OR Spectrum (2017) 39:303–319
DOI 10.1007/s00291-016-0454-y

REGULAR ARTICLE

The multi-vehicle profitable pickup and delivery problem

Margaretha Gansterer[1] · **Murat Küçüktepe**[1] · **Richard F. Hartl**[1]

Received: 27 April 2016 / Accepted: 21 June 2016 / Published online: 28 June 2016

Abstract The transportation industry expanded rapidly in a highly competitive environment. Logistics companies with insufficient volume of transport capacities are forced to make a selection of customers that they can integrate efficiently into their tours. This is of particular relevance in the pickup and delivery market, where shipments from several different customers can be moved on the same vehicle. In the literature, however, the problem of customer selection has not been applied for the given class of pickup and delivery problems so far. We want to fill this gap by introducing the multi-vehicle profitable pickup and delivery problem (MVPPDP), where multiple carriers transport goods from a selection of pickup customers to the corresponding delivery customers within given travel time limits. For this problem, we propose a method based on general variable neighborhood search (GVNS). We conduct experiments with two different variants of this method, namely a sequential (GVNSseq) and a self-adaptive (GVNSsa) version. Additionally, we compare it to an algorithm based on Guided Local Search (GLS), which is known to find good solutions for related problems very fast. The performance of these methods is examined on the basis of data instances with up to 1000 customer requests. In an experimental study, we observe that both variants of GVNS with 11 neighborhoods outperform GLS with regard to solution quality for all sizes of test instances. However, for medium sized and large instances, GLS shows an advantage in average runtimes.

Keywords Pickup and delivery · Profitable tour problem · Metaheuristics

✉ Margaretha Gansterer
margaretha.gansterer@univie.ac.at

Murat Küçüktepe
muratkucuktepe@gmail.com

[1] Department for Business Administration, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

1 Introduction

In the past decades, intensification of competition on global markets together with heightened customer expectations has led to an increased pricing pressure which negatively affects logistic providers' profit margins (Ruijgrok 2001). Companies need to achieve a maximum level of efficiency to stay in business. To reach this goal, they can, for instance, participate in collaborative networks where transportation requests are traded among competitors.

Thus, research has shifted its focus from strengthening internal to external relations along the supply chain (Skjoett-Larsen 2000; Ergun et al. 2007a, b). Collaborations in freight logistics have been extensively studied in the past years (e.g., Ackermann et al. 2011; Dahl and Derigs 2011; Krajewska and Kopfer 2006; Krajewska et al. 2008; Sheffi 2004; Puettmann and Stadtler 2010). For the carriers participating in such networks, it is of particular importance to (i) make a good selection of customer requests that should be served with their own fleet, (ii) assign selected requests to vehicles, and (iii) visit them in most efficient tours. This real world issue has been addressed in the literature of combinatorial optimization by the huge subclass of vehicle routing problems (VRPs) that consider the possibility of visiting only a subset of customers, associate a revenue with each of them and aim at maximizing the total profit.

This real world problem is, for instance, addressed by the well-known team orienteering problem (TOP), proposed by Chao et al. (1996b), or multiple tour maximum collection problem (Butt and Cavalier 1994). Both of them belong to the class of vehicle routing problems with profits. A classification of traveling salesman problems with profits is given in Feillet et al. (2005). While in the TOP, the objective is to maximize the total collected profit, other problems, like the prize-collecting VRP, minimize the total traveling cost (Feillet et al. 2005; Tang and Wang 2006; Archetti et al. 2014). If a carrier can either visit a customer with one of his vehicles or assign the customer to a common carrier, the problem is referred to as VRP with private fleet and common carrier which is investigated by, e.g., Chu (2005), Bolduc et al. (2008), Stenger et al. (2013). If the objective combines both minimization of travel time and maximization of collected profit, the problem is commonly referred to as the Profitable Tour Problem (PTP), addressed by, e.g., Archetti et al. (2009). In Chbichib and Chabchoub (2012), the problem is denominated as Profitable VRP with Multiple Trips.

In a recent book, Archetti et al. (2014) claim that an important application of VRP with profits arises in the context of the small packaging industry. Large companies in this branch outsource last-mile deliveries of unprofitable areas to subcontractors . However, from a practical point of view, the described collaborative setting among carriers seems to be most relevant in a pickup and delivery environment. Given amounts of goods have to be transported from pickup customers to corresponding delivery customers. Shipments from different customers can be moved on the same vehicle. This gives the carriers flexibility to select the most profitable set of customer requests and to outsource or to offer remaining ones to competitors. A PDP in the collaborative context has been recently addressed by Li et al. (2016). However, the authors assume that each request has two time windows and that carriers have reserved requests. This limits the number of requests that can be offered to other carriers.

To the best of our knowledge, no solution methods have been proposed for the general problem so far. Thus, in this study we propose and formally define the multi-vehicle profitable pickup and delivery problem (MVPPDP), where we assume carriers to serve less than truckload paired pickup and delivery requests, i.e., each request is associated with a prespecified origin and destination. Parragh et al. (2008) refer to this problem as the VRP with pickups and deliveries (VRPPD). It is classified by Berbeglia et al. (2007) as one-to-one PDP. Savelsbergh and Sol (1995) refers to this problem as the general PDP (GPDP). We solve this problem using general variable neighborhood search (GVNS), since variable neighborhood search (VNS) based methods show very good results for related problems (e.g., Parragh et al. 2010; Stenger et al. 2013). We develop two variants of GVNS and evaluate them based on newly created instances with up to 1000 customer requests covering different scenarios. We compare our algorithm to a guided local search (GLS)-based method, which has been proven to find good solutions for a related problem, namely the TOP, in very short time (Souffriau et al. 2009a). Based on our experimental study, we show that on average GVNS yields higher solution quality for all types of test instances, while GLS exhibits an advantage in average runtimes.

The remainder of this paper is structured as follows. Section 2 gives a literature review. The problem formulation is given in Sect. 3. We present the solution methods in Sect. 4. The computational study is described in Sect. 5. Conclusions and further research are summed up in Sect. 6.

