RAIRO-Oper. Res. 50 (2016) 503–517
DOI: 10.1051/ro/2015030

RAIRO Operations Research
www.rairo-ro.org

THE TEAM ORIENTEERING PICK-UP AND DELIVERY PROBLEM WITH TIME WINDOWS AND ITS APPLICATIONS IN FLEET SIZING

D.G. Baklagis[1], G. Dikas[1] and I. Minis[1]

Abstract. In this paper we consider the problem of prioritized pick-up and delivery operations under resource constraints. Our proposed formulation combines the Team Orienteering Problem with the case of Pick-up and Delivery with Time Windows and Capacity Constraints. We solved this model to optimality using an exact Branch-and-Price method, which is based on previous work. To study the performance of the solution method and its refinements, we conducted extensive computational experiments. We also applied the proposed model and method to a relevant logistic system and investigated its performance under various conditions. Finally, we present a practical method to determine the most suitable fleet configuration for a pick-up and delivery system that delivers prioritized operations to a known client base.

Mathematics Subject Classification. 68W01, 65K05, 90C06, 90C08, 90C11, 90C27, 90C29.

Received February 11, 2014. Accepted August 17, 2015.

1. Introduction

Optimizing routing decisions has significant implications in improving the efficiency of logistic operations. In this paper we deal with the problem of a fleet of vehicles that perform prioritized pick-up and delivery operations to serve a known client base under resource constraints. Specifically, we consider a limited fleet of identical capacitated vehicles that serve pick-up and delivery requests. A request, which must be served by a single vehicle, comprises a quantity of goods that should be picked up from an origin and delivered to a destination within certain time intervals. The vehicles are initially located at a central depot to which they must return after completing their routes. Due to limitations concerning the number of vehicles and their capacity, the vehicle fleet may be unable to serve all requests during the available planning horizon. Priority among requests is defined by assigning a known profit to each. The problem seeks to determine which requests will be served in order to collect the maximum overall profit.

We propose a new model to describe this problem. It combines the Team Orienteering Problem (TOP) as introduced in reference [6], and the Pick-up and Delivery Problem with Time Windows and Capacity Constraints (PDPTWCC), as presented in reference [12]. We designate the proposed problem as the Team Orienteering Pick-up and Delivery Problem with Time Windows (TOPDPTW).

Keywords. Pick-up and delivery problem, Team orienteering problem, Branch and Price, Vehicle Fleet Sizing.

[1] Design, Operations and Production Systems Lab, Department of Financial and Management Engineering, University of the Aegean, 41, Kountourioti Street, 82100 Chios, Greece. i.minis@fme.aegean.gr

The basic difference between TOPDPTW and PDPTWCC is that in the latter all requests are served and the total travel cost is minimized, while in TOPDPTW the resources are insufficient to serve all requests. In this case the objective is to maximize the profit occurring from serving selected requests.

The pickup and delivery problem (PDP) is a generalization of the vehicle routing problem (VRP). In the PDP each customer request is related to two different locations: a pickup location where products (or passengers) must be picked up, and a destination location where they must be delivered. The PDP's objective is to serve all requests minimizing the total travel cost. Over the past decades, several PDP classes have been studied and numerous papers and books are available in the literature; interested readers may refer to [4, 21, 22, 25]. According to reference [27] the PDP is a NP-hard problem since it is a generalization of the VRP. Taking into consideration the classification introduced by [4] to characterize types of PDP, the TOPDPTW belongs to the *one-to-one* (1-1) category of problems, since each pickup node is associated with a unique delivery node.

On the other hand, the Orienteering Problem (OP) was named after the game of "Orienteering" [6]. In the OP, a vehicle starts from a certain starting point and attempts to visit as many points as possible, striving to maximize the score collected by covering a distance or time within a predefined limit. OP could be considered as a combination of the traveling Salesman Problem (TSP) and the Knapsack Problem [29]. Early applications of the OP were presented by [28], where a travelling salesperson within limited time must serve those customers who maximize total sales. [15] also modeled an inventory routing problem in an attempt to deliver heating fuel on a daily basis to maximize profit. They showed that OP is a NP-hard problem. The Team Orienteering Problem (TOP) is an extension of the OP; in TOP a team of players attempt to maximize the total points collected by the team.

In contrast to the classical routing problems, in which all customers must be served in order to minimize the distance traveled, routing problems aiming to maximize profit have received limited attention [29]. According to the latter authors, only [1, 5] have assumed capacitated vehicles in the OP; this feature is a key aspect of our work.

Recently researchers are focusing on problems that incorporate aspects of both TOP and PDP. In reference [1] the authors address a PDP case in which some pick-up nodes may not be visited but all delivery nodes should be served. Their proposed problem is solved using a memetic algorithm. The design of a public bus service which provides door-to-door services to people with disabilities is studied in reference [11]. In this case, the objective is to maximize the number of PD requests under route sequence constraints. The Pickup and Delivery Selection Problem addressed by reference [25] aims to identify which routes to execute in order to maximize total profit without considering a limit on the number of the available vehicles. The authors proposed a hybrid Genetic Algorithm to solve the PDSP. The difference with the current work is subtle but essential, since in [25] the objective is to identify for which orders executing a new route (or combination of routes) will contribute to the total profit, while in the current work the objective is to serve those orders that maximize profit while considering the constraints in resources.

In this paper we first model TOPDPTW by a binary programming formulation that is suitable for applying the exact *Branch-and-Price* (BP) method [2, 7]. The BP method consists of the *Column Generation* (CG) algorithm embedded in a *Branch and Bound* (BB) tree [8]. Reference [10] mentions that BP techniques are suitable for problems with large numbers of variables [2]. We performed extensive computational experiments to test the proposed solution method. We also applied the model and solution method in order to determine the appropriate configuration of logistic fleet that serves a certain client base at a desired service level. For investigating this case, we considered the benchmark instances found in reference [19] for the PDPTW, which we modified appropriately.

The rest of the paper is structured as follows: Section 2 describes the binary programming formulation for TOPDPTW. Section 3 presents the solution method including a proposed refinement for addressing the sub-problem of the BP framework. Section 4 presents the experimental study conducted in order to assess the method's performance. Section 5 focuses on the fleet sizing application. Finally, Section 6 discusses the conclusions of the current work.

2. Formulating the TOPDPTW

In order to describe the TOPDPTW, we assume a directed graph $G = (NA)$ where $N = \{0, 2n+1\} \cup P \cup D$ is the node set. In this notation n is the number of requests, and nodes and $2n+1$ are the starting and destination depot(s), respectively. Furthermore, $P = \{1, \ldots, n\}$ is the set of the pick-up nodes, and $D = \{n+1, \ldots, 2n\}$ is the set of the delivery nodes, considering that each request i is associated with node i of set P and node $i+n$ of set D. Let p_i be the profit gained after serving request i; the profit is assumed to be collected at the pick-up node. Also, assume that set A contains all arcs (ij) connecting the nodes of set N, and t_{ij} is the time required to traverse arc (ij). s_i represents the time when the service of node i starts, and $[e_i, b_i]$ is the time interval within which each node must be served. Note that the allowed period of operation for the fleet is defined by the time interval $[e_0, b_{n+1}]$. Also q_i is the load the vehicle carries after serving node i and d_i is the demand of each node with $d_i = -d_{i+n}$, $i \in P$. Finally, let V be the number of available vehicles, each of them with capacity of Q units.

