vehicle case and infinite capacity, known as PDTSP, used in Dumitrescu et al. (2008)[3]. From the instances used in this paper, we consider the subset of random instances. We use these instances to assess for the quality of the results obtained by our approach given that the optimal solution is known for many of them. We remark that PDTSP is a particular case and some of the constraints are relaxed (i.e., capacity and inter-route movements).
- *Set 4:* PDVRPTW instances from Li and Lim (2003), also considered in Ropke and Pisinger (2006)[4]. Similar to Set 1, the information regarding the time windows is discarded. Since our problem requires a fixed number of routes, we set this parameter using the information of the Best Known Solution (BKS) for each instance.

In sets 1 and 2, the demand of a request is computed as the average of the demands from the corresponding two customers in the original instance. This prevents the instance to become infeasible due to capacity limitations. For Set 1, for the instances having an odd number of customers the one having the highest number is discarded.

Regarding the methods evaluated, we consider two approaches following the scheme described in the previous section and where the difference relies in the ILP formulation used for the reallocation step. Therefore, we abuse notation and refer to the methods as RMPD and S-RMPD to account for the standard and lifted RMs, respectively. We also remark that we do not restrict the execution time of the ILP solver, in order to study and obtain a deeper insight on the behavior of the formulation in practice. Preliminary experiments were conducted in

[1] Instances retrieved from http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html. Last access: July 2016.

[2] Instances retrieved from http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-with-time-windows-instances/. Last access: July 2016.

[3] Instances retrieved from http://www.diku.dk/~sropke/DataSets/. Last access: July 2016.

[4] Instances retrieved from http://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/100-customers/. Last access: July 2016. Due to some differences regarding the solutions obtained in some of the instances with respect to the ones published by Li and Lim (2003), we consider the optimal and best known solutions for the instances in this site.

order to define the combination of parameters for each method. We observed that the best results were obtained by SCATTERED(p) using $p = 0.35$. Therefore, this scheme will be the one considered for the experiments. The intuition behind the length of the sequences is the following. For RMPD, having long sequences generate a large number of precedence constraints, which results in a considerable time to generate the ILP formulation. Regarding the S-RMPD, the lifted precedence constraints allow to consider longer sequences and the impact regarding the size of the resulting ILP formulation is manageable. As to the values of N_{min} and N_{max}, we observe better results by deciding whether to include or not a variable in the formulation based on the dual information. This is of course influenced by the presence of the dynamic pricing. The resulting combinations are:

- RMPD: We set $L_{max} = 3$, $N_{min} = 1$, $N_{max} = 6$, SCATTERED(p) with $p = 0.35$,
- S-RMPD: We set $L_{max} = 7$, $N_{min} = 1$, $N_{max} = 6$, SCATTERED(p) with $p = 0.35$.

The computational results are divided in two parts. First, we conduct a series of experiments over sets 1 and 2 to compare the results obtained for RMPD and S-RMPD, where the latter produces the best results in terms of quality and computational times. Based on these results, we conducted further experiments on a larger set of instances to analyze the particular behavior of S-RMPD.

4.1 Comparison between RMPD and S-RMPD

For each instance, both methods start from the same initial feasible solution as described in Sect. 3.1. We report the value of the solution obtained by the initial heuristic and the computation time required. For both RMPD and S-RMPD, we report the improvement percentage obtained at two different moments during the process, i.e., a partial improvement %Im-p and an overall improvement %Im-t, to be specified later, as well as the overall number of iterations #it and the total time spent for the optimization of the ILP formulation, ILP-T. Regarding the execution time, for instances having less than 400 customers we impose a maximum of 3600 s. However, since the time required for both the construction and the resolution of the ILP is considerable and depends on n, for instances having $n \geq 400$ we impose a maximum of 9000 s for the execution time. We remark that this value represents the limit to start the optimization of an ILP, and both methods may eventually use some extra time to perform the last optimization. In addition, we let %Im-p be the value of the improvement obtained at 900 s when $n < 400$, and the improvement obtained after 3600 s when $n \geq 400$. This partial improvement aims to give an insight of the behavior of the algorithms during the whole process and not only at the end. In both cases, %Im-t represents the overall improvement obtained for the instance. In the tables shown below, the best improvements are shown in bold.