2 Literature review

The TOP is known to be NP-hard (Laporte and Martello 1990; Boussier et al. 2007). Thus, several heuristics and metaheuristics for this problem class have been proposed in the literature. An LP-based granular VNS is presented by Labadie et al. (2012). A simulated annealing heuristic is applied by Lin and Yu (2012) and Lin (2013), and tabu search by Tang and Miller-Hooks (2005). Two variants of a generalized tabu search algorithm and a variable neighborhood search algorithm are proposed by Archetti et al. (2007). Ant-colony optimization is used by Ke et al. (2008), and a Particle Swarm Optimization-based Memetic Algorithm by Dang et al. (2011). An iterative framework incorporating three components is developed by Qian and Andrew (2014). The first two components are a local search procedure and a simulated annealing procedure, while the third component recombines routes to identify high quality results. Finally, metaheuristics based on GLS (Souffriau et al. 2009a), Iterated Local Search (Souffriau et al. 2009b) and path relinking (Souffriau et al. 2010) find very good solutions for this problem class in relatively short amount of time.

Much fewer studies can be found on the PTP (Archetti et al. 2014). An approximation algorithm for the asymmetric PTP is presented by Nguyen and Nguyen (2010). Large neighborhoods for TOP as well as for PTP are studied by Vidal et al. (2016). They have been tested in a local-improvement method, an iterated local search, and a hybrid genetic search. Archetti et al. (2009, 2013) propose heuristic and exact procedures for the capacitated TOP and PTP. A branch-and-cut algorithm for the PTP is presented by Jepsen et al. (2014). A rich variant of the PTP is studied by Lahyani

et al. (2013). The authors propose a VNS search algorithm embedded with an adaptive large neighborhood search.

Detailed surveys on PDP are provided by Parragh et al. (2008) and Berbeglia et al. (2007). Various studies on this problem class have been published since then. A heuristic for the PDP with split loads is developed by Nowak et al. (2008). The dynamic PDP is surveyed by Berbeglia et al. (2010). Dynamic programming solutions for a multi-commodity, capacitated PDP are explored by Psaraftis (2011). A VNS approach for solving the one-commodity PDP is presented by Mladenović et al. (2012). Mathematical models and a branch-and-cut-and-price algorithm for the PDP with shuttle routes have been recently proposed by Masson et al. (2014). The PDP with transshipment is addressed by Rais et al. (2014).

Hartl and Romauch (2013) present an extension of the one-commodity pickup and delivery traveling salesman and the pickup and delivery VRP including aspects of an orienteering problem. Note that the authors do not consider the classical PDP (Parragh et al. 2008). They investigate unpaired service requests for a homogenous product, which is also known as the many-to-many problem (Berbeglia et al. 2007). The Static Bike Redistribution Problem, which also combines the orienteering problem with the one-commodity PDP, is investigated by, e.g., Rainer-Harbach et al. (2013) and Raviv et al. (2013). Also the selective PDP (SPDP) by Ting and Liao (2013.) and the single vehicle routing problem with deliveries and selective pickups (SVRPPD) by Gribkovskaia et al. (2008) refer to this problem class. The authors find a route for a vehicle to visit some pickup nodes and supply the demand of all delivery nodes. Privé et al. (2006) investigate a problem arising in the soft drink distribution, where all delivered goods originate from a unique distribution center and all collected goods end up at a unique destination as well. A related problem for maritime transportation is presented by Hemmati et al. (2014). They consider optional spot cargoes and solve it as routing and scheduling problem.

The PTP studied in Ko et al. (2010) differs from the definition commonly used by researchers in transportation science, which is, for instance, described in Feillet et al. (2005). The authors aim at finding a tour among all available pickup and delivery points, while the incremental profit is to be maximized. Profits are to be maximized by adjusting pickup and delivery times at service centers. Hence, the aspect of selecting a profitable subset of requests is not considered.

In the following section, we present the formal problem description of the MVPPDP.

3 Problem formulation

The MVPPDP models the static problem with a central depot and a homogenous fleet of vehicles. We assume carriers to serve less-than-truckload (LTL) paired pickup and delivery requests, i.e., each request is associated with a prespecified origin and destination. Hence, delivery customers have to be visited after the paired pickup customer has been served. As long as the capacity and time constraints are not violated, many consecutive pickup customers or many consecutive delivery customers might be visited. The objective is to maximize the total collected revenues minus the total travel cost.

The following notation is used for the mathematical model (based on Ropke and Cordeau 2009 and Parragh et al. 2008):

- n number of pickup vertices and number of delivery vertices
- m number of vehicles
- P set of pickup vertices, $P = \{1, .., n\}$
- D set of delivery vertices, $D = \{n + 1, \ldots, 2n\}$
- V set of all vertices including start depot 0 and end depot $2n + 1$, $V = P \cup D \cup \{0, 2n + 1\}$
- K set of available vehicles, $K = \{1, \ldots, m\}$
- r_i revenue to be gained when visiting delivery/pickup vertex i
- q_i supply (pickup, $q_i > 0$) or demand (delivery, $q_{n+i} = -q_i$) at vertex i. At start depot 0 and at the end depot $2n + 1$, there is no supply or demand ($q_0 = 0$, $q_{2n+1} = 0$)
- d_i service duration at vertex i
- c_{ij} transportation cost when traveling from i to j
- t_{ij} travel time between vertex i and vertex j
- C loading capacity of a vehicle
- T maximum tour time of a vehicle
- x_{ijk} binary decision variable equal to one if and only if arc ij is used by vehicle k
- Q_{ik} decision variable giving the loading amount of vehicle k after visiting vertex i
- B_{ik} decision variable for the beginning of service time of vehicle k at vertex i

The mathematical model can be formulated as follows (based on Ropke and Cordeau 2009 and Parragh et al. 2008):

$$\text{maximize} \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} (r_i - c_{ij}) x_{ijk} \tag{1}$$