TOPDPTW consists of finding the set of vehicle routes M for which the total profit collected is maximized. The latter should be achieved considering that each vehicle route must start from node and end at node $2n+1$. Each node i is served at most once, and the related service should commence within time interval $[e_i b_i]$. Furthermore, if $i \in P$ is served by vehicle m, then $i+n \in D$ is also served by m and *vice versa*. The service of each node $i \in P$ must precede the service of the related node $i+n$ Furthermore, $|M| \leqslant V$ and $q_i \leqslant Q$ for each $i \in N$.

To formulate TOPDPTW as a binary program, let Ω be the set of all feasible routes, considering the aforementioned constraints. Let $\hat{p}_r = \sum\limits_{i \in P} a_{ir} p_i$ be the profit collected from route r, in which $a_{ir} = 1$ if request i is served in route r and $a_{ir} = 0$ otherwise. Also, let the binary decision variable x_r to be equal to 1 if route r participates in the solution, else $x_r = 0$. The objective function aims to maximize the total profit collected:

$$TP = \max \sum_{r \in \Omega} \hat{p}_r x_r \tag{2.1}$$

Subject to:

$$\sum_{r \in \Omega} a_{ir} x_r \leqslant 1, i \in P \tag{2.2}$$

$$\sum_{r \in \Omega} x_r \leqslant V \tag{2.3}$$

$$x_r \in \{0, 1\}\, r \in \Omega. \tag{2.4}$$

Inequality (2.2) guarantees that the set of requests are served at most once and inequality (2.3) limits the number of vehicles. Note that in the above formulation it is assumed that any vehicle route $r \in \Omega$ is feasible. The conditions to be satisfied for a route to be feasible have been described above, and are formulated mathematically in the following Section.

Since the size of Ω is too large, even for problems of limited size, we propose a Branch a Price solution framework to obtain the optimal solution, which is also described below.

3. A Branch and Price framework and solution approach

3.1. Branch and price framework

As mentioned before, the Branch and Price solution framework consists of the Column Generation (CG) technique embedded into a Branch and Bound scheme. CG is applied to solve the linear relaxation of model (2.1)−(2.4), which is called the Master Problem (MP). The relaxation is achieved by converting Constraint (2.4) to: $0 \leqslant x_r \leqslant 1, r \in \Omega$.

CG comprises two parts: (a) the Restricted Master Problem (RMP) and (b) the Sub-Problem (SP). The RMP at iteration t considers only a subset $\Omega_t \subseteq \Omega$ of MP's variables. The SP generates feasible routes (columns) r by exploiting the *dual* or *shadow prices* provided by the solution of RMP. These generated routes (columns) form a set ω that enhances the restricted set Ω_t through $\Omega_{t+1} = \Omega_t \cup \omega$ This iterative procedure is repeated until no new variables can be generated, *i.e.* $\omega = \emptyset$. Interested readers may refer to [2,7,20] for further details on CG.

Specifically, the aim of SP at each iteration t is to generate feasible routes $\omega \subseteq \Omega \backslash \Omega_t$ with positive enhanced profit which will improve the value of the objective function; that is:

$$\sum_{i \in a_{ir}} (a_{ir} p_i - a_{ir} \pi_i) - \pi_0 > 0. \tag{3.1}$$

where p_i is the profit gained after serving request i, π_i are shadow prices associated with Inequality (2), and π_0 is the shadow price associated with the vehicle Constraint (2.3). The SP that generates feasible routes and thus should satisfy all route feasibility conditions discussed in the previous Section. It may be formulated and treated as an Elementary Shortest Path Problem with Time Windows, Capacity Constraints, and Pick-up and Deliveries (ESPPTWCCPD) as presented by [24].

3.2. Formulating the SP

Since the ESPPTWCCPD is a minimization problem, condition (3.1) may be transformed to the equivalent:

$$\sum_{i \in a_{ir}} (a_{ir} \pi_i - a_{ir} p_i) + \pi_0 < 0. \tag{3.2}$$

Furthermore, it is typical to: (a) introduce the binary variables $x_{ij}, ij \in N$ that receive the value 1 if the vehicle traverses arc (ij) and (b) associate a cost to each arc, as follows:

$$c_{ij} = \begin{cases} \pi_i - p_i, & i \in P, \quad j \in N \\ \pi_0, & i \in \{0\}, \quad j \in N \\ 0, & i \in D \cup \{n+1\}, \quad j \in N. \end{cases} \tag{3.3}$$

Then, the route with the most positive enhanced profit (to maximize the objective function of RMP) may be identified by solving the following program:

$$\min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{3.4}$$

subject to:

$$\sum_{j \in N} x_{ij} \leqslant 1, \;\; i \in P \tag{3.5}$$

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{n+i,j} = 0, i \in P \tag{3.6}$$

$$\sum_{i \in N} x_{i,2n+1} = \sum_{j \in N} x_{0j} = 1 \tag{3.7}$$

$$\sum_{j \in N} x_{ji} - \sum_{j \in N} x_{ij} = 0, i \in P \cup D \tag{3.8}$$

$$s_j \geqslant (s_i + t_{ij})\, x_{ij}, \quad i \in N j \in N \tag{3.9}$$

$$q_j \geqslant (q_i + d_j)\, x_{ij}, \;\; i \in N j \in N \tag{3.10}$$

$$s_i + t_{i,n+i} \leqslant s_{n+i}, \;\; i \in P \tag{3.11}$$

$$e_i \leqslant s_i \leqslant b_i, \;\; i \in N \tag{3.12}$$

$$\max\{0, d_i\} \leqslant q_i \leqslant \min\{Q, Q + d_i\}, \;\; i \in N \tag{3.13}$$

$$x_{ij} \in \{0, 1\}, \quad i \in N j \in N \tag{3.14}$$

Constraints (3.5) and (3.6) ensure that each request may be served at most once and that if the pickup node is visited, then the respective delivery node is also visited. Constraints (3.7) guarantee that the route starts at the origin depot and ends at the destination depot. Constraints (3.8) ensure that if the vehicle arrives at a node must also depart from it. Constraints (3.9) and (3.10) define the change of the time and load variables. Constraints (3.11) ensure that the pickup node $i \in P$ is visited before the respective delivery node $i + n$. Inequalities (3.12) and (3.13) define time windows and capacity constraints. Finally, Constraints (3.14) designate the binary nature of the x_{ij}variables. Note that the above program is non-linear although it can be linearized by standard reformulation techniques.

3.3. Solving the SP

The labeling algorithm for ESPPTWCCPD, also presented in [24], may be used and enhanced to provide all routes that satisfy condition (3.2), *i.e.* columns with positive enhanced profit for the RMP. The labeling algorithm is a progressive dynamic programing algorithm which is based on the principles of the classic Bellman Ford algorithm. The algorithm associates each partial path with a label which contains information about the consumption of resources. Dominance rules are applied during the process to discard non-promising labels.