We show in Table 1 the results obtained for Set 1. The columns n and m stand for the number of customers and vehicles considered, respectively. The main message of this table is that S-RMPD obtains better results than RMPD in almost all instances, and in the remaining ones the values obtained by S-RMPD are very close to the ones produced by RMPD. For instances having a few vertices, the methods find similar solutions and only small differences can be observed in some of the instances, always in favor of S-RMPD. For medium and large instances, i.e. $n \geq 200$, the improvements obtained by S-RMPD are significantly larger compared to those obtained by RMPD, where in almost all cases the improvements are at least the double and we remark that on some instances it behaves also three or four times better. This difference is justified by the number of iterations performed, where we can observe that S-RMPD is able to perform more iterations than RMPD, and also in the time spent for

Table 1 Computational results on VRPLIB instances

Instance	n	m	Start		RMPD				S-RMPD			
			Obj.	Time	%Im-p	%Im-t	#it	ILP-T	%Im-p	%Im-t	#it	ILP-T
D022-04g	20	4	485.70	0.0	2.9	2.9	22581	277.4	2.9	2.9	70686	110.02
D023-03g	22	3	815.57	0.0	0.0	0.0	23543	284.6	0.0	0.0	63221	131.67
D030-03g	28	3	687.97	0.0	0.3	0.3	13085	383.1	**3.2**	**3.2**	31461	179.39
D033-04g	32	4	949.06	0.0	2.8	2.8	8146	460.4	2.8	2.8	18962	191.62
D051-06c	50	6	830.42	0.0	8.5	8.5	2471	448.9	**9.1**	**9.1**	5744	79.52
D076-11c	74	11	1269.05	0.1	2.5	2.5	669	670.2	2.5	**2.6**	2137	169.92
D101-09c	100	9	1322.06	0.3	5.5	5.9	374	645.3	**5.6**	5.9	885	116.01
D101-11c	100	11	1720.02	0.1	11.4	12.5	329	780.0	**12.6**	**12.6**	887	131.81
D121-11c	120	11	1795.78	1.6	0.4	**1.7**	232	657.5	**1.5**	1.5	497	107.39
D151-14b	150	14	1826.37	2.7	0.1	0.1	106	840.3	**0.2**	**0.2**	388	181.81
D151-14c	150	14	1702.51	1.7	0.4	0.6	118	730.7	**2.2**	**2.2**	388	134.69
D200-18b	198	18	2344.25	3.6	1.3	1.4	35	1120.1	**4.5**	**7.4**	219	297.88
D200-18c	198	18	2364.66	4.1	0.8	4.2	8	1608.0	**8.7**	**8.7**	225	249.63
D201-05k	200	5	10916.14	27.4	3.9	4.9	90	533.5	**6.8**	**8.0**	219	112.8
D241-10k	240	10	8926.83	41.9	0.2	0.8	33	770.2	**5.4**	**10.1**	96	240.85
D281-08k	280	8	14354.97	56.5	0.5	0.5	29	531.6	**2.0**	**4.0**	86	52.12
D321-10k	320	10	13967.40	130.7	0.3	0.7	16	658.3	**0.8**	**1.8**	71	73.94
D361-09k	360	9	17681.01	298.7	0.2	0.3	10	572.9	**3.6**	**5.2**	33	106.59
D401-10k	400	10	18759.52	1205.1	0.7	3.9	30	4376.5	**4.2**	**8.3**	107	913.1
D441-11K	440	11	19970.15	879.0	0.1	2.1	19	4902.0	**3.5**	**6.3**	75	802.44
D481-12k	480	12	25212.01	1796.0	0.7	2.7	15	4026.5	**3.7**	**11.2**	61	1019.07

solving the corresponding ILP formulation. These two factors, combined with the fact that S-RMPD allows to exactly assess the quality of a variable by the complete computation of its corresponding reduced cost, increase the chances of finding better solutions. Furthermore, we can make the following observation: in almost all instances, the solution found by S-RMPD in the intermediate evaluation, reported in column %Im-p is better than the final solution obtained by RMPD. This suggests that the lifting formulation produces, as expected, better results in less computation times.

In Table 2 we present the results for the instances in Set 2 where $n = 200$ and $m = 20$. The tendency is similar as in the previous experiment, with S-RMPD outperforming RMPD in 58 out of the 60 instances. Furthermore, even considering the results obtained after 900 s S-RMPD obtains better results than RMPD. The latter is only able to obtain better improvements in 3 of the instances, and the results obtained by S-RMPD are comparable. We remark the special case of instance RC2_2_1, where neither RMPD nor S-RMPD are able to find any improvement. We conducted further experiments, firstly varying the order of the local search operators, and noted that the initial solutions are indeed worse than the actual (around 5%) and that the final results obtained are approximately 1% below the reported one. In addition, we tested also 5 different seeds for the current configuration, and only in one case the solution is 0.66% better than the reported one. This suggests that, for this case, the initial solution is indeed a good one and obtaining an improved solution is difficult.