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} \leq 1 \qquad j \in V \tag{2}$$

$$\sum_{j \in V} \sum_{k \in K} x_{ijk} \leq 1 \qquad i \in V \tag{3}$$

$$x_{i0k} = 0 \qquad i \in V, k \in K \tag{4}$$

$$x_{2n+1,jk} = 0 \qquad j \in V, k \in K \tag{5}$$

$$\sum_{i \in V} (x_{ijk} - x_{jik}) = 0 \qquad j \in V \setminus \{0, 2n + 1\}, k \in K \tag{6}$$

$$\sum_{j \in V} (x_{ijk} - x_{n+i,jk}) = 0 \qquad i \in P, k \in K \tag{7}$$

$$\sum_{j \in V} x_{0jk} = \sum_{i \in V} x_{i,2n+1,k} = 1 \qquad k \in K \tag{8}$$

$$(x_{ijk} = 1) \Rightarrow Q_{jk} = Q_{ik} + q_j \qquad i \in V, j \in V \setminus \{0\}, k \in K \tag{9}$$

$$Q_{ik} \leq C \qquad i \in V, k \in K \tag{10}$$

$$Q_{0k} = 0 \qquad k \in K \tag{11}$$

$$B_{ik} \leq B_{n+i,k} \qquad i \in P, k \in K \tag{12}$$

$$B_{jk} \geq x_{ijk}(B_{ik} + d_i + t_{ij}) \qquad i \in V, j \in V, k \in K \tag{13}$$

$$B_{2n+1,k} \leq T \qquad k \in K \tag{14}$$

$$B_{0k} = 0 \qquad k \in K \tag{15}$$

$$x_{ijk} \in 0, 1 \qquad i \in V, j \in V, k \in K \tag{16}$$

$$Q_{ik}, B_{ik} \geq 0 \qquad i \in V, k \in K. \tag{17}$$

The objective function (1) maximizes the total profit by subtracting the total travel costs from the revenues collected.Constraints (2) and (3) ensure that each vertex is visited at most once. The origin depot 0 cannot be entered and the destination depot $2n+1$ cannot be left by any vehicle k. This is defined by (4) and (5). Flow conservation is ensured by (6). Pickup and delivery pairs have to be visited by the same vehicle, as stated by (7). Vehicles start from the depot and return back to the depot (8). Constraints (9) are used to control the vehicles load. This can easily be reformulated as a linear constraint by means of the usual big M formulation. The load of a vehicle is bounded to C by (10). Vehicles start empty (11). Constraints (12) state that each pickup node has to be visited before the corresponding delivery node. The earliest beginning of service at vertex j is given by the beginning of service at vertex i plus the service time at i and the travel time between i and j. This is formulated by (13). The total tour length of each vehicle is restricted by (14). Each vehicle starts at the depot at time 0 (15). Finally, the binary property and nonnegativity of decision variables are defined by (16) and (17).

4 Metaheuristics

The VRP with Pickup and Delivery (VRPPD) and the PTP have been shown to be NP-hard by Toth and Vigo (2002) and Feillet et al. (2005), respectively. Thus, also the MVPPDP belongs to the class of NP-hard problems. To find good solutions in reasonable amount of time, we develop an algorithmic framework based on GVNS (Hansen et al. 2008; Mladenović et al. 2012). GVNS is a modified extension of the well-known VNS (Mladenović and Hansen 1997). Unlike the VNS, it uses more than one neighborhood in a local search. This search strategy is called Variable Neighborhood Descent (VND). We compare the performance of our framework to the GLS-based metaheuristic presented by Souffriau et al. (2009a), which has been proven to find good solutions for the TOP in very short computing time.

4.1 Construction and neighborhood operators

In the following, we describe the construction heuristic as well as the neighborhood operators that we use for both solution methods, i.e., GLS and GVNS.

Construction The greedy construction heuristic is based on cheapest insertion. First, m seed nodes, i.e., requests where the pickup and the delivery point are farthest away from the depot, are determined. For filling up tours, we calculate an insertion ratio for each request. This ratio divides the request's revenue by the insertion cost. The latter is the sum of additional travel time caused by inserting the pickup and then the delivery point in the best possible, i.e., cheapest, position in one of the tours. The request with the highest insertion ratio is included in the best tour. This procedure is repeated until no further requests can be inserted in any tour.

Insert This operator inserts requests, that are not included in a tour yet. The position where they consume the least travel time has to be found. First, the pickup node is considered and then the best position for the delivery node is determined. Also the maximum tour length has to be taken into account. Requests are inserted in a sequence which is based on their maximum appropriateness for any tour, which is calculated based on the center of gravity of each tour (Souffriau et al. 2009a). For the PDP we assume that a request's appropriateness is the sum of those of the pickup and the delivery nodes associated with it.

Move Requests from one tour are excluded and then included into other tours. This comes from observations indicating that it is more favorable to have a lot of time available in one tour and almost no time budget left in other tours (Souffriau et al. 2009a). In each tour, it is checked, whether moving a request to another tour improves the solution. Starting with the request 1 in tour 1, the first removal that improves the solution is performed. For finding the best position for including a request, the *insert* operator is used.

2-opt This operator is based on the well-known 2-opt mechanism invented by Croes (1958). Two edges in a tour are removed and the nodes are then reconnected in the best possible way. Since we are dealing with pickup and delivery requests, we have to test feasibility with regard to precedence constraints after modifying a tour. Clearly, the probability of finding feasible solutions after changing sequences of nodes decreases with the length of the sequence. To keep the computational effort on a low level, we only consider reconnections where two nodes, that are already linked by a direct arc, are exchanged.

Replace Included requests are replaced by non-included requests if this replacement yields a higher objective value. Again the sequence of considering requests to be included is based on their appropriateness and the *insert* operator is used to find the best position within a tour. If, by including a request, the maximum tour length is exceeded, all included locations with a lower revenue are considered for exclusion.

Swap This inter-tour operator swaps two requests that are included in different tours if the objective value is increased and tour length restrictions are not violated. The best position for including a request is again determined by the *insert* operator.

