An Efficient Heuristic for the Multi-Vehicle Profitable Pickup and Delivery Problem

Matheus Nohra Haddad
Universidade Federal de Viçosa - Campus Rio Paranaíba - UFV
Rodovia MG-230 - Km 7, Rio Paranaíba - MG, 38810-000
matheus.haddad@ufv.br

Luiz Satoru Ochi
Universidade Federal Fluminense - UFF
Av. Gal. Milton Tavares de Souza, s/n, São Domingos, Niterói - RJ, 24210-346
satoru@ic.uff.br

Simone de Lima Martins
Universidade Federal Fluminense - UFF
Av. Gal. Milton Tavares de Souza, s/n, São Domingos, Niterói - RJ, 24210-346
simone@ic.uff.br

Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto - UFOP
Campus Morro do Cruzeiro, Ouro Preto - MG, 35400-000
marcone@ufop.edu.br

ABSTRACT

We address in this work the Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP), an important $\mathcal{NP}$-hard problem linked with various practical applications, including maritime transportation. This problem combines two challenging vehicle routing attributes: 1) one-to-one pickup and delivery and 2) customers selection. Dealing with both attributes requires advanced local-search neighborhoods and efficient exploration procedures. To solve the MVPPDP, we propose a heuristic algorithm (named IPPD) based on Iterated Local Search (ILS) and Random Variable Neighborhood Descent (RVND). This method explores infeasible solutions during the search with possible violations of duration constraints. To reduce the search space, we use neighborhood restrictions based on granular local search. Our computational experiments on 36 classical benchmark instances from the literature demonstrate the good performance of IPPD, which finds new best known solutions for 6 instances and also achieves 20 better or equal solutions from methods of the literature.

KEYWORDS. Vehicle routing. One-to-one pickup and delivery. Customer selection.

Paper topics (Metaheuristics. Combinatorial Optimization. Logistics and Transportation.)

1. Introduction

The objective of the Vehicle Routing Problem (VRP) is to find the minimum total route cost for each vehicle so that all customers are served. There are several variations of the VRP, where this constraint of visiting all customers is relaxed. In Chao et al. [1996b], the authors proposed the Team Orienteering Problem (TOP), in which the objective is to find a subset of points (customers) for the team (vehicles) to visit and also find a route for each member of the team so that the time limit per member is respected and the total collected profit (associated with each point) is maximized. Later, in Archetti et al. [2009], the authors defined the Profitable Tour Problem (PTP), changing the objective to the maximization of profit minus the minimization of the total route cost. The combination of TOP, PTP and the one-to-one Pickup and Delivery Problem (PDP) inspired the authors of Gansterer et al. [2017] to define the Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP).

The MVPPDP is a multi-level optimization problem that, firstly, requires to select a subset of customers for the vehicles to visit. Secondly, these customers must be assigned to each vehicle and finally, a route must be defined for each vehicle. These routes must be established in order to maximize the total profit, which is given by the sum of all profits, obtained from serving each customer, minus the travel costs. This problem, for example, is very relevant for maritime transportation, in particular the tramp ship routing problems [Norstad et al., 2011; Vilhelmsen et al., 2014], where cargo owners announce their transportation requests in a spot market and the shipping company decides which request to attend, considering profits, capacity and travel time.

As the MVPPDP is a "one-to-one pickup and delivery" problem, it leads to larger neighborhoods as the method for solving it must work with pairs of customers. In addition, an algorithm for solving MVPPDP must have an efficient customer selection procedure in order to obtain good solutions. These reasons can, partly, explain the existence of only two methods proposed for the MVPPDP, a variable neighborhood search and a guided local search, both presented in Gansterer et al. [2017]. The authors claimed to obtain good results by using them on generated benchmark instances for the MVPPDP.

This work presents an efficient heuristic solution method for the MVPPDP, relying on large-neighborhood search with efficient exploration procedures. The MVPPDP is a tightly constrained problem and the existing methods only accept feasible solutions during their search. This work aims to investigate the performance of an algorithm that allows infeasible solutions, related to the duration constraint, during its search procedure. By allowing infeasible solutions, the neighborhoods become larger and can contribute to find better solutions, however it can also increase the computational time. To counterbalance this effect, the concept of granular local search [Toth and Vigo, 2003] is applied to reduce the computational time on large instances.

A heuristic called IPPD based on Iterated Local Search (ILS) and the Random Variable Neighborhood Descent (RVND) is proposed. This algorithm explores classical neighborhoods for one-to-one pickup and delivery problems and also explores specific neighborhoods for selecting customers to be included or excluded from the solution. Moreover, this algorithm performs shaking procedures in order to escape from local optima. Experimental analyses on benchmark instances from the literature are also done to investigate the efficiency of the proposed methodology.

The main contributions of the work presented in this work are: a) a simple and efficient heuristic for the MVPPDP; b) a new strategy for dealing with infeasible solutions for the MVPPDP; c) granular local searches for the MVPPDP; d) new results for existing benchmark instances.