Interested readers should refer to [17] for an extensive review of Shortest Path Problems with Resource Constraints (SPPRC), to [14] for an application of an exact labeling algorithm on the Elementary Shortest Path Problem with Resource Constraints (ESPPRC), and to [5] for the application of ESPPRC on the OP.

The approach for solving the sub-problem affects computational time significantly. Taking this into account, we have introduced and tested three refinements to the solution of the SP, in an attempt to improve the efficiency of the labeling algorithm, and, hence, the performance of CG.

The first is a novel refinement[2] that consists of a Pruning Criterion that exploits the homogenous cost matrix created by equation (3.3). This criterion predicts which labels will not lead to the destination, while satisfying condition (3.2). According to the extensive review of CG by [8], during the final stages of the labeling procedure a relatively small number of paths have favorable cost (*i.e.* positive enhanced profit for the RMP), as compared to the number of such paths during the first stages of the procedure. It is expected, therefore, that this criterion will be more effective during the final iterations of CG, during which there are more paths that do not satisfy condition (3.2), and, thus, more candidates for pruning.

Considering equation (3.3), the minimum cost that may result by serving the nodes of the graph could be calculated as:

$$\text{minCost} = \sum_{i \in P | \pi_i - p_i < 0} (\pi_i - p_i) + \pi_0 \tag{3.15}$$

[2] Called hereafter PC.

It is obvious that if minCost $\geqslant 0$, then the CG procedure should be terminated. This fact may be applied to each label created by the labeling algorithm. Considering label l; the minimum cost that may result by extending label l may be calculated as follows:

$$\text{minCost}(l) = \sum_{i \in P \setminus U(l) | \pi_i - p_i < 0} (\pi_i - p_i) + \pi_0 \tag{3.16}$$

where $U(l)$ is the set of the so called *unreachable* nodes from label l, which is defined as follows (see also [24]): Given a label l for which: (a) $F(l)$ is the set of visited nodes with respect to label l, (b) $\eta(l)$ is the node of label l, *i.e.* the last node of the associated partial path, and (c) $T(l)$ is the arrival time to node $\eta(l)$, then the set of unreachable nodes with respect to label l may be defined as: $U(l) = F(l) \cup \{i \in P \cup D | T(L) + t_{\eta(l),i} > b_i\}$ thus including the nodes for which the time window constraint is violated. If minCost$(l) \geqslant 0$, then label l may be discarded without extending it further. We added this criterion in the algorithm, and applied it along with the feasibility check during the extension of the labels.

Following the work of [5,14], we imposed a limit on the number of the paths generated during each iteration of the labeling algorithm in order to improve the performance of CG. In our case we terminate the labeling algorithm when 200 paths have been generated. Furthermore, as in the work of [23], we store the labels produced into one bucket per node, creating in total of $2n$ buckets. This requires less effort in order to associate a label with a node, than doing so from a single list.

As a second refinement[3], we have ordered the labels into the bucket using an ascending lexicographic order according to the time consumed, the profit collected and the accumulated load on the vehicle, respectively.

As a final refinement to the labeling algorithm we tested the Limited Discrepancy Search (LDS), which is a heuristic method applied to increase the computational performance of the algorithm. LDS was introduced by [16] and it allows extending a label only to *good neighbor* nodes and not to all feasible nodes. According to [5], good neighbors are nodes with the minimum cost value c_{ij}. The number of the good neighbor nodes is doubled, when the labeling algorithm fails to identify paths with favorable enhanced profit. This process continues if necessary until the number of good neighbors reaches the maximum number of reachable nodes; therefore, optimality is guaranteed.

3.4. Obtaining integer solutions

Since CG deals with the linear relaxation of the MP, to obtain integer solutions for the MP, we apply the BB framework. Specifically, we apply the *branching on arc* strategy of [9], in which the variables are transformed in terms of the arcs (i, j) and the arc with the most fractional value (nearest to 0.5) is selected for branching. Interested readers may refer to [8,13] for more information about branching strategies applied on BP frameworks.

4. Computational study

In this section we investigate how different combinations of the refinements presented above affect the computational time of the Branch and Price algorithm. Specifically, we investigate all combinations among the three refinements: (a) the new pruning criterion to (PC), (b) sorting the lists of labels according to the consumption of resources (Sort), and (c) applying the LDS heuristic (LDS). All algorithms were implemented in the C programing language on a computer running Linux and equipped with a 2.66 GHz Intel Core 2 Quad processor and 2 GB of RAM**.**

For this study we modified the benchmark instances of [19] for the PDPTW. The latter are categorized in three classes according to the geographical distribution of the requests: uniformly distributed requests (*lr*), requests distributed in clusters (*lc*) and mixed distributed requests (*lrc*). In order to adapt to the characteristics of TOPDPTW, and to safeguard that it is not possible to serve all requests, the original benchmark problems

[3] Called hereafter SORT.

were modified by reducing the availability of resources. Specifically, the number of available vehicles is set to 5 instead of 25, the capacity of the vehicles is set to 100 product units instead of 250, the number of requests is reduced to 20, and a computational time limit of 4000 s is imposed. The resulting instances are available at [3].

The first step of this computational study concerned the effects of the refinements (all 2^3 combinations) on the computational time required to solve the linear relaxation of the problem (*i.e.* the root problem of BB). Note that all three refinements concern exactly the solution of this problem. Subsequently, in the second step of the computational study, we considered the effects of these refinements on the computational time required to solve the entire integer program. The following Tables present results with respect to the 27 instances for which at least one refinement combination produced an optimal solution for the integer program within the limit of 4000 s (see Tab. 2).

Table 1 presents the performance of the eight different combinations of refinements in terms of computational time when solving the linear relaxation of the modified instances of [19]. The labels in the second row of the table identify the refinement combinations; for example, (PC.Sort.LDS) is the case in which all refinements have been applied, and (-.-.-) is the case in which none of these refinements is applied. The first column of the Table presents the instance name, while the other columns present the computational time needed per refinement combination to solve the linear relaxation of the problem. The results of the Table indicate that the best performance is provided by applying the refinement combination (PC.-.LDS). The resulting improvement is 45.5% with respect to the base case. This is followed by the performance of (-.-LDS) and (PC.-.-), which result in improvements of 44.3% and 43.6%, respectively.

We also tested whether the differences in computational times as presented in Table 1 are statistically significant. To do so we applied the one tailed paired sample t-test to compare: (a) the computational times of the four combinations which resulted in improvements with (b) the computational time of (-.-.-). The results indicated that the improvements resulting from combinations (PC.-.LDS), (LDS.-.-) and (-.PC.-) are statistically significant with a significance level (p-value) of 0.005, 0.006 and 0.009, respectively.

Table 2 presents the computational time required to obtain the optimal solution for 27 of the modified instances of [19] by applying all combinations of the three (2.3) refinements. In this case as well, the labels in the first row of the Table identify the refinement combinations. The two first columns of the Table represent the instance name and the profit collected for each instance. Subsequently, the odd columns, from the third one and beyond, present the computational times needed to provide the optimal solution for each of the eight refinement combinations; the even columns, from the fourth one and beyond, present the number of branch and bound nodes solved in order to reach the optimal solution. The last two rows of Table 2 present two values of computational time per refinement combination: (a) the average computational time only for the instances solved within the limit of 4000 s, and (b) the average computational time for all instances; in this last average we assumed that the computational time for each unsolved instance is equal to 4000 s (a conservative assumption). The third from last row presents the number of problems solved within the 4000 s limit.