Finally, we report in Table 3 the results obtained for the 12 instances in Set 2 consisting of 400 customers. S-RMPD outperforms RMPD in 11 out of the 12 cases (some of them remarkably better, for example instances C1_4_2 and RC1_4_2), and it is comparable in the only instance where RMPD produced better results. This is consistent with the previous experiments and we observe the same tendency. However, in this case the improvements obtained by S-RMPD are significantly higher compared with the ones produced by RMPD. This seems to be related with the number of iterations performed and the time required to solve the corresponding ILP formulation. Indeed, once the ILP is defined, we can clearly observe that S-RMPD requires, in general, only a small portion of the time required by RMPD. Our conjecture is that the variables generated for the S-RMPD are of better quality than those generated for the RMPD, and as expected the lifting procedure applied over the precedence constraints produces a tighter formulation.

4.2 Evaluation of S-RMPD

Given the results reported in the previous section, we now concentrate on the particular behavior of the S-RMPD. Firstly, we analyze the impact of the quality of the initial solution on the final results obtained by the algorithm. For this purpose, we consider the randomized version of the initial heuristic described in Sect. 3.1. Within this framework, we generate 10 different initial solutions, on which the modified VND procedure is applied. From the resulting solutions, we select the best three among them as starting solutions. We name this approach S-RMPD(10,3). In addition, in order to obtain a fair comparison regarding the quality of the solutions and the computing time required for their computation, the total time assigned for the execution of S-RMPD is evenly divided among the 3 starting solutions for S-RMPD(10,3). Regarding the results, for S-RMPD(10,3) we report the objective value of the initial solution which, after applying the overall framework, resulted in the best solution found by S-RMPD(10,3). For both methods, we report the objective value of the best solution found (Obj-t) instead of the improvement percentages. For S-RMPD(10,3), we also report the average computing time for the 10 initial solutions.

Table 2 Computational results on Homberger adapted instances, $n = 200$ and $m = 20$

Instance	Start		RMPD				S-RMPD			
	Obj.	Time	%Im-p	%Im-t	#it	ILP-T	%Im-p	%Im-t	#it	ILP-T
C1_210	4941.23	2.4	1.5	2.3	39	1033.7	**4.2**	**5.3**	223	201.58
C2_210	4038.27	6.3	4.5	4.6	54	749.6	**5.8**	**6.2**	197	81.39
R1_210	5543.26	1.6	3.9	7.0	18	2249.8	**9.7**	**11.1**	229	205.97
R2_210	4743.28	4.4	2.3	3.9	59	673.0	**8.0**	**8.0**	199	90.20
C1_2_1	5481.69	1.5	2.7	5.9	24	1296.7	**8.2**	**8.2**	220	218.02
C1_2_2	5237.93	2.1	1.9	2.6	23	1441.7	**4.3**	**4.9**	227	278.48
C1_2_3	4900.27	3.8	0.1	1.3	12	1724.6	**11.5**	**11.7**	193	365.15
C1_2_4	5373.86	3.9	1.1	3.7	18	1437.1	**6.1**	**6.1**	174	520.33
C1_2_5	5200.77	1.3	0.7	2.7	13	1563.3	**7.0**	**7.7**	208	306.84
C1_2_6	5314.53	2.8	3.6	3.6	8	2437.4	**8.7**	**10.2**	174	542.11
C1_2_7	5173.74	2.5	3.7	5.2	30	1208.7	**6.7**	**7.3**	217	237.13
C1_2_8	5369.94	3.3	3.5	3.8	27	1236.7	**4.3**	**4.3**	217	295.02
C1_2_9	4980.07	2.1	3.0	4.2	28	1239.3	**5.3**	**6.3**	201	311.04
C2_2_1	3900.40	20.6	1.5	2.3	56	729.2	**3.8**	**3.8**	186	124.35
C2_2_2	3771.91	28.0	1.9	2.2	56	701.9	**2.5**	**2.8**	178	127.54
C2_2_3	3907.69	10.1	3.5	**4.7**	50	852.2	**4.4**	4.5	189	144.14
C2_2_4	3984.96	7.0	1.5	1.9	58	644.7	**3.1**	**8.8**	199	84.04
C2_2_5	3962.39	19.8	6.2	7.3	48	821.8	**8.6**	**8.8**	180	108.63
C2_2_6	4003.66	6.2	0.1	0.2	56	675.9	**0.3**	**1.0**	201	95.95
C2_2_7	4199.97	20.8	**5.2**	**5.6**	45	971.6	3.1	5.3	166	224.89
C2_2_8	3906.58	6.1	0.7	0.8	57	696.5	**0.9**	**1.8**	199	113.99
C2_2_9	3550.32	13.3	0.2	0.3	60	611.5	**1.1**	**1.5**	178	82.04
R1_2_1	5195.44	4.5	0.7	0.8	37	1068.4	**0.8**	0.8	229	204.10
R1_2_2	5348.37	2.6	1.0	2.1	27	1229.8	**5.3**	**6.2**	214	252.09
R1_2_3	5244.79	3.2	2.3	3.1	29	1193.0	**5.8**	**6.2**	213	243.21
R1_2_4	5364.92	4.4	1.0	1.5	29	1176.5	**4.3**	**5.0**	213	281.28
R1_2_5	5622.63	3.7	1.3	1.9	16	2786.8	**5.7**	**8.4**	160	657.44
R1_2_6	5395.59	2.4	1.1	5.7	29	2056.0	**7.1**	**7.1**	221	236.79
R1_2_7	5368.91	3.4	0.6	1.4	21	1873.3	**6.7**	**6.7**	198	387.19
R1_2_8	5593.93	2.1	1.0	2.7	13	1800.4	**6.0**	**6.3**	225	224.17
R1_2_9	5337.44	2.9	3.5	4.4	38	1002.1	**5.2**	**5.3**	236	210.37
R2_2_1	4348.15	10.2	4.4	5.0	60	662.6	**5.8**	**5.8**	191	120.45
R2_2_2	4387.83	17.2	4.0	4.8	59	671.8	**4.8**	4.8	175	119.49
R2_2_3	4644.41	8.7	5.6	8.9	52	830.2	**11.4**	**11.5**	169	138.77
R2_2_4	4624.38	8.3	2.2	3.1	56	699.3	**6.1**	**6.6**	182	88.17
R2_2_5	4339.37	19.2	3.6	4.7	59	655.5	**7.4**	**7.6**	170	121.65
R2_2_6	4099.04	9.5	4.1	4.3	62	705.7	**4.5**	**4.6**	167	133.68
R2_2_7	4743.23	6.4	4.5	4.9	58	687.5	**6.4**	**10.1**	189	93.38
R2_2_8	4250.89	4.9	0.2	0.6	62	617.8	**1.0**	**1.0**	187	89.44
R2_2_9	4270.29	11.4	1.1	1.2	60	622.7	**2.1**	**2.4**	193	85.32
RC1_210	5448.18	1.5	0.7	1.4	31	1176.0	**3.0**	**3.2**	215	390.21