Pairwise forward exchange This operator tries to find a better solution by swapping each node with each of the other nodes appearing later in the tour. Of course a swap is only possible if it does not violate precedence- or capacity restrictions.

Pairwise backward exchange This is similar to *Pairwise forward exchange*, but its swapping starts with the last node in the tour and moves backward.

Relocate pairs Consecutive pairs within a tour are relocated within a tour. Swapping consecutively ordered customer pairs, i.e., a pickup followed by a delivery node,

instead of single nodes reduces the CPU time, since the precedence feasibility needs not be tested. For each customer pair, we check whether it can be relocated by exchanging its pickup and the delivery positions with the pickup and delivery positions of any other customer pair.

Forward insertion This intra-tour operator removes a node which can be a pickup or a delivery customer and inserts it to a forward position.

Backward insertion Similar to *Forward insertion*, but the operator starts by removing nodes from the end of a tour and tries to move them to a backward position.

Gravity center exchange A gravity center for a tour is calculated using the included locations weighted by their revenues. The operator removes the request which is farthest from the gravity center and inserts non-included requests as long as the capacity constraints are met.

Note that we provide a portfolio of intra- and inter-tour operators. While the former manipulate existing tours by trying to minimize travel times, the latter also consider requests that are not included in tours yet.

4.2 GLS-based approach

GLS has been introduced by Voudouris and Tsang (1996) and has been successfully applied to a wide range of combinatorial optimization problems ever since. The idea is to escape from local minima by augmenting or diminishing the objective function of a problem with a set of penalty terms which are dynamically manipulated during the search process. Our algorithm follows the procedure as it is proposed by Souffriau et al. (2009a). However, the local search operators are adapted for the PDP, as it is described above.

After a construction phase (see Sect. 4.1), a subset of the neighborhood operators that we describe above is implemented in a given sequence for a maximum number of iterations. A global loop calls the local search phase at most n times and also destroys parts of the solution after $\frac{n}{2}$ iterations, or if the local search operators do not find a new best solution for the first time. The local search phase is composed of the following operators. Note that *Replace* and *2-opt* are adapted following the GLS paradigm:

1. *Insert*
2. *GLS Replace* Following the GLS paradigm, in each iteration the included request with the lowest revenue and the non-included request with the highest revenue are penalized and rewarded, respectively. This increases the probability that these requests are excluded or included in the next iteration and thus helps to leave local minima. Since pickup and delivery nodes must be in- or excluded together, rewards and penalties are always assigned to both of them. The magnitudes of rewards and penalties depend on the average revenues of all requests. Also the number of times a request has been rewarded or penalized is taken into account (see Souffriau et al. 2009a).
3. *Swap*
4. *Move*
5. *GLS 2-opt* Once a local minimum is reached, all edges of a solution are penalized by incrementing their distance values. Also the number of times this edge has

already been penalized is taken into account. The magnitude of the penalty depends on the average distance to other locations (Souffriau et al. 2009a). The penalization is then influencing the probability that a certain arcs gets removed from a tour.

Diversification strategies are used to explore the solution space extensively. In the following, we distinguish between *global shaking* and *local search shaking*. The first one is done only once after half of the total number of iterations is reached. The solution is destroyed by removing 75 % of the nodes from the beginning of each tour. *Local search shaking* is applied if operator *GLS Replace* does not find a better solution after 100 iterations. In this case, current tours are destroyed by removing the request having the lowest revenue plus nodes being located closest to the origin and destination of this request. Since requests are considered as inseparable couples, if a pickup node is removed, the respective delivery point has to be removed as well, and vice versa. By this, we remove 30 % of the current solution. To increase diversification, this procedure has been randomized. The least profitable request is selected with a probability of 70 %, but with probability of 20 and 10 %, the second and the third worst request is removed, respectively.

4.3 GVNS

The GVNS basically consists of three main parts: (i) construction, (ii) shaking, and (iii) VND. The construction heuristic is already described above (see Sect. 4.1). In the shaking step, there are three versions which are randomly chosen in each iteration. For the VND, two types, namely, a sequential and a self-adaptive one are proposed. For both of them, all 11 neighborhoods from the above are used.

4.4 Shaking

The GVNS investigates the solution space by changing neighborhoods in a deterministic manner. Therefore, we propose a stochastic shaking process to diversify widely over the solution space. Extensive tests with various shaking methods and different parameter settings have been tested. The best results were found with the following shaking phase. In each iteration, one of the procedures (S1, S2, S3) is selected randomly.

S1 A randomly chosen single request is deleted from one of the tours. Afterwards, the tour is filled up using *cheapest insertion* excluding the deleted node.

S2 This shaking procedure deletes a percentage of requests of a tour. The percentage, the tour, and the requests are selected randomly. The percentage is chosen such that at least 10 % and at most 40 % of a tour are destroyed. Again, the tour is filled up using *cheapest insertion* but also recently removed requests can be included.

S3 This is an extension to S2. Again 10–40 % are destroyed from a randomly chosen tour. Requests are not selected randomly but based on a removal ratio. The latter divides the request's revenue by the time saving that is yield if the request is dropped from the tour. Requests are deleted from a tour according to increasing removal ratio.

4.5 VND

This part of GVNS addresses the sequence and intensity of neighborhood exploration. VND is a heuristic that applies different neighborhood operators to an initial solution to improve it. In this research, it is used as local search heuristic after the perturbation of the initial solution. There are several VND search strategies presented in the literature, e.g., sequential VND, nested VND, mixed VND. While the cardinality of sequential VND is equal to the sum of cardinalities of all its components, the number of solutions visited by nested-VND is equal to the product of the cardinalities of all its components (Ilić et al. 2010). To keep the computational effort on a reasonable level, we propose two relatively fast VND search strategies: (i) sequential VND and (ii) self-adaptive VND as it is presented by (Hu and Raidl 2006).