In terms of the number of instances solved to optimality, combination (-.-.-) has the lowest performance (23 of 27), while (PC.-.-) and (-.Sort.LDS) had the best performance (26 of 27). In terms of the average computational time considering all instances, all refinements perform better than the base case (-.-.-). Specifically, (PC- -) provided an improvement of 67.5%, while combinations (PC.Sort.-) and (PC.-.LDS) improve the average computational time by 48.8% and 45.1%, respectively. Comparing the results of Tables 1 and 2 it seems that the performance of (PC.-.-) and (PC.-.LDS) refinement combinations improve significantly the solution of both the relaxed and integer problems.

5. Applications of TOPDPTW

An obvious application for the TOPDPTW is generating daily routes for a Pick-up and Delivery system in cases in which there are no sufficient resources to serve all requests within the allotted period. This application may be addressed directly using the models and methods of Sections 2 and 3. In fact, the experimental study of Section 4 focused precisely on such problems. A second application is in system design; that is, determine the most appropriate fleet configuration in order to satisfy the expected demand at some level of service while

TABLE 1. Computational time to obtain the solution of the linear relaxation per instance and per refinement combination for instances of 20 requests.

	Computational time (s)							
Instance	PC.Sort.LDS	PC.-.LDS	-.Sort.LDS	-.-.LDS	PC.Sort.-	PC.-.-	-.Sort.-	-.-.-
lr101	0.010	0.009	0.010	0.009	0.008	0.008	0.009	0.007
lr102	0.040	0.070	0.040	0.060	0.050	0.060	0.069	0.060
lr103	1.119	1.228	1.064	0.919	1.756	1.719	1.694	1.564
lr104	0.981	0.781	1.009	0.901	1.357	0.718	1.242	0.783
lr105	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010
lr106	0.078	0.093	0.077	0.092	0.104	0.078	0.101	0.074
lr107	0.590	0.565	16.700	0.538	22.020	2.780	21.670	2.650
lr108	674.065	1.699	1454.730	2.195	2.798	2.777	3.348	2.636
lr109	3.549	2.083	2.974	1.410	1.060	0.620	4.344	2.843
lr110	0.377	0.372	0.382	0.351	0.409	0.235	0.662	0.810
lr111	9.820	6.395	10.173	6.619	7.487	4.221	7.474	5.843
lr112	0.567	0.631	0.558	0.638	1.114	0.430	1.133	4.320
lc102	0.030	0.168	0.050	0.162	0.112	0.093	0.109	0.650
lc103	0.060	0.156	0.060	0.169	1.860	0.160	1.850	0.160
lc105	0.020	0.024	0.030	0.045	0.059	0.063	0.059	0.060
lc106	0.045	0.026	0.034	0.015	0.010	0.009	0.010	0.100
lc107	0.061	0.044	0.060	0.046	0.084	0.052	0.085	0.052
lc108	0.050	0.064	0.020	0.069	0.020	0.111	0.020	0.110
lc109	0.054	0.026	0.053	0.063	0.264	0.215	0.268	0.208
lrc101	0.022	0.015	0.220	0.026	0.021	0.020	0.021	0.023
lrc102	0.562	0.105	0.105	0.111	0.404	0.416	0.628	0.823
lrc103	0.030	0.020	0.030	0.030	0.040	0.060	0.050	0.070
lrc104	0.040	0.050	0.079	0.051	0.254	0.233	0.259	0.233
lrc105	0.131	0.070	0.050	0.092	0.081	0.082	0.082	1.420
lrc106	0.125	0.260	0.132	0.201	0.202	0.119	0.276	0.312
lrc107	0.108	0.147	0.106	0.142	0.063	0.057	0.062	0.560
lrc108	0.156	0.130	0.156	0.227	0.102	0.082	0.101	0.840
Average time	25.656	0.564	55.145	0.563	1.546	0.571	1.690	1.008

earning maximum profit. A fleet configuration is defined by the pair (Q, V) that is, the quantity of products that a vehicle may carry and the number of vehicles available. We study this second application below.

The performance issues that are relevant to the design problem under study involve: (a) quality of service, (b) profit collected, and (c) fleet utilization. The quality of service may be evaluated by the ratio of the requests served *versus* the total requests (see Tab. 3). In the same fashion, the profit collected may be evaluated as a percentage of the maximum profit available. Note that quality of service and profit collected are different metrics, since profit is in general not equal among customers. Fleet utilization may be characterized by the two last Key Performance Indicators (KPIs) of Table 3.

Regarding the fleet utilization KPIs and the differences between VU and CU, one may consider the following example. Let a single vehicle serve two requests, and let the quantity corresponding to each request be equal to the vehicle capacity. Also let the transportation time of each request be equal to 10% of the total shift time. In this case, $\text{VU} = 2$, while $\text{CU} = 0.2$.

Our analysis used thirteen problems of [19] selected among the sets of random uniformly distributed requests (*lr*), and mixed requests (*lrc*). From each problem we selected the first 30 requests and we considered 24 different fleet configurations resulting in a total of $13 \times 24 = 312$ instances. Specifically, the 24 fleet configurations were

Table 2. Performance of the combinations of refinements for selected instances with 20 requests.