Table 2 continued

Instance	Start		RMPD				S-RMPD			
	Obj.	Time	%Im-p	%Im-t	#it	ILP-T	%Im-p	%Im-t	#it	ILP-T
RC2_210	4484.72	7.9	0.0	0.4	60	685.3	**6.2**	**6.2**	182	98.48
RC1_2_1	4750.28	4.1	3.8	5.0	23	1319.3	**6.6**	**7.8**	187	384.68
RC1_2_2	5290.72	2.1	2.6	5.7	11	1609.9	**5.5**	**7.8**	176	528.6
RC1_2_3	5138.55	2.8	0.2	0.8	20	1514.9	**1.9**	**2.8**	193	409.52
RC1_2_4	5121.29	2.7	1.7	2.6	28	1266.1	**5.5**	**6.5**	190	431.23
RC1_2_5	5152.87	6.7	1.8	1.9	7	1804.5	**4.5**	**4.7**	190	472.49
RC1_2_6	4889.95	5.2	1.5	1.7	8	1893.9	**4.1**	**4.7**	143	807.29
RC1_2_7	5260.66	4.7	2.8	2.9	14	1513.5	**5.5**	**7.1**	184	471.36
RC1_2_8	5250.98	1.3	1.8	3.8	33	1142.7	**4.5**	**5.2**	224	270.48
RC1_2_9	5137.59	4.4	0.6	1.3	9	1604.8	**2.9**	**3.2**	181	496.00
RC2_2_1	4151.95	7.7	0.0	0.0	58	647.2	0.0	0.0	216	61.99
RC2_2_2	4138.40	14.7	1.4	1.7	62	681.0	**1.9**	**2.5**	182	114.17
RC2_2_3	4197.74	17.8	1.7	1.9	55	753.5	**3.8**	**4.3**	169	121.17
RC2_2_4	4314.48	3.9	2.2	2.2	57	696.5	2.2	2.2	210	99.01
RC2_2_5	4622.44	9.7	6.0	6.2	64	643.3	**8.6**	**10.2**	180	138.91
RC2_2_6	4182.00	19.2	0.9	1.0	61	669.9	**2.2**	**3.8**	191	115.55
RC2_2_7	4360.36	10.1	4.2	4.6	57	715.3	**6.5**	**8.4**	192	110.76
RC2_2_8	4391.38	6.9	3.6	3.6	60	635.2	**3.8**	**6.5**	203	102.64
RC2_2_9	4264.23	10.0	0.9	1.0	59	682.3	**3.3**	**3.5**	173	162.17