2. Problem statement

The multi-vehicle profitable pickup and delivery problem [Gansterer et al., 2017] is defined on a graph $G = (V, E)$, where $V = P \cup D \cup \{0, 2n + 1\}$ is composed by the set $P = \{1, 2, \ldots, n\}$, which contains n pickup customers, the set $D = \{n + 1, \ldots, 2n\}$, which contains the n corresponding delivery customers and the vertices $\{0, 2n+1\}$, which represent the initial and

final depots locations. For each service $i \in V$ there is a pickup customer $i \in P$ and the corresponding delivery customer $(n+i) \in D$. The set of edges is defined as $E = \{(i,j)|i,j \in V^2\}$.

To perform the services, there is also a set $K = \{1, \ldots, m\}$ of m homogeneous vehicles, in which, each vehicle $k \in K$ has the same capacity limit C. Each service i has a demand q_i, which is positive ($q_i > 0$) for the pickup customer $i \in P$ and negative ($q_{n+i} = -q_i$) for the delivery customer $(n+i) \in D$. As in Gansterer et al. [2017], we also assume that $q_i \leq Q$ for all services $i \in V$. The vehicles must depart and return from the depots without any load, that is, $q_0 = q_{2n+1} = 0$. When a vehicle arrives at a pickup customer, it must collect all available load and when it arrives at the corresponding delivery customer, all load must be delivered. Each service i has an associated revenue r_i to be gained if this service is attended, thus, not necessarily all services must be attended. For each edge $(i,j) \in E$, there is a distance cost c_{ij} associated and each vehicle is constrained by a maximum travel distance T.

The objective of the MVPPDP is to find a set of routes that maximizes the total profit. The total profit is obtained by the sum of all revenues collected minus the sum of all travel costs. Each route of the MVPPDP must start and end at the depot, respect the distance limit, the vehicle capacity and also the precedence of each pickup over its delivery in the same route.

Figure 1 illustrates a possible solution for an instance with ten p-d pairs and two available vehicles with capacity $C = 50$ and maximum travel distance $T = 5000$. In this instance, the set of pickup customers is $P = \{1, 2, \ldots, 10\}$ and the set of corresponding delivery customers is $D = \{11, 12, \ldots, 20\}$. The initial and final depots are at the same location, represented by 0. In this solution, $Route\ 1 = \{0, 10, 20, 8, 7, 18, 4, 14, 17, 0\}$ and $Route\ 2 = \{0, 9, 2, 3, 1, 11, 13, 12, 19, 0\}$. It is noteworthy that both pair of customers $(5, 15)$ and $(6, 16)$ are not supplied by any vehicle. The total distance for $Route\ 1$ is 4792.23 with the total revenue collected of 19101 and the total distance for $Route\ 2$ is 4892.25 with the total revenue collected of 27053, thus the total profit is calculated by $(19101 + 27053) - (4792.23 + 4892.25) = 36469.52$.

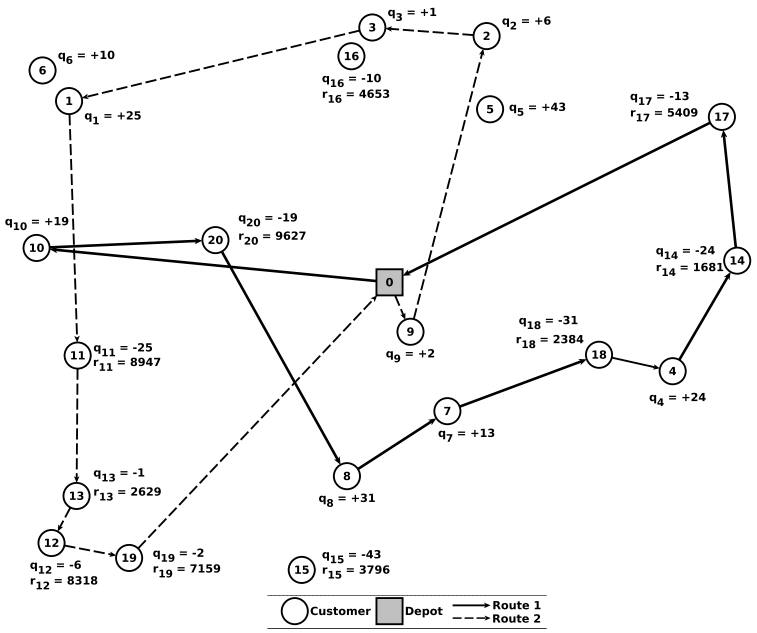

Figure 1: Example of a possible solution for the MVPPDP

3. Related literature

The team orienteering problem defined in Chao et al. [1996b] was proven to be in the $\mathcal{NP}$-hard class by Laporte and Martello [1990] and Boussier et al. [2007]. As a consequence,

many heuristics have been developed to this problem in the literature. In Labadie et al. [2012], a LP-based granular variable neighborhood search is proposed to solve the TOP with hard Time Window constraints. A simulated annealing is developed in Lin and Vincent [2012] and Lin [2013] for the TOP with Time Windows. A tabu search is proposed in Tang and Miller-Hooks [2005] for the TOP. Two variants of a generalized tabu search algorithm and a variable neighborhood search algorithm are used by Archetti et al. [2007] to solve the TOP. In Ke et al. [2008] an ant-colony optimization algorithm is proposed to solve the TOP. A particle swarm optimization-based memetic algorithm is developed in Dang et al. [2011] for solving the TOP. An iterative framework incorporating three components is developed by Hu and Lim [2014], the first two components are a local search procedure and a simulated annealing, while the third component recombines routes to identify high quality results. Finally, heuristics based on guided local search [Vansteenwegen et al., 2009a], iterated local search [Vansteenwegen et al., 2009b] and path relinking [Souffriau et al., 2010] were able to find very good solutions for the TOP in relatively short amount of time.