		PC.Sort.LDS		PC.-.LDS		-.Sort.LDS		-.-.LDS		PC.Sort.-		PC.-.-		-.Sort.-		-.-.-	
Instance	Profit	CPU[1]	BN[2]	CPU[1]	BN[2]	CPU[1]	BN[2]	CPU[1]	BN[2]	CPU[1]	BN[2]	CPU[1]	BN[2]	CPU[1]	BN[2]	CPU[1]	BN[2]
lr101	185	0.01	1	0.01	1	0.01	1	0.01	1	0.01	1	0.01	1	0.01	1	0.01	1
lr102	259	0.04	1	0.07	1	0.04	1	0.08	1	0.06	2	0.05	1	0.06	2	0.06	1
lr103	232	–	–	331.53	2	3176.77	4	998.19	4	730.78	2	1023.03	3	689.81	2	882.83	3
lr104	247	191.85	18	103.82	13	293.19	17	–	–	46.84	6	71.28	20	103.23	23	–	–
lr105	217	0.01	1	0.01	1	0.01	1	0.01	1	0.01	1	0.01	1	0.01	1	0.01	1
lr106	300	1.51	20	–	–	1.74	22	–	–	–	–	–	–	–	–	–	–
lr107	281	–	–	1.82	2	16.7	1	4.87	2	37.69	2	2.78	1	21.67	1	2.65	1
lr108	276	2967.2	4	1129.34	3	2609.84	3	–	–	–	–	325	1	3786.41	6	579.42	1
lr109	259	3.07	2	3.52	2	3.06	2	1.41	3	1.06	1	0.62	1	5.43	4	3.68	2
lr110	291	–	–	–	–	58.71	98	45.42	80	29.84	39	80.43	160	55	88	25.8	52
lr111	266	12.92	2	10.33	4	10.54	2	11.16	4	24.9	13	15.14	6	27.32	11	15.02	6
lr112	292	75.94	4	65.66	8	259.76	10	77.95	11	29.94	4	0.43	1	103.9	5	23.53	5
lc102	370	0.03	1	5.54	17	0.05	1	3.56	14	0.68	6	0.84	3	0.69	6	0.82	3
lc103	350	0.06	1	2.51	20	0.06	1	2.59	20	1.86	1	0.16	1	1.85	1	0.16	1
lc105	400	0.02	1	0.04	2	0.03	1	0.05	2	6.82	18	0.56	9	7.98	19	0.81	14
lc106	360	0.08	2	0.04	2	0.06	3	0.44	9	0.01	1	0.11	3	0.01	1	0.21	3
lc107	370	1.58	9	0.1	6	1.48	9	0.12	6	2.14	23	1.21	8	2.04	23	1.27	8
lc108	420	0.4	8	1.59	12	0.02	1	2.11	15	0.02	1	7.32	10	0.02	1	4.54	10
lc109	370	0.13	10	0.07	4	0.15	10	0.12	4	48.2	14	164.61	26	56.05	14	36.08	27
lrc101	369	0.03	2	0.02	2	0.03	2	0.03	2	0.04	6	0.03	6	0.05	6	0.04	6
lrc102	370	2.16	34	0.99	11	1.78	17	1.27	11	1.69	30	0.96	14	–	–	–	–
lrc103	356	0.03	1	0.02	1	0.03	1	0.03	1	0.04	1	0.06	1	0.05	1	0.07	1
lrc104	316	0.04	1	0.05	1	–	–	0.05	1	3.61	15	0.83	14	3.2	15	0.89	14
lrc105	399	3.69	60	0.07	1	0.05	1	0.09	1	2.54	10	1.79	44	10.36	98	–	–
lrc106	370	1.05	5	0.26	1	0.43	2	1.4	10	0.5	3	0.31	3	2.17	21	2.45	20
lrc107	450	0.98	8	0.92	13	1.01	8	1.02	13	0.03	2	0.03	2	0.03	2	0.03	2
lrc108	390	55.46	38	6.31	16	50.31	10	21.58	12	27.94	52	18.91	23	38.54	24	20.41	26
Instances Solved[3]		24/27		25/27		26/27		24/27		25/27		26/27		25/27		23/27	
Average time[4]		138.26		66.59		249.46		48.90		39.89		66.02		196.64		69.60	
Global average time[5]		567.34		357.95		388.37		487.91		333.23		211.72		478.37		651.88	

[1]CPU is the computational time in seconds, [2]BN is the number of BB nodes solved to obtain the optimal solution,[3]within the 4000 s limit, [4]for instances solved within the 4000 s limit, [5]for all instances, assuming that the computational time for the problems not solved within the limit is 4000 s.

Table 3. Key Performance Indicators (KPIs) for TOPDPTW.

Category	**KPI**	**Description**
Quality of Service	QOS	Number of requests served *vs.* the total requests
Profit Collected	PC	Profit collected *vs.* maximum available profit
Fleet Utilization	VU	Quantity of products actually transported by the entire fleet throughout the operations *vs.* the total fleet capacity
	CU	It is the ratio of (a) Quantity of products transported times the transportation time, over (b) Total fleet capacity times the shift time

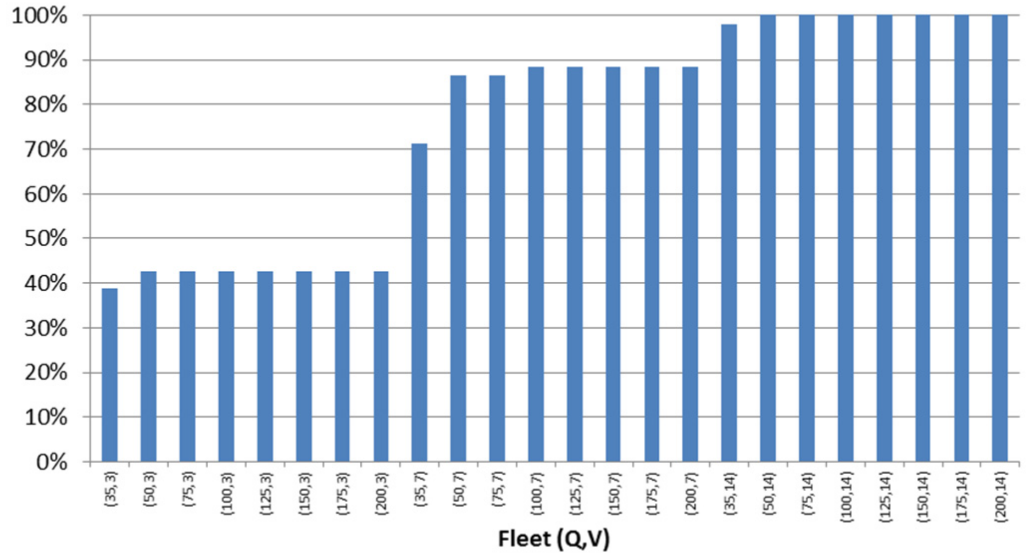

FIGURE 1. QOS *vs.* fleet configuration.

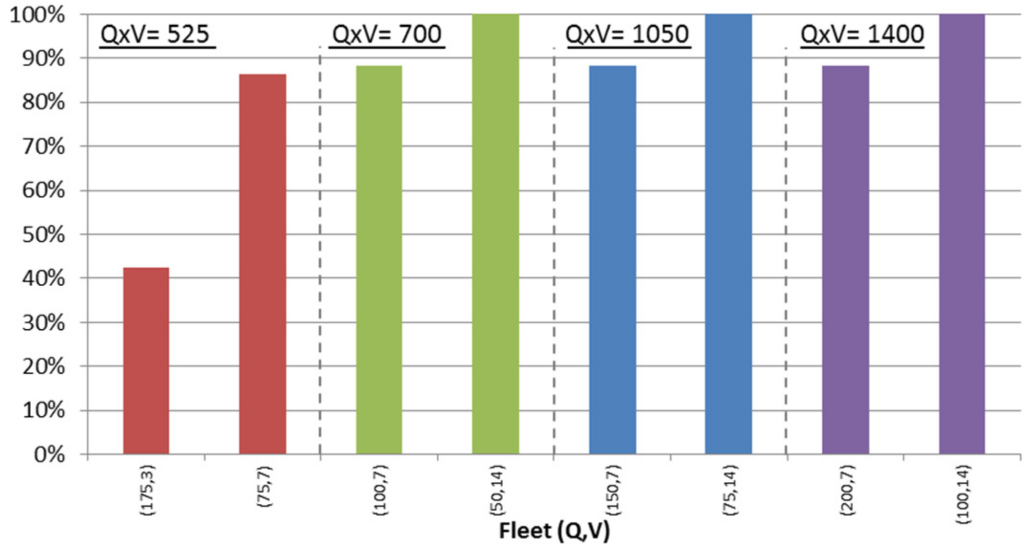

FIGURE 2. QOS for fleet configurations *vs.* $\mathbf{Q} \times \mathbf{V}$.

generated by all combinations between $V \in \{3, 7, 14\}$ and $Q \in \{35, 50, 75, 100, 125, 150, 175, 200\}$. All instances were solved to optimality by applying the Branch and Price framework of Section 3. The results shown in the remainde of this Section for each configuration are the average values of the KPIs over all 13 instances.