4.5.1 Sequential VND

This search strategy explores the neighborhoods one by one in a sequential and deterministic manner. The final solution has to be the local minimum of all neighborhood structures. The procedure starts with a given neighborhood and explores the solution space. As long as an improvement is found, it resumes to search within the same neighborhood. If there is no improvement, it leaps to the second neighborhood. Afterwards, if there is no improvement, other unused neighborhoods are explored. In case of an improvement, it starts from the first neighborhood again. The process terminates if a local minimum (or maximum) for all neighborhoods is detected. Clearly, the ordering of neighborhoods affects the solution quality. To reduce computational effort, it seems reasonable to order the neighborhoods by CPU time. This is due to the fact that the first neighborhoods are used more often and if time-consuming neighborhoods are applied first, the total CPU time increases substantially. Extensive tests with regard to good orderings have been performed. We propose the following sequence of neighborhoods: *Insert*, *Gravity Center Exchange*, *Move*, *Replace*, *Backward insertion*, *Forward insertion*, *2-opt*, *Pairwise backward exchange*, *Pairwise forward exchange*, *Relocate pairs*, *Swap*. To evaluate the effectiveness of these neighborhoods, some additional tests have been conducted. More precisely, we generate comparative values by running all test instances while excluding the neighborhoods one by one. By this, we verify that each neighborhood has a positive impact on the final solution and should thus be included in the list.

4.5.2 Self-adaptive VND

To control the neighborhoods in an effective way, Hu and Raidl (2006) present a self-adaptive VND which adjusts itself to the entire process. It applies the solution-improving neighborhoods more often by changing the order of them through the execution. An initial order of neighborhoods is chosen randomly or in an intuitive way. Each neighborhood is associated with a rating value. In the local search process, after a neighborhood of an initial solution is explored, the rating belonging to this neighborhood is updated. If that specific neighborhood does not improve the solution, then the CPU time of that neighborhood is added to its rating. Otherwise, the rating

is halved and the CPU time divided by a given influence parameter is added. If an updated rating value is smaller than the minimum rating value or bigger than the maximum rating value, the order of neighborhoods is updated. The search continues with the neighborhood that would have also been chosen according to the old ordering. In this study, initial rating values are assumed to be the averages of the CPU times of executing the neighborhoods individually right after the construction heuristic. The influence parameter is set to 1, which guarantees that CPU times have strong impact on the self-adaption procedure. The initial ordering is the same as for the sequential VND.

5 Computational Study

The computational experiment aims at quantitatively comparing the performance of our GVNS-based approach to GLS. For this purpose, we use a set of newly created data instances. The algorithms are coded in Java and executed single threaded on an Intel Core i5-3570 3.4 GHz computer.

5.1 Test instances

We randomly generated 36 data instances where the number of requests is set to be 20, 50, 100, 250, 500, and 1000. Customers are scattered on a two-dimensional plane. The coordinates (x, y) are generated as integers in the range $[-1000, 1000]$ excluding the point (0, 0), which is the location of the depot. Each request has an integer demand value between [1, 50]. Revenues are generated as to be either (i) equal for all requests (F), (ii) proportional to the demands (P), or (iii) randomly distributed (R). Additionally, the time constraints are either tight or relaxed enabling the generation of relatively short (S) and long tours (L), respectively. Vehicle numbers vary between two and eight. The test instances are named as [Index–Revenue type–Tour length type], e.g., 1FS. They are publicly available (http://prolog.univie.ac.at/research/MVPPDP/instances.zip).

5.2 Experimental results

For the GVNS, we present two VND search strategies, which are the GVNS sequential (GVNSseq) as well as the GVNS self-adaptive (GVNSsa) one (see Sect. 4.3). We run each data set 5 times with each variant. CPU time limit is set to 1, 10, and 100 s for small (<50 requests), medium (100–250 requests) and large (>250 requests), respectively.

The following tables report the gaps to the best known solutions for each algorithm and each instance. Best known solutions refer to all results ever found by GVNS and GLS, including GVNS runs with a stopping time of 20 h. Furthermore, we report the average runtime needed to reach the final solution.

Table 1 displays the results for small-sized instances, where the first half of datasets (1FS-6RL) comprises 20 and the second half (7FS-12RL) 50 customer requests. We observe that GVNSsa yields the best solution quality in minimum amount of time. Both GVNS variants clearly dominate GLS.

Table 1 Results for small instances (20 and 50 requests; see column *Req*)

Inst	Req	GVNSseq			GVNSsa			GLS		
		Best (%)	Avg (%)	Time	Best (%)	Avg (%)	Time	Best (%)	Avg (%)	Time
1FS	20	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.12
2FL	20	−3.38	−3.38	**0.00**	−3.38	−3.38	**0.00**	**−3.15**	**−3.15**	0.05
3PS	20	**−0.55**	**−0.55**	**0.00**	**−0.55**	**−0.55**	**0.00**	**−0.55**	**−0.55**	0.25
4PL	20	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	−1.14	−1.14	0.10
5RS	20	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	−0.06	0.12
6RL	20	−2.63	−2.63	**0.00**	−2.63	−2.63	**0.00**	**−2.43**	**−2.43**	0.05
7FS	50	**−0.60**	**−0.60**	**0.00**	**−0.60**	**−0.60**	**0.00**	−7.83	−7.83	0.24
8FL	50	**0.00**	**−0.37**	**0.00**	**0.00**	−0.66	**0.00**	−1.24	−3.89	0.45
9PS	50	−4.27	−5.69	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.15
10PL	50	**0.00**	**−0.77**	**0.00**	**0.00**	−3.76	**0.00**	−5.31	−7.69	0.28
11RS	50	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	−0.52	−0.89	0.27
12RL	50	**0.00**	**−0.03**	**0.00**	**0.00**	−2.01	**0.00**	−4.14	−5.22	0.40
Avg		−0.95	−1.17	**0.00**	**−0.60**	**−1.13**	**0.00**	−2.19	−2.74	0.21

Column *Best* gives the deviation (%) from the best solution found in 5 runs to the best known solution. A negative value indicates that the result is below the best known solution. Average gaps (%) to the best known solutions are in column *Avg*. Average CPU time needed to reach the final solution in seconds is displayed in column *Time*. Best solutions per line are bold

The results for the medium-sized instances with 100 (13FS-18RL) and 250 (19FS-24RL) requests are given in Table 2. On average, GLS is now faster than all GVNS variants, but with respect to solution quality GVNSseq and GVNSsa perform significantly better.