In contrast with the several works on the TOP, there are fewer studies on the profitable tour problem [Archetti et al., 2014]. An approximation algorithm for the asymmetric PTP is presented by Nguyen and Nguyen [2010]. Large neighborhoods for the TOP as well as for the PTP are studied in Vidal et al. [2015]. Three heuristics and one exact procedure for the capacitated TOP and the capacitated PTP were proposed in Archetti et al. [2009] and Archetti et al. [2013]. In Jepsen et al. [2014], a branch-and-cut algorithm for the capacitated PTP is presented. A rich variant of the PTP is studied in Lahyani et al. [2013] and a variable neighborhood search embedded with an adaptive large neighborhood search is developed.

The multi-vehicle profitable pickup and delivery problem was studied for the first time in Gansterer et al. [2017]. The authors developed a heuristic based on a variable neighborhood search that performs the local searches using the variable neighborhood descent. The initial solution is built using a greedy construction heuristic based on cheapest insertion ratio ($revenue/insertion\_cost$), calculated for each request and then, inserting into the partial solution the request with the highest insertion ratio. The algorithm performs local searches for minimizing the total distance travelled and for including requests that do not belong to the current solution. The shaking procedures are chosen randomly and they either remove a single request from a route or remove 10% to 40% of the route and then insert new requests based on cheapest insertion. The authors tested two VNDs, the first one executes the local searches in a sequential order (GVNSseq) and the second is based on a self-adaptive strategy that chooses the best order based on solution improvements during the execution of the algorithm (GVNSsa). The authors also developed a guided local search (GLS) based on the algorithm from Vansteenwegen et al. [2009a]. Computational experiments were conducted on 36 generated instances that contain from 20 to 1000 customers. The experiments showed that both variants of VNS outperform GLS in solution quality.

4. Methodology

Only two heuristics, one based on the variable neighborhood search (VNS) and the other based on the guided local search (GLS), have been developed specifically for the MVPPDP. Both algorithms explore only feasible solutions for the MVPPDP, in terms of the tour time constraint. Because of this constraint, the search space for the MVPPDP becomes very reduced, therefore highly efficient strategies are required to reach good solutions and also to avoid getting stuck in local minima. This work aims to investigate the exploration of a larger search space for the MVPPDP, providing an efficient algorithm for the MVPPDP that deals with solutions containing routes violating the duration constraint. The motivation of this relaxation is to help the algorithm to escape from local optima, possibly reaching better solutions. In addition, as the neighborhoods become larger, a granular search is designed to prune many unpromising moves during the search with the aim to reduce the computational time.

4.1. Evaluation of a solution

A solution s for the MVPPDP is evaluated using the evaluation function $f(s) = p(s) - c(s) - \beta d(s)$, which is responsible for calculating the total profit. In order to obtain the total profit, the evaluation function $f(s)$ subtracts the total revenue gained by s, obtained by $p(s)$, from the total travel costs of s, given by $c(s)$. Let $d(s)$ be the total tour time exceeded by the routes and β a coefficient factor used to penalize the violation of the duration constraint.

We adopted a dynamic strategy for updating the coefficient factor β during the execution of the proposed algorithm. This strategy widely used for similar problems [Cordeau and Laporte, 2003; Parragh et al., 2010; Vidal et al., 2012; Parragh et al., 2014; Vidal et al., 2014]. The coefficient factor β starts with the value $\text{MAX}_{i \in V}\{r_i\}$, which represents the maximum revenue of all requests. After each group of u_{MAX} iterations without improvement of the proposed algorithm, the value of β can be increased or decreased, depending on the performance of the algorithm in previous iterations. Let $n_{\text{INFEASIBLE}}$ be the number of all infeasible solutions s, that is if $d(s) > 0$, found by the algorithm in the last iterations and n_{FEASIBLE} the number of all feasible solutions s, that is $d(s) = 0$, obtained in the last iterations. If more infeasible solutions were produced in the last u_{MAX} iterations without any improvement, then the new value is updated to $\beta = \beta(1 + \delta)$. On the other hand, if more feasible solutions were generated then $\beta = \beta/(1 + \delta)$. The value of δ is randomly chosen in the interval $\{0.05 \ldots 0.1\}$ using an uniform distribution probability.

4.2. General structure of the method

The proposed algorithm for the MVPPDP combines an iterated local search with a random variable neighborhood descent, which performs, randomly, the local searches of ILS with the objective of finding a local optimum with respect to several neighborhoods. The pseudo code of the Iterated local search for the Profitable Pickup and Delivery problem (named IPPD) is shown in Algorithm 1.