Table 3 Computational results on Homberger instances, $n = 400$ and $m = 10$

Instance	Start		RMPD				S-RMPD			
	Obj.	Time	%Im-p	%Im-t	#it	ILP-T	%Im-p	%Im-t	#it	ILP-T
C1_4_1	7543.64	382.0	0.2	0.7	10	8199.8	**4.0**	**6.8**	155	1818.75
C1_4_2	8244.04	283.7	0.9	2.6	10	8908.5	**10.4**	**15.7**	128	2848.77
C2_4_1	5426.59	730.4	1.1	**6.5**	42	3685.3	**4.9**	5.3	159	528.42
C2_4_2	5860.51	806.5	0.5	3.1	47	2963.3	**3.4**	**5.0**	158	373.25
R1_4_1	8719.48	246.0	0.4	1.4	8	7828.9	**4.3**	**8.6**	151	2160.08
R1_4_2	8512.00	260.6	0.0	0.1	7	7906.4	**8.9**	**11.1**	167	1856.76
R2_4_1	6249.83	639.1	0.3	2.0	56	2595.3	**0.4**	**4.2**	143	692.97
R2_4_2	6450.44	448.6	1.3	4.1	48	3287.6	**4.1**	**7.5**	166	498.95
RC1_4_1	7879.39	233.2	0.2	0.2	6	8618.8	**5.4**	**7.4**	187	1170.97
RC1_4_2	8570.19	211.8	0.2	0.9	13	7234.1	**7.9**	**13.6**	155	2020.04
RC2_4_1	6298.29	2586.8	1.2	4.8	35	2892.5	**4.0**	**5.1**	115	443.90
RC2_4_2	7111.61	132.1	0.6	2.0	52	2706.3	**7.0**	**14.1**	184	478.13

In Table 4 we report the comparison between S-RMPD and S-RMPD(10,3) for the instances in Set 1. Recall that a time limit of 1800 s is imposed to instances having less than 400 vertices, and 9000 s for the remaining three instances. Therefore, S-RMPD(10,3)

Table 4 Computational results on VRPLIB instances

Instance	n	m	S-RMPD				S-RMPD(10,3)			
			Init. Obj.	Init. Time	Obj-t	#it	Init. Obj.	Avg. Time	Obj-t	#it
D022-04g	20	4	485.70	0.0	471.72	70686	499.74	0.00	471.72	26264
D023-03g	22	3	815.57	0.0	815.57	63221	802.36	0.00	**784.10**	21473
D030-03g	28	3	687.97	0.0	665.81	31461	690.05	0.00	665.81	12314
D033-04g	32	4	949.06	0.0	922.65	18962	905.39	0.00	**905.39**	7638
D051-06c	50	6	830.42	0.0	755.21	5744	858.74	0.01	**730.32**	2057
D076-11c	74	11	1269.05	0.1	1235.60	2137	1292.87	0.02	**1185.00**	776
D101-09c	100	9	1322.06	0.3	1244.21	885	1265.54	0.09	**1219.69**	328
D101-11c	100	11	1720.02	0.1	**1504.02**	887	1740.39	0.07	1541.60	296
D121-11c	120	11	1795.78	1.6	1769.00	497	1922.33	0.23	**1628.82**	180
D151-14b	150	14	1826.37	2.7	1822.42	388	1941.17	0.28	**1738.38**	137
D151-14c	150	14	1702.51	1.7	1665.24	388	1767.09	0.29	**1657.73**	145
D200-18b	198	18	2344.25	3.6	**2171.68**	219	2460.54	0.65	2253.92	65
D200-18c	198	18	2364.66	4.1	**2157.92**	225	2495.06	0.69	2210.53	75
D201-05k	200	5	10916.14	27.4	10043.28	219	10376.77	4.22	**9628.85**	80
D241-10k	240	10	8926.83	41.9	**8021.72**	96	9081.84	3.67	8526.54	39
D281-08k	280	8	14354.97	56.5	13773.46	86	13995.92	13.34	**13498.59**	27
D321-10k	320	10	13967.40	130.7	13718.56	71	14069.67	13.96	**13246.27**	25
D361-09k	360	9	17681.01	298.7	16768.70	33	17736.52	40.39	**16623.80**	11
D401-10k	400	10	18759.52	1205.1	17209.13	11	18166.54	56.29	**16714.09**	39
D441-11K	440	11	19970.15	879.0	**18712.93**	7	20319.90	63.75	18731.33	32
D481-12k	480	12	25212.01	1796.0	**22394.35**	5	25208.57	113.25	22832.23	27

considers 600 s for each initial solution in the first case, and 3000 s in the latter. The main message of this table is that the approach is sensitive to the initial solution. The results obtained by S-RMPD(10,3) in general are better than the original results. In this particular table, given an instance we identify the best initial solution between the two methods by underlining its objective value. We can observe that in this case there is no clear tendency indicating that a better initial solution would lead to a better final solution. In this sense, the results are aligned with the ones reported in De Franceschi et al. (2006), being able to achieve a considerable improvement when starting from solutions of different quality.