Figure 1 illustrates the average values of the Quality of Service (QOS) for the 24 different fleet configurations. The latter are grouped according to the number of vehicles V, and within each V group, the QOS values are provided in ascending order of the vehicle capacity Q. It is clear that QOS improves as the number of available vehicles increases, while the capacity of the vehicles also affects QOS, but to a lesser degree. This latter observation is apparent when comparing fleet configurations with the same total capacity $Q \times V$. Figure 2

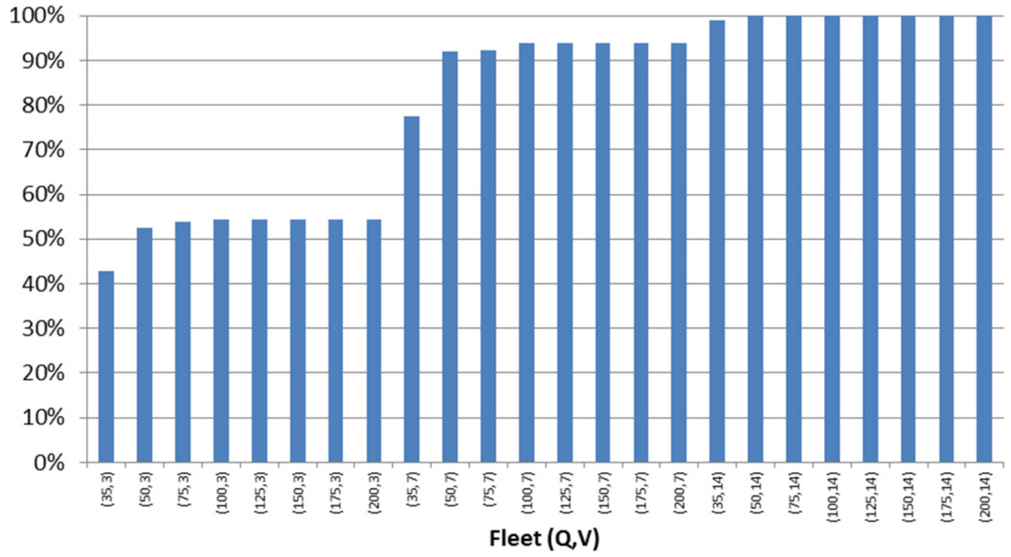

FIGURE 3. PC *vs.* fleet configuration.

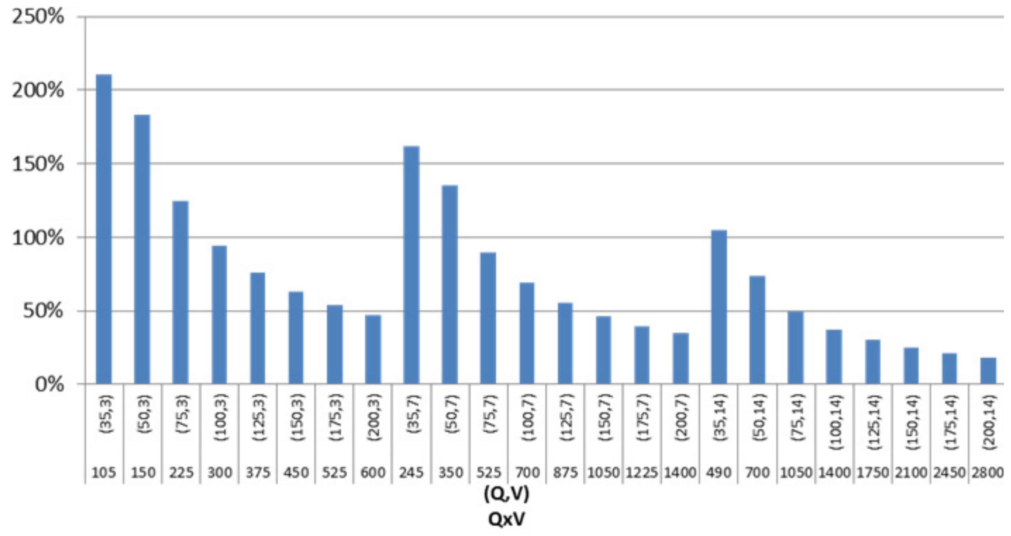

FIGURE 4. VU *vs.* total fleet capacity.

shows four such cases; in all four cases the configuration with the higher number of vehicles provides improved values of QOS.

Considering the effect of the vehicle capacity in QOS, a significant jump exists between fleets with the same number of vehicles when Q increases from 35 to 50. It seems that beyond $Q = 50$ the remaining problem constraints, such as the time windows, do not allow significant performance improvement with increasing Q.

A similar picture is shown in Figure 3, which presents the Profit Collected (PC) for the 24 fleet configurations sorted in the same way as in Figure 1. As expected, the PC values are higher or equal to the respective values of QOS, since the profitable requests are served with higher priority.

Figure 4 presents the variation of the first Vehicle Utilization metric (VU). In this case, the x-axis represents the total fleet capacity $Q \times V$. Note that the values of VU may exceed 100% since a vehicle may load and

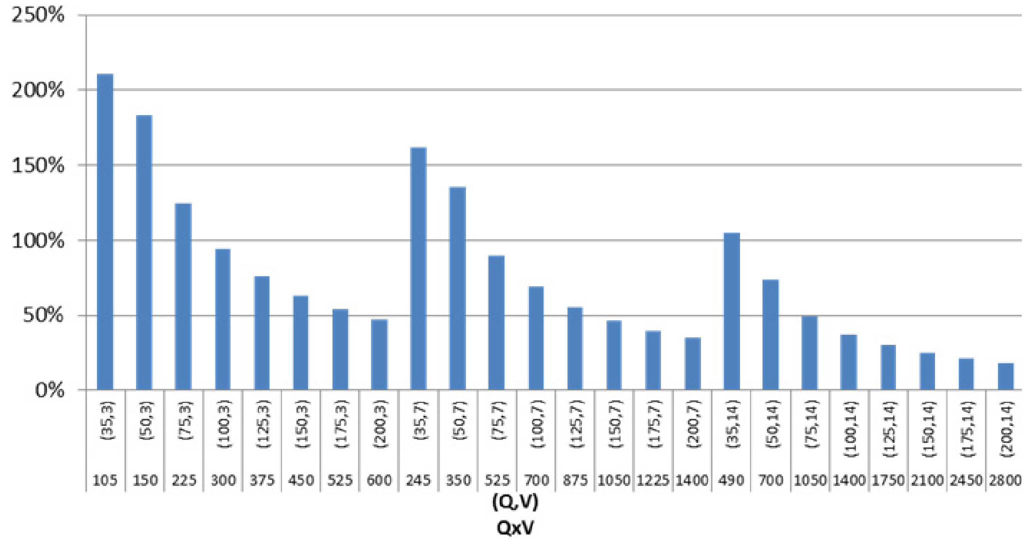

FIGURE 5. CU *vs.* total fleet capacity.