Algorithm 1: IPPD

```
   input : T_MAX, p_MAX, u_MAX
 1 β ← max_{i∈V}{r_i};
 2 s ← greedyInitialSolution();
 3 s ← RVND(s);
 4 u ← 0;
 5 n_INFEASIBLE ← 0;
 6 n_FEASIBLE ← 0;
 7 while time ≤ T_MAX do
 8     s' ←Perturbation(s,p_MAX);
 9     s' ←RVND(s');
10     if isFeasible(s') then n_FEASIBLE ← n_FEASIBLE + 1 ;
11     else n_INFEASIBLE ← n_INFEASIBLE + 1 ;
12     if f(s') > f(s) and isFeasible(s') then s ← s' ;
13     else u ← u + 1 ;
14     if u == u_MAX then
15         δ ← rand(0.05, 0.1) ;
16         if n_FEASIBLE > n_INFEASIBLE then β ← β/(1 + δ) ;
17         else β ← β(1 + δ) ;
18         u ← 0;
19         n_INFEASIBLE ← 0;
20         n_FEASIBLE ← 0;
21     end
22 end
23 Return s ;
```

The IPPD algorithm receives as input three parameters: the time limit T_{MAX} for executing the algorithm, the maximum limit of perturbations p_{MAX} and the maximum number of iterations without improvement u_{MAX}. Firstly, the value of β is defined to be the maximum revenue of all

services (line 1). Then, a solution s is built using a greedy constructive heuristic (line 2) and this solution is improved using the RVND (line 3). Next, (lines 4 – 6), the variable that stores the current number of iterations without improvement, u, is initialized to zero, as well as the variables $n_{\text{INFEASIBLE}}$ and n_{FEASIBLE}, responsible for counting the number of infeasible and feasible solutions, respectively. Inside the iterative loop of the algorithm, firstly, a perturbation to escape from local optima is applied on the current solution s, generating a new solution s' (line 8). This new solution s' is improved by the RVND (line 9). If s' is feasible, after been improved by the RVND, then n_{FEASIBLE} is updated, otherwise $n_{\text{INFEASIBLE}}$ must be updated (lines 10 – 11). The best solution s is updated if s' has a better evaluation function value and also s' must be feasible, if not, then u is incremented (lines 12 – 13). In the end of the process (lines 14 – 21), if the algorithm has not found improvement on u_{MAX} iterations, then the value of β is decreased if n_{FEASIBLE} is greater than $n_{\text{INFEASIBLE}}$ (line 16) or increased if n_{FEASIBLE} is less than or equal to $n_{\text{INFEASIBLE}}$ (line 17). After updating β, variables u, $n_{\text{INFEASIBLE}}$ and n_{FEASIBLE} are set to zero again (lines 18 – 20). This iterative loop continues until a termination criterion is reached, defined here by a time limit (line 7). In the end of the execution the best solution found is returned (line 23).

The following components of the algorithm are detailed next: the construction of the initial solution, the local search procedures and the perturbation operator.

4.3. Initial solution

The initial solution s is produced by a greedy constructive heuristic, inspired by Chao et al. [1996b,a] and also used in Gansterer et al. [2017]. Initially m seed *p-d* customers are included into each route, which are the farthest away from the depot. Then, to fill up the routes, this heuristic iteratively computes for each pickup customer i its best insertion ratio, which is the maximum revenue divided by the minimum increase of distance. The pickup customer i with the maximum insertion ratio is inserted at each iteration, together with its corresponding delivery $(n + i)$, which is included into its best position after i. If no more customers can be included into the solution, because of the distance constraints, then a new solution is created and the procedure finishes.

4.4. Local search procedures

An important concept, center of gravity (COG) for a route k, also used in Tsiligirides [1984], Vansteenwegen et al. [2009a] and Gansterer et al. [2017], must be defined here to understand how the *GravityCenterExchange* neighborhood (Section 4.4.4) is investigated. The COG is based on the Cartesian coordinates (x_i, y_i) of all customers included into the solution weighted by their corresponding revenues r_i. The Cartesian coordinates of the center of gravity $(x_{\text{COG}}, y_{\text{COG}})$ are given by $x_{\text{COG}} = \sum_{i \in k} x_i r_i / \sum_{i \in k} r_i$ and $y_{\text{COG}} = \sum_{i \in k} y_i r_i / \sum_{i \in k} r_i$.

By knowing the center of gravity, then the appropriateness A_i can be calculated $A_i = r_i / c_{i,\text{COG}}$, where $c_{i,\text{COG}}$ is the distance cost between the customer i and the COG. Like in Gansterer et al. [2017], we consider that an appropriateness for a pickup and delivery pair is given by the sum $A_i + A_{n+i}$. The appropriateness is used in profit-increasing neighborhoods for inserting new requests into the solution.