For the three instances having 400 vertices or more, the S-RMPD(10,3) produced better results. This is justified by the fact that each initial solution of the S-RMPD(10,3) consumes the whole time assigned for the execution, and further improved solutions could be found if more time is available. On the other hand, S-RMPD is able to find further improvements by using the available time with a unique starting solution. Although we do not report the details, a similar behavior is observed on the instances in Set 2. S-RMPD(10,3) produces slightly worse results than S-RMPD, around 1% on average for $n = 200$ and 2% for $n = 400$, but in all cases we observe that S-RMPD(10,3) is still improving the incumbent solution when approaching to the time limit imposed.

Based on this analysis, we report in Table 5 the results obtained on instances in Set 3. The objective is to analyze the quality of the solutions obtained by the S-RMPD(10,3). We report the results obtained on these instances by Dumitrescu et al. (2008). Optimality gaps

Table 5 Computational results on $m = 1$ instances

Instance	n	S-RMPD				S-RMPD(10,3)			
		Best	Best LB	Time	Opt	Start	Obj-t	Time	%Im-t
prob15a	30	5150	–	8	✓	5309	5286	0.00	2.64
prob15b	30	5391	–	21	✓	5769	5457	0.83	1.22
prob15c	30	5008	–	0	✓	5008	–	–	0.00
prob15d	30	5566	–	14	✓	6034	5566	4.07	0.00
prob15e	30	5229	–	0	✓	5811	5229	1.15	0.00
prob20a	40	5698	–	12	✓	6116	5698	1.65	0.00
prob20b	40	6213	–	20	✓	6241	6213	0.43	0.00
prob20c	40	6200	–	19	✓	7043	6200	2.12	0.00
prob20d	40	6106	–	17	✓	6830	6243	10.97	2.24
prob20e	40	6465	–	58	✓	6551	6465	3.01	0.00
prob25a	50	7332	7168.14	14400	2.29	7781	7386	64.98	0.74
prob25b	50	6665	–	3138	✓	7346	6956	17.76	4.37
prob25c	50	7095	–	291	✓	8153	7347	33.29	3.55
prob25d	50	7069	–	14323	✓	7712	7405	56.69	4.75
prob25e	50	6754	–	72	✓	7740	7058	7.39	4.50
prob30a	60	7309	7196.27	14400	1.57	7900	7412	21.11	1.41
prob30b	60	6857	–	2843	✓	8087	7116	125.56	3.78
prob30c	60	7723	–	1891	✓	9019	8007	26.91	3.68
prob30d	60	7310	–	573	✓	7947	7310	28.30	0.00
prob30e	60	7213	7166.34	14400	0.65	7746	7294	514.33	1.12
prob35a	70	7746	–	2104	✓	9091	8026	149.52	3.61
prob35b	70	7904	7496.03	14400	5.44	8956	8595	234.67	8.74
prob35c	70	7949	7858.39	14400	1.15	8777	8151	478.97	2.54
prob35d	70	7905	7686.77	14400	2.84	10006	8386	82.90	6.08
prob35e	70	8530	8069.74	14400	5.70	10057	9070	69.48	6.33

are computed with respect to the best solution reported by them. For instances having 10 and 20 vertices, the initial heuristic is able to find the optimal solution and therefore are omitted. For the remaining instances, the average optimality gap is 2.55%, obtaining optimal or near-optimal solutions in many cases. We can observe gaps below 1% for instances with 30 and 40 vertices, and below 4% for instances having 50 and 60 vertices. We can also observe that the time required to find the best solution is considerably below the time limit imposed for each initial solution in most of the cases. We remark that both S-RMPD and consequently S-RMPD(10,3) are tunned considering a multi-vehicle context. Indeed, some of the algorithmic decisions are taken assuming multiple routes, such as the node removal procedure. Despite that inter-route and capacity restrictions in the S-RMPD(10,3) are not binding, removing requests (i.e., a pair pickup and delivery) are not necessary for the single vehicle route. These kind of decisions may clearly affect the overall performance of the approach, for which further investigations could be conducted.