Table 3 shows the results for the large instances with 500 (25FS-30RL) and 1000 (31FS-36RL) customer requests. GLS is again quick, but GVNSseq and GVNSsa find much better solutions. It is noticeable that for the largest instances (31FS-3RL), GLS needs relatively long time to reach the reported results. There is a gap in CPU times compared to the next smaller instances (25FS-30RL). We do not observe this gap with the GVNS variants.

Total average values in Table 3 show that GLS converges to its final solutions rather quickly, but the solution quality is poor compared to both GVNSseq and GVNSsa.

It seems that the instance type, i.e., revenue generation (F, P, R) and time constraint (S, L), does not influence the algorithm's performance.

Comparing GVNSseq and GVNSsa, for instances up to 500 requests, there is no clear picture. We spent much effort in fine tuning GVNSseq, i.e., in finding a good sequence of operators, which is the reason why it shows such a strong performance. The same sequence is used in the initial phase of GVNSsa. If runtimes are short, GVNSsa cannot show its advantage, which is the self-adaption of this sequence. However, on the biggest instances (1000 requests), we see that GVNSsa is clearly dominating GVNSseq.

Table 2 Results for medium-sized instances (100 and 250 requests; see column *Req*)

Inst	Req	GVNSseq			GVNSsa			GLS		
		Best (%)	Avg (%)	Time	Best (%)	Avg (%)	Time	Best (%)	Avg (%)	Time
13FS	100	**−5.02**	**−5.22**	4.00	**−5.02**	−5.46	3.00	−6.75	−6.98	**0.61**
14FL	100	**0.00**	**−0.12**	2.40	−0.18	−4.22	2.40	−0.89	−3.04	**1.31**
15PS	100	**−0.95**	**−3.46**	3.00	−4.20	−7.22	**0.60**	−1.21	−3.80	0.63
16PL	100	**−0.61**	**−2.59**	3.60	−3.15	−3.41	3.20	−4.83	−7.60	**0.89**
17RS	100	−5.99	**−5.99**	**0.00**	−5.99	**−5.99**	**0.00**	**−3.68**	−6.05	0.43
18RL	100	**−0.28**	**−1.10**	3.20	−1.44	−2.07	**0.20**	−2.84	−5.01	1.03
19FS	250	**−4.37**	**−5.15**	**4.20**	−4.45	−5.89	5.20	−7.96	−12.15	4.81
20FL	250	**−1.40**	**−4.35**	**7.20**	−2.98	−4.84	7.60	−5.10	−5.57	11.47
21PS	250	−3.84	−4.94	4.20	**−2.77**	**−4.42**	4.40	−5.82	−9.91	**3.27**
22PL	250	−3.30	−4.35	7.00	**−2.40**	**−3.32**	**6.60**	−7.11	−9.42	6.74
23RS	250	−2.04	−3.15	3.20	**−1.11**	**−2.03**	**2.20**	−5.11	−7.69	2.98
24RL	250	−3.30	**−4.06**	5.60	**−3.22**	−5.07	6.00	−6.88	−9.14	**4.81**
Avg		**−2.59**	**−3.71**	3.97	−3.08	−4.50	3.45	−4.85	−7.20	**3.25**

Column *Best* gives the deviation (%) from the best solution found in 5 runs to the best known solution. A negative value indicates that the result is below the best known solution. Average gaps (%) to the best known solutions are in column *Avg*. Average CPU time needed to reach the final solution in seconds is displayed in column *Time*. Best solutions per line are bold

In general, we observe relatively large gaps to the best known solutions. Good solutions methods are expected to solve problems of the VRP class with gaps of 0.1–1 %. However, it seems that this guideline does not apply to problems with selective nodes (e.g., Souffriau et al. 2009b; Tricoire et al. 2013), where gaps of up to 10 % are commonly reported.

6 Conclusion and future work

In this study, we present the MVPPDP, which applies the option to select customer requests to the world of pickup and delivery. This innovative combination is of high practical relevance in particular in collaborative transportation markets. As solution methods we propose two variants of a GVNS-based approach. The local search which is vital for intensification purposes over the solution space consists of 11 neighborhoods. Since the order of the neighborhoods affects the quality of the final solution, two alternative VND heuristics are built. One of them (sequential) using a predefined sequence of neighborhoods, while the second (self-adaptive) determines the sequence by adapting itself to the input data. We implement an alternative algorithm, namely GLS, which is expected to find good results in very short amount of time. The quality of the algorithms is examined using newly created data instances covering different scenarios.

Our experiments show that both variants of GVNS outperform GLS with regard to solution quality. Generally, with GVNS we reach very good solution quality in

Table 3 Results for large instances (500 and 1000 requests; see column *Req*)

Inst	Req	GVNSseq			GVNSsa			GLS		
		Best (%)	Avg (%)	Time	Best (%)	Avg (%)	Time	Best (%)	Avg (%)	Time
25FS	500	**−2.46**	**−3.86**	84.80	−3.23	−4.22	46.60	−8.83	−11.69	**39.92**
26FL	500	−3.64	−4.23	**47.60**	**−2.67**	**−3.92**	70.80	−3.39	−6.00	93.89
27PS	500	**−1.02**	**−2.52**	82.60	−1.37	−2.99	78.60	−9.19	−11.99	**12.80**
28PL	500	**−2.54**	**−2.99**	68.80	−2.74	−3.33	55.60	−7.52	−8.71	**35.19**
29RS	500	**0.00**	−2.74	63.20	−1.95	**−2.65**	70.40	−12.04	−14.38	**13.01**
30RL	500	−1.35	**−1.95**	65.80	**−0.98**	−2.06	73.20	−6.73	−9.72	**26.12**
31FS	1000	**−7.25**	−9.96	**73.20**	−8.48	**−9.27**	90.80	−8.31	−11.02	99.90
32FL	1000	−5.39	−6.50	**70.60**	**−3.80**	**−5.26**	75.80	−4.07	−6.52	99.90
33PS	1000	−5.16	−6.13	**86.00**	**−4.31**	**−5.18**	86.20	−6.83	−7.90	92.77
34PL	1000	−3.12	−4.85	**84.00**	**−2.55**	**−3.79**	89.00	−4.92	−6.23	99.90
35RS	1000	−4.83	−5.54	76.20	**−3.59**	**−4.63**	86.20	−9.25	−12.23	**72.14**
36RL	1000	−3.20	−4.12	92.60	**−3.06**	**−4.03**	**72.20**	−5.56	−6.83	99.90
Avg		−3.33	−4.62	74.62	**−3.23**	**−4.28**	74.62	−7.22	−9.44	**65.45**
Total Avg		**−2.29**	**−3.16**	26.19	−2.30	−3.30	26.02	−4.75	−6.46	**22.97**