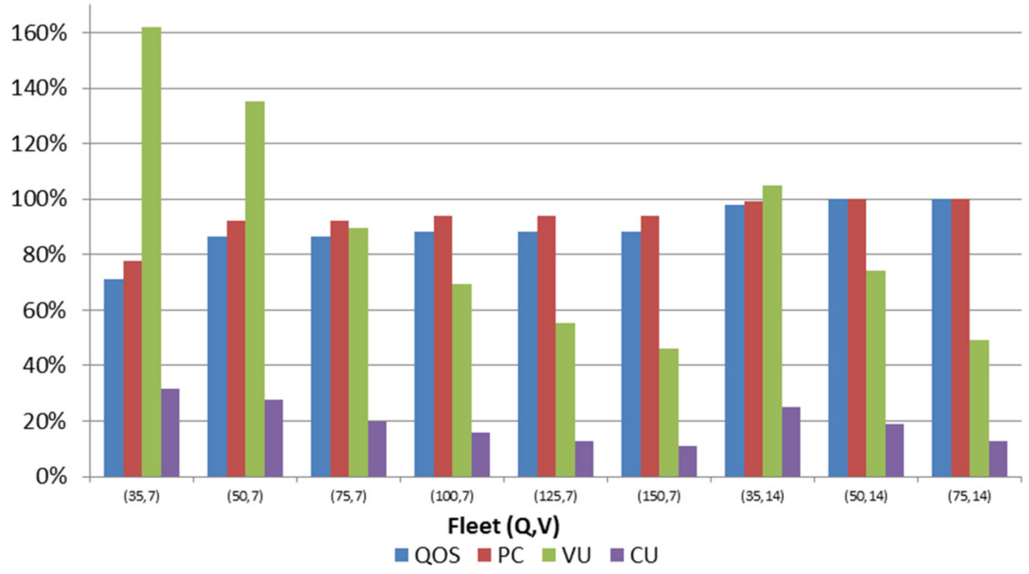

FIGURE 6. Performance of qualified fleet configurations.

unload products several times during its single trip (route). As expected, higher values of VU (*i.e.* higher fleet utilization) are observed for fleets with relatively smaller values of $Q \times V$. Furthermore, although there is a smooth drop of VU as $Q \times V$ increases, it is also obvious that fleet configurations with relatively lower Q are better utilized than those of similar $Q \times V$, but of lower V. An illustrative case of this later observation is the case of fleet configurations (75,3) and (35,7), both of which result in similar total capacity. In addition to higher utilization, fleet configuration (35,7) which includes the higher number of vehicles, was able to serve more requests, and thus, transport more products.

Finally, Figure 5 shows the values of the second Vehicle Utilization metric (CU) using the same x-axis as the one in Figure 4. In this case, lower values of CU are acceptable considering that a vehicle may, in some

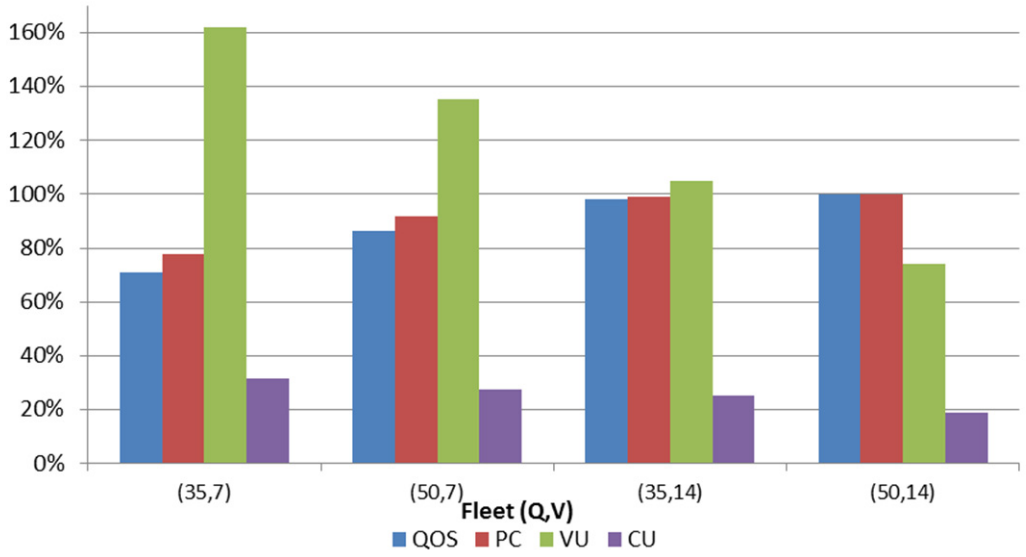

FIGURE 7. Performance of qualified fleet configurations.

cases, travel less loaded on its way to pick up a request. Similarly to the results for the VU, higher values of CU indicate that the vehicle loading space is better utilized. These results imply that fleets of large size (higher values of V or Q) are utilized less efficiently than fleets of smaller size.

From the above results (Figs. 1–5) it seems that for the service level and the profit collected, the number of vehicles is more significant than the vehicle capacity. However, vehicle capacity is significant in fleet utilization, since vehicles with high capacity may not be efficiently utilized.

Based on the above analysis, the appropriate fleet configuration may be determined for a certain client base following the steps below:

Step 1. Define an acceptance limit for each of the aforementioned KPIs.
Step 2. Define the likely demand scenarios. Solve the TOPDPTW for all scenarios and all configurations (V, Q) to determine the performance metrics and their average values.
Step 3. Identify for which configurations (V, Q) the KPI limits of Step 1 are met by the average values of Step 2.
Step 4. Discard fleet configurations for which there is at least another fleet configuration that achieves superior values in all KPIs, *i.e.* discard dominated fleet configurations.
Step 5. Select the fleet configuration with the minimum acquisition and operational cost.

As an example, for Step 1 consider the following KPI limits: (a) QOS $>$ 70%, (b) PC $>$ 70%, (c) VU $>$ 40%, (d) CU $>$ 10%. In Step 2, consider that the likely scenarios are the 13 instances studied in the previous Section. In this example, the average performance of only nine fleet configurations surpasses the thresholds set of Step 1. Figure 7 displays the performance of these nine acceptable configurations sorted in ascending order of PC values.

Fleet configurations (50,14) and (75,14) serve all requests and collect all available profit, although the fleet utilization is lower for configuration (75,14). Thus, configuration (50,14) dominates configuration (75,14). Similarly, configurations (75,7), (100,7), (125,7) and (150,7) are dominated by configurations (50,14) and (35,14), and thus, may be discarded (Step 4). Figure 7 presents all qualified non-dominated configurations among which the less costly one (in the total cost sense) may be selected (Step 5).

6. A multi-objective perspective

Cases in which the ranges of both the number of vehicles V and the vehicle capacity Q are wide may benefit from multi-objective optimization methods, such as the epsilon-constraint schemes (see [18]). Considering the current case, the multiple objectives would be to maximize each of the four KPIs and to minimize the acquisition and operational costs. This would require solving a number of constrained single-objective sub-problems. Note that the Branch and Price framework presented in Section 3 is designed to maximize the profit collected, and thus, should be modified appropriately to address the different single objective sub-problems.