We extend the results to the instances in Set 4. Since the VRPPD represents a relaxation of the problem studied in Li and Lim (2003), Ropke and Pisinger (2006), we compare

the improvements obtained with respect to the solutions reported for the PDVRPTW. The aggregated results by type of instances are shown in Table 6. For each group of instances, we report the average objective value of the Best Known Solution (BKS) and the average of vehicles used for the PDVRPTW. In addition, we report the results obtained by S-RMPD(10,3) in terms of the average objective value of the starting solution that produced the best solution, as well as the relative improvement percentage with respect to the BKS. We further include in our experiments a variation of S-RMPD, named S-RMPD(BKS),

Table 6 Computational results on Li and Lim (2003) instances

Instance	Avg. BKS	Avg. m	S-RMPD(10,3)		S-RMPD(BKS)
			Avg. Init	%Im-t	%Im-t
LC1	874.12	9.67	796.32	9.61	9.15
LC2	589.86	3.00	667.74	−4.83	0.44
LR1	1219.62	11.92	1051.48	15.68	15.40
LR2	970.84	2.73	844.73	18.03	14.94
LRC1	1386.74	11.50	1132.19	21.46	20.19
LRC2	1133.12	3.25	909.75	21.99	16.06

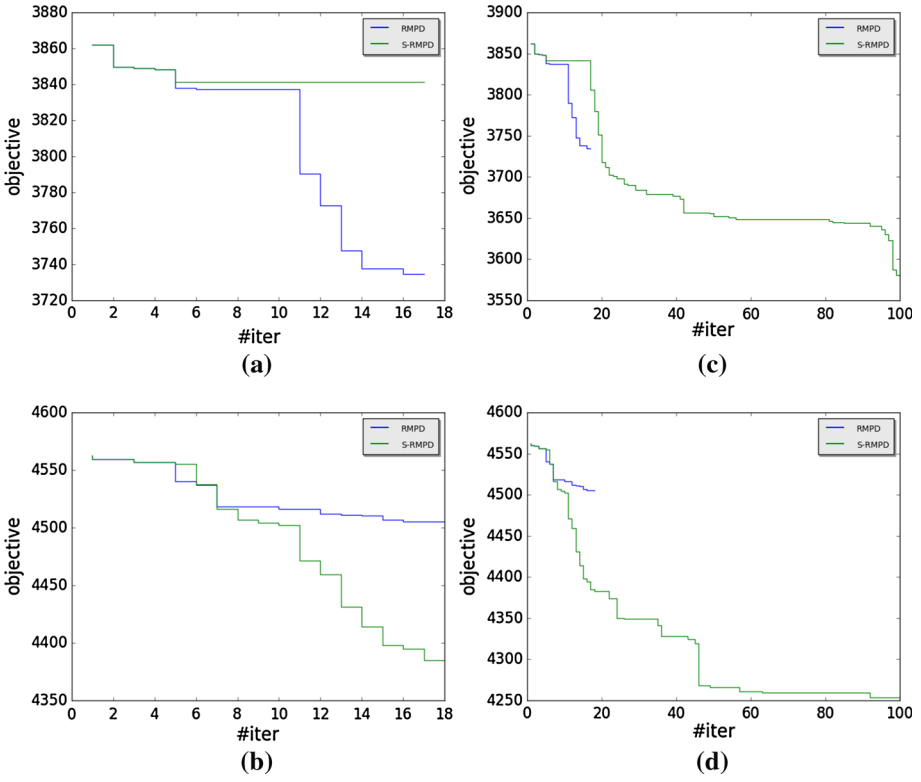

Fig. 2 Objective values versus number of iterations for two instances **a** instance C2_2_2, initial iterations **b** instance R2_2_2, initial iterations **c** instance C2_2_2, first 100 it **d** instance R2_2_2, first 100 it

where the BKS is set as the unique initial solution. As expected, since we are considering a relaxation of the problem, the improvements obtained are significant. There is a particular behavior for LC2 type instances, where S-RMPD(10,3) is not able to improve the solution. This is related to the quality of the initial solutions, and the results show that when starting from the BKS we are able to improve these solutions. Another interesting observation regards the quality of the solutions depending on the instance type. The improvements obtained are smaller for the clustered instances LC1 and LC2 compared to the rest of the instances.

Finally, we make some remarks regarding the impact of the number of iterations for both RMPD and S-RMPD. If instead of limiting the execution time, we impose a limit on the number of iterations, in general the results obtained are somehow mixed and do not show a clear tendency. When comparing the evolution of the objective function of the incumbent solution in terms of the number of iterations executed, RMPD finds better solutions than S-RMPD on some instances, and the opposite occurs in other cases. We show in Fig. 2 two representative examples. In Fig. 2a, b we show the first 17 iterations for two different instances showing the two situations. We remark that both methods start from the same initial solution. In addition, in Fig. 2c, d, we show the results for the first 100 iterations. We remark that in these instances, RMPD is not able to reach this number due to the time required in each iteration. These last two images illustrate the advantage of considering formulation S-RMPD as a local search operator.