4.4.1. Random Neighborhood Variable Descent

In the random neighborhood variable descent of IPPD, just like the RVNDs in Souza et al. [2010] and Subramanian et al. [2010], there is no predefined order for exploring neighborhoods, that is, before every execution of the local search, a new neighborhood order is randomly chosen. Each neighborhood is defined relatively to one type of move, which can be applied on different p-d pairs and routes. Each neighborhood is efficiently pre-evaluated exhaustively, considering the moves in random order of p-d pairs, and applying the best improving move. After each improvement, the search restarts from the first neighborhood structure. Otherwise, the search continues on the next neighborhood structure and finishes when all neighborhoods have been examined without success.

Due to the fact that the IPPD deals with infeasible solutions during its search, the search space becomes much bigger, thus the time needed to explore efficiently the search space grows

proportionally. To avoid spending much time on each neighborhood and to seek improvements quickly, the IPPD follows Toth and Vigo [2003] and adopts a similar idea of the granular local search. In the IPPD, a set of nearest neighborhoods for each pickup customer is initially defined and, before each move is applied. It is verified if this move involves at least one customer belonging to the set of nearest neighborhoods for this pickup customer. If this is the case, then the move is applied, if not, then this move is rejected. In IPPD, the size of this set for each pickup customer was set to $\{50, 100, 150, 200, 250, 300\}$ and the value that provided better results was 250.

All intra-route and inter-route neighborhoods used by IPPD are defined next.

4.4.2. Intra-route neighborhood structures:

$N^{(1)}$ – *PairSwap* considers two pairs of customers $(i, n+i)$ and $(j, n+j)$ and swaps the pickup customer i with the pickup customer j, as well as the delivery customer $(n+i)$ with the delivery customer $(n+j)$.

$N^{(2)}$ – *PairShift* considers a pair of customers $(i, n+i)$ and relocates the pickup customer i in another position of the route and then finds another position to insert the corresponding delivery customer $(n+i)$, after the pickup customer.

$N^{(3)}$ – *PickShift* relocates a pickup customer i in another position before the delivery customer $(n+i)$.

$N^{(4)}$ – *DelShift* relocates a delivery customer $(n+i)$ in another position after the pickup customer i.

4.4.3. Inter-route neighborhood structures:

$N^{(5)}$ – *InterPairSwap* selects a pair of customers $(i, n+i)$ from a route k_1 and another pair $(j, n+j)$ from a route k_2 and swaps the pickup customer i with the pickup customer j. The delivery customer $(n+i)$ is swapped with the delivery customer $(n+j)$.

$N^{(6)}$ – *InterPairShift* takes a pair of customers $(i, n+i)$ from a route k_1 and transfer this pair to a route k_2. After defining the position to insert the pickup customer i in k_2, the delivery customer $(n+i)$ is inserted in a following position.

4.4.4. Profit-increasing neighborhood structures:

$N^{(7)}$ – *Insert* takes a pair of customers $(i, n+i)$ not included in the solution and insert this pair on a route k. The pickup customer i is inserted on a position of k and the delivery customer $(n+i)$ is inserted on a following position of k.

$N^{(8)}$ – *Replace* takes a pair of customers $(i, n+i)$ not included in the solution and a pair of customers $(j, n+j)$ that belongs to the solution and swaps them.

$N^{(9)}$ – *GravityCenterExchange* removes the farthest pair $(i, n+i)$ from the center of gravity from a route k and inserts non-included pairs into k as long as the duration constraints are met. New requests are inserted by considering a descending order of appropriateness.

4.4.5. Repairing neighborhood structure:

$N^{(10)}$ – *Remove* takes a pair of customers $(i, n+i)$ and removes it from the solution.

4.5. Perturbation operator

The perturbation operator of the IPPD algorithm is based on the *Remove* neighborhood and consists in removing n_{PERT} random p-d pairs from the solution. The number of pairs n_{PERT} to be removed is randomly selected in $\{1, 2, ..., p_{\text{MAX}}\}$ using an uniform distribution probability. Thus, p_{MAX} is a parameter of IPPD that limits the number of perturbation moves.

5. Computational results

The set of instances from Gansterer et al. [2017] was used in order to attest the performance of the IPPD algorithm. This set contains 36 instances divided in six subsets of six instances each. These subsets have 10, 25, 50, 125, 250 and 500 pickup and delivery pairs, thus corresponding to 20, 50, 100, 250, 500 and 1000 customers. They are subdivided into groups called: small (20 and 50 customers), medium (100 and 250 customers) and large (500 and 1000 customers). The customer locations were randomly generated on a bi-dimensional plane (x, y), in which both x and y are in the interval $[-1000, 1000]$ and both depots are located at $(0, 0)$. Each pair has an integer demand between $[1, 50]$. The revenues were generated using three strategies: *i)* equal for all pairs (F); *ii)* proportional to the demands (P); and *iii)* randomly distributed (R). The distance constraint can be tight (S) or relaxed (L) and the vehicle numbers vary from 2 to 8.