Column *Best* gives the deviation (%) from the best solution found in 5 runs to the best known solution. A negative value indicates that the result is below the best known solution. Average gaps (%) to the best known solutions are in column *Avg*. Average CPU time needed to reach the final solution in seconds is displayed in column *Time*. Best solutions per line are bold

a reasonable amount of time. As expected, the GLS-based method converges to final solutions very quickly. However, the solution quality falls behind the GVNS approaches for all tested instance classes.

Acknowledgements Open access funding provided by [University of Vienna]. This work is supported by FWF the Austrian Science Fund (Projectnumber P27858-G27).

References

Ackermann H, Ewe H, Kopfer H, Küfer KH (2011) Combinatorial auctions in freight logistics. In: Böse JW, Hao H, Carlos C, Shi X, Stahlbock R, Voss S (eds) Computational Logistics, Lecture Notes in Computer Science, vol 6971. Springer, Berlin, pp 1–17

Archetti C, Bianchessi N, Speranza MG (2013) Optimal solutions for routing problems with profits. Discrete Appl Math 161(45):547–557

Archetti C, Feillet D, Hertz A, Speranza MG (2009) The capacitated team orienteering and profitable tour problems. J Oper Res Soc 60(6):831–842

Archetti C, Hertz A, Speranza MG (2007) Metaheuristics for the team orienteering problem. J Heuristics 13(1):49–76

Archetti C, Speranza MG, Vigo D (2014) Vehicle routing problems with profits. In: Toth P, Vigo D (eds) Vehicle routing: problems, methods, and applications. MOS-SIAM series on optimization, Philadelphia, pp 273–297

Berbeglia G, Cordeau J-F, Laporte G (2010) Dynamic pickup and delivery problems. Eur J Oper Res 202(1):8–15

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

Bolduc M, Renaud J, Boctor F, Laporte G (2008) A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. J Oper Res Soc 59:776–787

Boussier S, Feillet D, Gendreau M (2007) An exact algorithm for team orienteering problems. 4OR 5(3):211–230

Butt SE, Cavalier TM (1994) A heuristic for the multiple tour maximum collection problem. Comput Oper Res 21(1):101–111

Chao I-M, Golden BL, Wasil EA (1996b) The team orienteering problem. Eur J Oper Res 88(3):464–474

Chbichib A, Mellouli R, Chabchoub H (2012) Profitable vehicle routing problem with multiple trips: Modeling and variable neighborhood descent algorithm. Am J Oper Res 2(6):104–119

Chu C-W (2005) A heuristic algorithm for the truckload and less-than-truckload problem. Eur J Oper Res 165(3):657–667

Croes GA (1958) A method for solving traveling-salesman problems. Oper Res 6(6):791–812

Dahl S, Derigs U (2011) Cooperative planning in express carrier networks an empirical study on the effectiveness of a real-time decision support system. Decis Support Syst 51(3):620–626

Dang DC, Guibadj R, Moukrim A (2011) A pso-based memetic algorithm for the team orienteering problem. In: Di Chio C, Brabazon A, Di Caro GA, Drechsler R, Farooq M, Grahl J, Greenfield G, Prins C, Romero J, Squillero G, Tarantino E, Tettamanzi AGB, Urquhart N, Uyar AS (eds) Applications of Evolutionary Computation, Lecture Notes in Computer Science, vol 6625. Springer, Berlin, pp 471–480

Ergun O, Kuyzu G, Savelsbergh MWP (2007a) Reducing truckload transportation costs through collaboration. Trans Sci 41(2):206–221

Ergun O, Kuyzu G, Savelsbergh MWP (2007b) Shipper collaboration. Comput Oper Res 34:1551–1560

Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. Trans Sci 39(2):188–205

Gribkovskaia I, Laporte G, Shyshou A (2008) The single vehicle routing problem with deliveries and selective pickups. Comput Oper Res 35(9):2908–2924

Hansen P, Mladenović N, Pérez JAM (2008) Variable neighbourhood search: methods and applications. 4OR 6(4):319–360

Hartl RF, Romauch M (2013) The influence of routing on lateral transhipment. In: Moreno-Daz R, Pichler F, Quesada-Arencibia A (eds) Computer Aided Systems Theory—EUROCAST 2013. Lecture Notes in Computer Science. vol 8111. Springer, Berlin. pp 267–275

Hemmati A, Hvattum LM, Fagerholt K, Norstad I (2014) Benchmark suite for industrial and tramp ship routing and scheduling problems. INFOR Inf Syst Oper Res 52(1):28–38

Hu B, Raidl G (2006) Variable neighborhood descent with self-adaptive neighborhood-ordering. In: Cotta C, Fernandez AJ, Gallardo JE (eds) Proceedings of the 7th EU/Meeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics, Malaga, Spain

Ilić A, Urošević D, Brimberg J, Mladenović N (2010) A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. Eur J Oper Res 206(2):289–300

Jepsen MK, Petersen B, Spoorendonk S, Pisinger D (2014) A branch-and-cut algorithm for the capacitated profitable tour problem. Discrete Optim 14:78–96