5 Conclusions and future research

In this paper we consider the Vehicle Routing Problem with Pickups and Deliveries (VRPPD) and propose an adaptation of the Reallocation Model (RM) proposed by De Franceschi et al. (2006), which involves exploring a large neighborhood of a feasible solution by the resolution of an ILP formulation. To account for the precedences among customers, we adapt and redefine some basic notation and propose a first ILP formulation, RMPD, which is then improved by applying strengthening techniques to limit the number of precedence-related constraints in the formulation, S-RMPD. In both cases, we also adapt the formulation in order to account for the problem where exactly m vehicles have to be used. The overall scheme proposed by De Franceschi et al. (2006) is adapted for the VRPPD and the two ILP formulations are evaluated experimentally on a large number of instances having up to 481 customers. The computational results show that S-RMPD outperforms RMPD in almost all instances, and that their behavior in terms of the improvements obtained as well as regarding the computational effort required makes S-RMPD a suitable option to be applied in practice.

As future research, it would be interesting to extend the S-RMPD to the case where the VRPPD includes time windows at the customers as well. Similarly to our case, the inclusion of time windows may require considerable modifications to the overall scheme, both at a modeling and at an experimental level, due to feasibility issues. In terms of the RM, due to the presence of time windows, the assignment of a sequence to an insertion point may become infeasible depending on which sequences are assigned previously in the route, and how this affects the arrival times at its vertices. One alternative could be to control the feasibility of the assignments of sequences to insertion points by adapting the idea of *infeasible paths* (see, e.g., Ascheuer et al. (2000)) within each route, and incorporate them on demand during the optimization of the RM.

Acknowledgements This research is partially supported by Grants PICT-2010-0304, PICT-2011-0817, PICT-2013-2460 and UBACyT 20020100100666. The authors also thank the two anonymous referees and the editor for providing valuable suggestions and comments for improving this paper.

References

Archetti, C., & Speranza, M. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, *2*(4), 223–246.

Ascheuer, N., Fischetti, M., & Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, *36*(2), 69–79.

Ball, M. O. (2011). Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, *16*(1), 21–38.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *Top*, *15*(1), 1–31.

Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, *153*(1), 29–46.

Danna, E., Rothberg, E., & Pape, C. L. (2005). Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, *102*(1), 71–90.

De Franceschi, R., Fischetti, M., & Toth, P. (2006). A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, *105*(2–3), 471–499.

Dumitrescu, I., Ropke, S., Cordeau, J.-F., & Laporte, G. (2008). The traveling salesman problem with pickup and delivery: Polyhedral results and a branch-and-cut algorithm. *Mathematical Programming*, *121*(2), 269–305.

Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming*, *98*(1–3), 23–47.

Golden, B., Raghavan, S., & Wasil, E. (Eds.). (2008). *The vehicle routing problem : Latest advances and new challenges (Vol. 43)*., Operations research/computer science interfaces US Boston: Springer.

Hansen, P., Mladenovic, N., & Urosevic, D. (2006). Variable neighborhood search and local branching. *Computers & Operations Research*, *33*(10), 3034–3045.

Helsgaun, K. (2000). An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, *126*(1), 106–130.

Hernández-Pérez, H., & Salazar-González, J.-J. (2009). The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *European Journal of Operational Research*, *196*(3), 987–995.

Li, H., & Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, *12*(02), 173–186.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, *21*(2), 498–516.

Muter, I., Birbil, I., & Bülbül, K. (2012). Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. *Mathematical Programming*, *142*(1), 47–82.

Muter, I., Birbil, I., Bülbül, K., & Güvenç, (2012). A note on A LP-based heuristic for a time-constrained routing problem. *European Journal of Operational Research*, *221*(2), 306–307.

Naji-Azimi, Z., Salari, M., & Toth, P. (2012). An integer linear programming based heuristic for the capacitated m-Ring-Star problem. *European Journal of Operational Research*, *217*(1), 17–25.

Nanry, W. P., & Barnes, J. W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, *34*(2), 107–121.

Rodríguez-Martín, I., & Salazar-González, J. J. (2012). A hybrid heuristic approach for the multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *Journal of Heuristics*, *18*(6), 849–867.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472.

Salari, M., Toth, P., & Tramontani, A. (2010). An ILP improvement procedure for the open vehicle routing problem. *Computers & Operations Research*, *37*(12), 2106–2120.

Toth, P., & Tramontani, A. (2008). An integer linear programming local search for capacitated vehicle routing problems. *The Vehicle Routing Problem: Latest Advances and New Challenges*, *43*, 275–295.

Toth, P. & Vigo, D. (eds.) (2014). *Vehicle routing: Problems, methods, and applications, 2nd ed*. MOS-SIAM Series on Optimization. Philadelphia, PA, USA.