The IPPD algorithm was developed in C++ using OptFrame [Coelho et al., 2011]. Each test was performed on a single core of a Intel Core i7 3.4 GHz, 16 GB of RAM using Ubuntu 14.04. It is noteworthy that the computer used is similar to the one used in Gansterer et al. [2017]. Moreover, the IPPD uses three main parameters: the strength of the perturbation operator p_{MAX}, which has been set to $\max_{k \in K} |k|$, representing the maximum route size of the current solution (this value is a solution-dependent parameter and provided good solutions during our preliminary experiments), u_{MAX}, the maximum number of iterations without improvement to update the coefficient factor β, which has been set to 5 iterations at most and finally, the stopping criterion T_{MAX}, which has been set for each group of instances to the same CPU time of the current state-of-the-art methods from Gansterer et al. [2017]: 1 second per run for each instance with 20 and 50 customers, 10 seconds for each instance with 100 and 250 customers, and 100 seconds for each instance with 500 and 1000 customers.

Previous authors have reported results over 5 runs, thus we also report the results on 5 runs of the algorithm. For each instance, we obtain a solution using IPPD, z_{IPPD}. The percentage gap relative to the best known solution (BKS), z_{BKS} is computed as $Gap = 100 \times (z_{\text{IPPD}} - z_{\text{BKS}})/z_{\text{IPPD}}$. All best known solutions were collected from Gansterer et al. [2017] after 20 hours executing their algorithms on each instance.

The objective in MVPPDP is to maximize the total profit, thus if a negative gap is found, it means that this solution has a lower value compared to the best known solution, therefore worst quality. On the other hand, if a positive value is found, it means that the solution obtained has a greater value than the best known solution, hence a new best known solution is found.

5.1. Performance comparisons on MVPPDP instances

Tables 1, 2 and 3 display the Gap_{BKS} and the Gap_{Avg} obtained after 5 runs for each algorithm on each instance from Gansterer et al. [2017]. The Gap_{BKS} represents the relative percentage deviation given by the best solution obtained on each algorithm (5 runs) and the BKS obtained in Gansterer et al. [2017] after 20 hours running their algorithms for each instance. The Gap_{Avg} is the relative percentage deviation calculated with the average solutions on 5 runs and the BKS. The algorithms GVNSseq, GVNSsa and GLS were all implemented in Gansterer et al. [2017] and the IPPD corresponds to our algorithm. For each instance, the best result considering all methods is highlighted in boldface.

Looking at these tables, it is evident the good performance of the IPPD algorithm on small and medium-sized instances, as the average gaps are greater than gaps from the other algorithms: -0.43% for the small set and -3.58% for the medium set. For the large set, the best performance observed is for the GVNS with the self-adaptive VND (GVNSsa). It seems that the IPPD did not have enough time to explore the search space of feasible and infeasible solutions, even adopting the concept of granular local search. Still, it could achieve better solutions, on average, on two instances of 500 customers and one of 1000 customers. By working with infeasible solutions, the IPPD was able to reach new better solutions for the instances **16PL**, **4PL**, **6RL**, **27PS**, **28PL** and **33PS**, because the gaps found are greater than zero. To sum up, the IPPD achieved greater or equal

Table 1: Results for the MVPPDP with 20 and 50 customers on 5 runs for each instance. Time limit set to 1 second per run.

		GVNSseq		GVNSsa		GLS		IPPD	
Inst	**Req**	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$
1FS	**20**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
2FL	**20**	-3.38	-3.38	-3.38	-3.38	-3.15	-3.15	**-0.41**	**-0.41**
3PS	**20**	**-0.55**	**-0.55**	**-0.55**	**-0.55**	**-0.55**	**-0.55**	**-0.55**	**-0.55**
4PL	**20**	0.00	0.00	0.00	0.00	-1.14	-1.14	**0.70**	**0.70**
5RS	**20**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	-0.06	**0.00**	**0.00**
6RL	**20**	-2.63	-2.63	-2.63	-2.63	-2.43	-2.43	**0.04**	**0.04**
7FS	**50**	**-0.60**	**-0.60**	**-0.60**	**-0.60**	-7.83	-7.83	**-0.60**	**-0.60**
8FL	**50**	**0.00**	-0.37	**0.00**	-0.66	-1.24	-3.89	**0.00**	**0.00**
9PS	**50**	-4.27	-5.69	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
10PL	**50**	**0.00**	**-0.77**	**0.00**	-3.76	-5.31	-7.69	**0.00**	-2.32
11RS	**50**	**0.00**	**0.00**	**0.00**	**0.00**	-0.52	-0.89	**0.00**	**0.00**
12RL	**50**	**0.00**	**-0.03**	**0.00**	-2.01	-4.14	-5.22	-1.95	-1.95
	Avg	-0.95	-1.17	-0.60	-1.13	-2.19	-2.74	**-0.23**	**-0.43**

Table 2: Results for the MVPPDP with 100 and 250 customers on 5 runs for each instance. Time limit set to 10 seconds per run.