Ke L, Archetti C, Feng Z (2008) Ants can solve the team orienteering problem. Comput Ind Eng 54(3):648–665

Ko CS, Lee HK, Ferdinand FN, Kim T (2010) A genetic algorithm based approach to the profitable tour problem with pick-up and delivery. Ind Eng Manag Syst 9(2):80–87

Krajewska M, Kopfer H (2006) Collaborating freight forwarding enterprises. OR Spectr 28(3):301–317

Krajewska M, Kopfer H, Laporte G, Ropke S, Zaccour G (2008) Horizontal cooperation among freight carriers: request allocation and profit sharing. J Oper Res Soc 59:1483–1491

Labadie N, Mansini R, Melechovský J, Wolfler Calvo R (2012) The team orienteering problem with time windows: an LP-based granular variable neighborhood search. Eur J Oper Res 220(1):15–27

Lahyani R, Khemakhem M, Semet F (2013) Heuristics for rich profitable tour problems. In: 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), Hammamet, Tunisia. pp 1–3

Laporte G, Martello S (1990) The selective travelling salesman problem. Discrete Appl Math 26(23):193–207
Li Y, Chen H, Prins C (2016) Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. Eur J Oper Res 252(1):27–38
Lin S-W (2013) Solving the team orienteering problem using effective multi-start simulated annealing. Appl Soft Comput 13(2):1064–1073
Lin S-W, Yu VF (2012) A simulated annealing heuristic for the team orienteering problem with time windows. Eur J Oper Res 217(1):94–107
Masson R, Ropke S, Lehuédé F, Péton O (2014) A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes. Eur J Oper Res 236(3):849–862
Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100
Mladenović N, Urošević D, Hanafi S, Ilić A (2012) A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. Eur J Oper Res 220(1):270–285
Nguyen VH, Nguyen TTT (2010) Approximating the asymmetric profitable tour. Electron Notes Discrete Math 36:907–914
Nowak M, Ergun O, White CC III (2008) Pickup and delivery with split loads. Transp Sci 42(1):32–43
Parragh SN, Dörner KF, Hartl RF (2008) A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft 58:21–51
Parragh Sophie N, Doerner Karl F, Hartl Richard F (2010) Variable neighborhood search for the dial-a-ride problem. Comput Oper Res 37(6):1129–1138
Privé Julie, Renaud Jacques, Boctor Fayez, Laporte Gilbert (2006) Solving a vehicle-routing problem arising in soft-drink distribution. J Oper Res Soc 57(9):1045–1052
Psaraftis HN (2011) A multi-commodity, capacitated pickup and delivery problem: the single and two-vehicle cases. Eur J Oper Res 215(3):572–580
Puettmann C, Stadtler H (2010) A collaborative planning approach for intermodal freight transportation. OR Spectr 32(3):809–830
Qian H, Andrew L (2014) An iterative three-component heuristic for the team orienteering problem with time windows. Eur J Oper Res 232(2):276–286
Rainer-Harbach M, Papazek P, Hu B, Raidl GR (2013) Balancing bicycle sharing systems: a variable neighborhood search approach. In: Blum MMC (eds) Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, vol 7832. Springer, Berlin. pp 121–132
Rais A, Alvelos F, Carvalho MS (2014) New mixed integer-programming model for the pickup-and-delivery problem with transshipment. Eur J Oper Res 235(3):530–539
Raviv Tal, Tzur Michal, Forma. IA (2013) Static repositioning in a bike-sharing system: models and solution approaches. EURO J Transp Logist 2(3):187–229
Ropke S, Cordeau J-F (2009) Branch and cut and price for the pickup and delivery problem with time windows. Transp Sci 43(3):267–286
Ruijgrok C (2001) European transport: insights and challenges. In: Brewer A, Button KJ, Hensher DA (eds) Handbook of Logistics and Supply Chain Management, Amsterdam. pp 29–46
Savelsbergh MWP, Sol M (1995) The general pickup and delivery problem. Transp Sci 29(1):17–29
Sheffi Y (2004) Combinatorial auctions in the procurement of transportation services. Interfaces 34(4):245–252
Skjoett-Larsen T (2000) European logistics beyond 2000. J Phys Distrib Logist Manag 30(5):377–387
Souffriau W, Vansteenwegen P, Vander Berghe G, van Oudheusden D (2009a) A guided local search metaheuristic for the team orienteering problem. Eur J Oper Res 196(1):118–127
Souffriau W, Vansteenwegen P, Vander Berghe G, van Oudheusden D (2009b) Iterated local search for the team orienteering problem with time windows. Comput Oper Res 36(12):3281–3290
Souffriau W, Vansteenwegen P, Vander Berghe G, van Oudheusden D (2010) A path relinking approach for the team orienteering problem. Comput Oper Res 37(11):1853–1859
Stenger Andreas, Vigo Daniele, Enz Steffen, Schwind Michael (2013) An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. Transp Sci 47(1):64–80
Tang H, Miller-Hooks E (2005) A TABU search heuristic for the team orienteering problem. Comput Oper Res 32(6):1379–1407
Tang L, Wang X (2006) Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. Int J Adv Manuf Technol 29(11–12):1246–1258

Ting C-K, Liao X-L (2013) The selective pickup and delivery problem: formulation and a memetic algorithm. Int J Product Econ 141(1):199–211

Toth P, Vigo D (2002) The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia

Tricoire F, Romauch M, Doerner KF, Hartl RF (2013) Addendum to "heuristics for the multi-period orienteering problem with multiple time windows" (Computers & Operations Research 37(2), 351–367). Comput Oper Res 40(5):1516–1519

Vidal T, Maculan N, Vaz Penna PH, Satoru Ochi L (2016) Large neighborhoods with implicit customer selection for vehicle routing problems with profits. Transp Sci. doi:10.1287/trsc.2015.0584

Voudouris C, Tsang EPK (1996) Partial constraint satisfaction problems and guided local search. In: Proceedings of PACT'96 (Practical Application of Constraint Technology), London. pp 337–356