Another approach would be to consider a triple-objective problem that aims to: (a) maximize the total profit, (b) minimize the fleet size, and (c) minimize the fleet capacity. In this case the branch and price framework could be directly embedded to an epsilon-constraint scheme, since both the fleet size and the fleet capacity are constraints of the problem. The latter would reduce the computational burden by reducing the number of fleet configurations cases to be solved by the Branch and Price framework. Again the appropriate fleet configuration could be determined for a certain client base following the aforementioned Steps 1 to 5.

7. Conclusions

In this paper, we introduced the TOPDPTW which combines the Team Orienteering Problem and the Pick-up and Delivery Problem with Time Windows and Capacity Constraints. We modeled the TOPDPTW accordingly, and presented an exact solution algorithm to solve it. For the latter, we proposed a new pruning technique to accelerate the solution of the related sub-problem, and used two additional accelerating refinements from the literature. To test the effects of these three refinements, we conducted extensive computational experiments. The results of our study indicated that for the linear relaxation of the problem applying the proposed pruning technique (PC), or LDS improve the performance of the algorithm significantly. While with respect to the optimal solution it seems that all combinations of the three refinements improve the performance of the algorithm. Specifically applying the proposed pruning technique (PC) improves the computational time significantly (by 67%).

We performed an extensive parametric study to determine the effects of vehicle capacity and fleet size to fleet performance while serving a certain request base. The results of this study indicate that although fleets of high total capacity may result to higher profit collected, as expected, their overall utilization may be significantly low. Furthermore, it seems that in terms of service level and the profit collected, the number of vehicles is more significant than the vehicle capacity. Finally, vehicle capacity should be sized correctly, since vehicles with unnecessarily high capacity may be underutilized significantly.

Given that fleet sizing should balance multiple objectives, we presented a practical method to determine the most suitable fleet, in terms of number of vehicles V and vehicle capacity Q for a given request base.

Acknowledgements. This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program ”Education and Lifelong Learning” of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund. The authors thank Mr. K. Tavlaridis – Gyparakis for the fruitful discussions regarding this topic.

References

[1] C. Archetti, D. Feillet, A. Hertz and M. Speranza, The capacitated team orienteering and profitable tour problems. *J. Oper. Res. Soc.* **60** (2009) 831–842.

[2] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance, Branch-and-price, Column generation for solving huge integer programs. *Oper. Res.* **46** (1998) 316–329.

[3] Benchmark Instances for: TOPDPTW and applications in fleet sizing. Retrieved from: http://labs.fme.aegean.gr/deopsys/topptw_instances (2014).

[4] G. Berbeglia, J. Cordeau, I. Gribkovskaia and G. Laporte, Static pickup and delivery problems, a classification scheme and survey. *TOP* **15** (2007) 1–37.

[5] S. Boussier, D. Feillet and M. Gendreau, An exact algorithm for the team orienteering problem. *Oper. Res.* **5** (2007) 211–230.

[6] I. Chao, B. Golden and E. Wasil, The team orienteering problem. *Eur. J. Oper. Res.* **88** (1996) 464–474.

[7] G. Desaulniers, J. Desrosiers, I. Ioachim, M. Solomon, F. Soumis and D. Villeneuve, A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems. Fleet Management and Logistics, edited by T.G. Crainic and G. Laporte. Kluwer, Norwell, MA (1998) 57–93.

[8] G. Desaulniers, J. Desrosiers and M. Solomon, Column Generation. Springer Science and Business Media, Inc., New York (2005).

[9] J. Desrochers, J. Desrosiers and M. Solomon, A new optimization algorithm for the vehicle routing prolem with time windows. *Oper. Res.* **40** (1992) 342–354.

[10] J. Desrosiers, M. Dumas, M. Solomon and F. Soumis, Time Constrained Routing and Scheduling. *Handbooks in Operations Research and Management Science, 8: Network Routing*, edited by M. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser. North-holland, Amsterdam (1995) 35–139.

[11] G. Dikas and I. Minis, Scheduled Paratransit Transport Systems. *Transp. Res. Part B* **67** (2014) 18-34.

[12] Y. Dumas, J. Desrosiers and F. Soumis, The pickup and delivery problem with time windows. *Eur. J. Oper. Res.* **54** (1991) 7–22.

[13] D. Feillet, A tutorial on column generation and branch-and-price for the vehicle routing problems. *4OR: A Quarterly J. Oper. Res.* **8** (2010) 407–424.

[14] D. Feillet, P. Dejax, M. Gendreau and C. Gueguen, An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints, Application to Some Vehicle Routing Problem. *Networks* **44** (2004) 217–229.

[15] B. Golden, L. Levy and R. Vohra, The orienteering problem. *Naval Res. Logistics* **34** (1987) 307–318.

[16] W. Harvey and M. Ginsberg, Limited Discrepancy Search. *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada*. Morgan Kaufmann Publishers Inc. (1995) 607–615.

[17] S. Irnich and G. Desaulniers, Shortest Path Problems with Resource Constraints. In Column Generation, edited by G. Desaulniers, J. Desrosiers and M.M. Solomon. Springer (2005) 33–65.

[18] M. Laumanns, L. Thiele and E. Zitzler, An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *Eur. J. Oper. Res.* **169** (2006) 932–942.

[19] H. Li and A. Lim, A MetaHeuristic for the Pickup and Delivery Problem with Time Windows. In *13th International Conference on Tools with Artificial Intelligence, Dallas* (2001).

[20] M. Lubbecke and J. Desrosiers, Selected Topics in Column Generation. *Oper. Res.* **53** (2005) 1007–1023.

[21] S.N. Parragh, K.F. Doerner and R.F. Hartl, A survey on pickup and delivery problems. Part I, Transportation between customers and depots. *J. für Betriebswirtschaft* **58** (2008a) 21–51.

[22] S.N. Parragh, K.F. Doerner and R.F. Hartl, A survey on pickup and delivery problems Part II, Transportation between pickup and delivery locations. *J. für Betriebswirtschaft* **58** (2008b) 81–117.

[23] G. Righini and M. Salani, New dynamic programming algorithms for the elementary shortest path problem with resource constraints. *Networks* **51** (2008) 155–170.

[24] S. Ropke and J.F. Cordeau, Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transp. Sci.* **43** (2009) 267–286.

[25] J. Schonberger, H. Kopfer and D.C. Mattfeld, A combined approach to solve the pickup and delivery selection problem. *Oper. Res. Proc.* **2002** (2003) 150–155.

[26] C.K. Ting and X.L. Liao, The selective pickup and delivery problem: Formulation and a memetic algorithm. *Int. J. Prod. Econ.* **141** (2013) 199–211.

[27] P. Toth and D. Vigo, The Vehicle Routing Problem, *SIAM Monogr. Discr. Math. Appl.* Philadelphia (2002).

[28] T. Tsiligirides, Heuristic methods applied to orienteering. *J. Oper. Res. Soc.* **37** (1984) 351–367.

[29] P. Vansteenwegen, W. Souffruau and D. Van Oudheusden, The orienteering problem, A survey. *Eur. J. Oper. Res.* **209** (2011) 1–10.