		GVNSseq		GVNSsa		GLS		IPPD	
Inst	**Req**	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$
13FS	**100**	**-5.02**	**-5.22**	**-5.02**	-5.46	-6.75	-6.98	-5.83	-5.98
14FL	**100**	**0.00**	**-0.12**	-0.18	-4.22	-0.89	-3.04	**0.00**	-3.17
15PS	**100**	-0.95	-3.46	-4.20	-7.22	-1.21	-3.80	**-0.39**	**-2.99**
16PL	**100**	-0.61	-2.59	-3.15	-3.41	-4.83	-7.60	**0.60**	**0.02**
17RS	**100**	-5.99	-5.99	-5.99	-5.99	-3.68	-6.05	**0.00**	**-5.02**
18RL	**100**	-0.28	-1.10	-1.44	-2.07	-2.84	-5.01	**0.00**	**-0.60**
19FS	**250**	**-4.37**	**-5.15**	-4.45	-5.89	-7.96	-12.15	-4.54	-6.37
20FL	**250**	**-1.40**	**-4.35**	-2.98	-4.84	-5.10	-5.57	-4.26	-5.49
21PS	**250**	-3.84	-4.94	-2.77	-4.42	-5.82	-9.91	**-1.01**	**-3.59**
22PL	**250**	-3.30	-4.35	-2.40	-3.32	-7.11	-9.42	**-1.09**	**-3.12**
23RS	**250**	-2.04	-3.15	**-1.11**	**-2.03**	-5.11	-7.69	-1.40	-3.57
24RL	**250**	-3.30	-4.06	-3.22	-5.07	-6.88	-9.14	**-1.79**	**-3.14**
	Avg	-2.59	-3.71	-3.08	-4.50	-4.85	-7.20	**-1.64**	**-3.58**

average gaps on 20 instances, the GVNSsa obtained 14 better or equal average gaps, the GVNSseq found 13 greater or equal average gaps and the GLS was not able to find better gaps, only the same values for just 3 instances.

The statistical significance of these results is investigated by performing a Friedman test to compare the average gap values obtained for each algorithm on each instance. The Friedman test returned a value $p < 4.5 \times 10^{-8}$, which means that there is a significant difference of performance. Pairwise Wilcoxon tests were also performed to locate these differences and the tests returned: IPPD–GLS = 0.00, IPPD–GVNSseq = 0.60 and IPPD–GVNSsa = 0.41. Thus, these tests confirmed that IPPD is significantly better than the GLS, but it does not differ from the GVNSseq and the GVNSsa, probably because of the performance presented on large instances. Nevertheless, because of its good performance on small and medium-size instances, the IPPD still can be considered a good choice for solving the MVPPDP, as it can achieve good solutions by exploring larger and infeasible neighborhoods.

Table 3: Results for the MVPPDP with 500 and 1000 customers on 5 runs for each instance. Time limit set to 100 seconds per run.

		GVNSseq		**GVNSsa**		**GLS**		**IPPD**	
Inst	**Req**	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$	**Gap**$_{BKS}$	**Gap**$_{Avg}$
25FS	**500**	**-2.46**	**-3.86**	-3.23	-4.22	-8.83	-11.69	-4.04	-7.32
26FL	**500**	-3.64	-4.23	**-2.67**	**-3.92**	-3.39	-6.00	-4.10	-7.08
27PS	**500**	-1.02	-2.52	-1.37	-2.99	-9.19	-11.99	**0.72**	**-1.40**
28PL	**500**	-2.54	-2.99	-2.74	-3.33	-7.52	-8.71	**1.82**	**-0.50**
29RS	**500**	**0.00**	-2.74	-1.95	**-2.65**	-12.04	-14.38	-6.29	-8.14
30RL	**500**	-1.35	**-1.95**	-0.98	-2.06	-6.73	-9.72	**-1.29**	-2.07
31FS	**1000**	**-7.25**	-9.96	-8.48	**-9.27**	-8.31	-11.02	-11.80	-14.09
32FL	**1000**	-5.39	-6.50	**-3.80**	**-5.26**	-4.07	-6.52	-10.12	-11.39
33PS	**1000**	-5.16	-6.13	-4.31	-5.18	-6.83	-7.90	**1.47**	**-1.49**
34PL	**1000**	-3.12	-4.85	**-2.55**	**-3.79**	-4.92	-6.23	-3.26	-4.63
35RS	**1000**	-4.83	-5.54	**-3.59**	**-4.63**	-9.25	-12.23	-4.90	-6.53
36RL	**1000**	-3.20	-4.12	**-3.06**	**-4.03**	-5.56	-6.83	-2.90	-4.29
	Avg	-3.33	-4.62	**-3.23**	**-4.28**	-7.22	-9.44	-3.72	-5.75

6. Conclusions

The study of the multi-vehicle profitable pickup and delivery problem has been conducted in this work. A solution for the MVPPDP is not so trivial to find, because firstly, a good selection of a subset of customer requests is needed, next an efficient decision of defining the routes for each vehicle, considering a maximum travel time, the capacity of the vehicle and the precedence of a pickup over the delivery for each request. These characteristics lead the search space of the MVPPDP to have many basins of attractions. Thus, an efficient method for solving the MVPPDP must have good techniques to avoid getting trapped in these basins.

We designed an algorithm with the objective of escaping from these basins of attractions, by allowing it to work with infeasible solutions. This algorithm, called IPPD, is based on the iterated local search and applies, successively, local searches using the random variable neighborhood descent. The local searches explore neighborhoods for optimizing the order of visits for each vehicle and also for increasing the total profits. In order to reduce the computational time for all neighborhoods, because the search space is increased when dealing with infeasible solutions, the idea of granular local search was applied in the IPPD.

The proposed algorithm was tested on benchmark instances and the results obtained were compared to three algorithms (GLS, GVNSsa and GVNSseq) from literature and the IPPD produced the best results on small and medium-size instances within the same time limits. It was even able to find new best solutions on 6 instances. However, probably because of the time limits, the IPPD did not achieve the best solutions for large instances, but its performance was not so far from the other two algorithms (GVNSsa and GVNSseq), because no statistical difference was found, considering the GAPs found by the algorithms. Thus, the IPPD has potential to be adapted to quickly produce better solutions for the large set of instances of the MVPPDP.

Acknowledgements

This research was partially supported by CNPq, as well as CAPES and FAPEMIG, Brazil.

References

Archetti, C., Bianchessi, N., and Speranza, M. G. (2013). Optimal solutions for routing problems with profits. *Discrete Applied Mathematics*, 161(4):547–557.

Archetti, C., Feillet, D., Hertz, A., and Speranza, M. G. (2009). The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, 60(6):831–842.

Archetti, C., Hertz, A., and Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76.

Archetti, C., Speranza, M. G., and Vigo, D. (2014). Vehicle routing problems with profits. *Vehicle Routing: Problems, Methods, and Applications*, 18:273.

Boussier, S., Feillet, D., and Gendreau, M. (2007). An exact algorithm for team orienteering problems. *4OR: A Quarterly Journal of Operations Research*, 5(3):211–230.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996a). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489. ISSN 0377-2217.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996b). The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474. ISSN 0377-2217.

Coelho, I. M., Munhoz, P. L. A., Haddad, M. N., Coelho, V. N., Silva, M. M., Souza, M. J. F., and Ochi, L. S. (2011). A computational framework for combinatorial optimization problems. In *VII ALIO/EURO Workshop on Applied Combinatorial Optimization*, p. 51–54, Porto.

Cordeau, J.-F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594.

Dang, D.-C., Guibadj, R. N., and Moukrim, A. (2011). A pso-based memetic algorithm for the team orienteering problem. In *European Conference on the Applications of Evolutionary Computation*, p. 471–480. Springer.

Gansterer, M., Küçüktepe, M., and Hartl, R. F. (2017). The multi-vehicle profitable pickup and delivery problem. *OR Spectrum*, 39(1):303–319. ISSN 1436-6304.

Hu, Q. and Lim, A. (2014). An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 232(2):276–286.

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

Ke, L., Archetti, C., and Feng, Z. (2008). Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54(3):648–665.

Labadie, N., Mansini, R., Melechovskỳ, J., and Calvo, R. W. (2012). The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15–27.

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

Laporte, G. and Martello, S. (1990). The selective travelling salesman problem. *Discrete applied mathematics*, 26(2-3):193–207.

Lin, S.-W. (2013). Solving the team orienteering problem using effective multi-start simulated annealing. *Applied Soft Computing*, 13(2):1064–1073.

Lin, S.-W. and Vincent, F. Y. (2012). A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1):94–107.

Nguyen, V. H. and Nguyen, T. T. T. (2010). Approximating the asymmetric profitable tour. *Electronic Notes in Discrete Mathematics*, 36:907–914.

Norstad, I., Fagerholt, K., and Laporte, G. (2011). Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies*, 19(5):853 – 865. ISSN 0968-090X. Freight Transportation and Logistics (selected papers from {ODYSSEUS} 2009 - the 4th International Workshop on Freight Transportation and Logistics).

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 37(6):1129–1138.

Parragh, S. N., Pinho de Sousa, J., and Almada-Lobo, B. (2014). The dial-a-ride problem with split requests and profits. *Transportation Science*, 49(2):311–334.

Souffriau, W., Vansteenwegen, P., Berghe, G. V., and Van Oudheusden, D. (2010). A path relinking approach for the team orienteering problem. *Computers & operations research*, 37(11):1853–1859.

Souza, M., Coelho, I., Ribas, S., Santos, H., and Merschmann, L. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, 207(2):1041–1051.

Subramanian, A., Drummond, L., Bentes, C., Ochi, L., and Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.

Tang, H. and Miller-Hooks, E. (2005). A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333–346.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35(9):797–809. ISSN 01605682, 14769360.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Van Oudheusden, D. (2009a). A guided local search metaheuristic for the team orienteering problem. *European journal of operational research*, 196(1):118–127.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Van Oudheusden, D. (2009b). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3): 658 – 673. ISSN 0377-2217.

Vidal, T., Maculan, N., Ochi, L. S., and Vaz Penna, P. H. (2015). Large neighborhoods with implicit customer selection for vehicle routing problems with profits. *Transportation Science*, 50(2):720–734.

Vilhelmsen, C., Larsen, J., and Lusby, R. M. (2014). *Tramp Ship Routing and Scheduling - Incorporating Additional Complexities*. PhD thesis.