Submitted to *INFORMS Journal on Computing*
manuscript (Please, provide the manuscript number!)

The Vehicle Routing Problem with Floating Targets: Formulation and Solution Approaches

Claudio Gambella
DEI, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy, claudio.gambella@unibo.it

Joe Naoum-Sawaya
Ivey Business School, 1255 Western Road, London, ON Canada, N6G 0N1, jnaoumsa@uwaterloo.ca

Bissan Ghaddar
IBM Research, IBM Technology Campus Building 3, Mulhuddart, Dublin 15, Ireland, bghaddar@ie.ibm.com

This paper addresses a generalization of the vehicle routing problem in which the pick-up locations of the targets are non-stationary. We refer to this problem as the vehicle routing problem with floating targets and the main characteristic is that targets are allowed to move from their initial home locations while waiting for a vehicle. This problem models new applications in drone routing, ridesharing, and logistics where a vehicle agrees to meet another vehicle or a customer at a location that is away from the designated home location. We propose a Mixed Integer Second Order Cone Program (MISOCP) formulation for the problem, along with valid inequalities for strengthening the continuous relaxation. We further exploit the problem structure using a Lagrangian decomposition and propose an exact branch-and-price algorithm. Computational results on instances with varying characteristics are presented and the results are compared to the solution of the full problem using CPLEX. The proposed valid inequalities reduce the computational time of CPLEX by up to 30% on average while the proposed branch-and-price is capable of solving instances where CPLEX fails in finding the optimal solution within the imposed time limit.

Key words: Vehicle Routing, Moving Targets, Lagrangian Decomposition, Branch-and-Price.

1. Introduction

Since the introduction of the Vehicle Routing Problem (VRP) in Dantzig and Ramser (1959), several variations of the VRP have been addressed in the literature. Such routing problems cover a range of practical situations and are generally challenging for decision makers mainly due to the difficulty in developing efficient algorithms for finding optimal

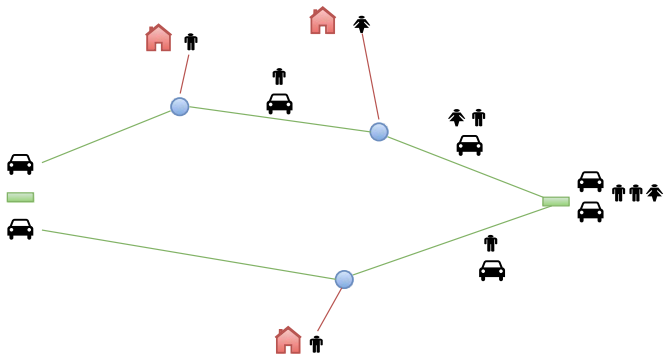

Figure 1 **A ride-sharing system with floating pick-up locations**

or even sub-optimal solutions, especially when modeling real-life applications (Toth and Vigo 2014).

The focus of this paper is to formulate and solve a dynamic variant of VRP, which we refer to as the Vehicle Routing Problem with Floating Targets (VRPFT). The problem seeks to determine a set of vehicle routes in order to visit a set of target points that can freely move in the Euclidean plane. VRPFT has several applications in practice. One such application arises in ridesharing where the target points are individuals requiring a means of transport for reaching a common destination, such as employees of the same company (e.g., Aïvodji et al. (2015), Bruck et al. (2015), Bit-Monnot et al. (2013) and Varone and Aissat (2015)). Traditionally, a common assumption is that vehicles will pick-up the individuals from their home locations which sometimes creates undesired detours for the drivers, while in practice, the individuals might meet the car at a more convenient location midway on the route to the common destination (Figure 1). Another area of practical applications is related to target tracking such as missions conducted by Unmanned Aerial Vehicles in military and civilian contexts for example for surveillance, defense, security, reconnaissance, weather monitoring, and pollutant estimation (e.g., Sundar and Rathinam (2013), Mallick et al. (2013)) and aerial refueling (e.g., Barnes et al. (2004) and Thomas et al. (2014)).

Contributions of this paper. The main contributions of this paper are (a) a mixed integer second order cone program for the vehicle routing problem with floating targets, (b) a branch-and-price approach to address this problem, and (c) the derivation of valid inequalities coupled with a computationally efficient implementation that is capable of solving problems of reasonable size.

Outline of this paper. Following this introductory section, Section 2 provides a literature review. The VRPFT along with a variant that assumes that the locations can move only in a fixed direction are formulated in Section 3. Valid cuts for strengthening the formulations are derived in Section 4. Section 5 presents the proposed branch-and-price approach. Section 6 discusses the implementation details and presents the computational results. Finally, Section 7 provides a brief conclusion and future research directions.

2. Literature Review

The vehicle routing problem is a generalization of the Traveling Salesman Problem (TSP) which has been studied extensively in the literature. The TSP requires the determination of a minimum-cost Hamiltonian cycle graph (Laporte 1992, Gutin and Punnen 2002). A variant of the TSP, in which the nodes of the graph are moving during the planning horizon, is typically referred to as moving-target TSP, kinetic TSP, or non-stationary TSP. Hammar and Nilsson (2002) considered the moving target TSP where the nodes are moving at a constant speed on a fixed line in the Euclidean plane and proved the existence of a polynomial time approximation scheme for the problem where the targets are sharing the same direction and speed. Helvig et al. (2003) studied several variants of the moving-target TSP and proposed an exact polynomial algorithm based on dynamic programming for the case where all the locations are on a line. In addition, the authors addressed the moving-target TSP with resupply, in which a vehicle is required to head back to the depot after intercepting each target, under the restriction of targets moving along lines passing from the depot. Jiang et al. (2005) considered the problem in which the targets are moving with a constant speed in a straight line in a two-dimensional space. A genetic algorithm is also proposed and computational results are presented on a randomly generated test set with 10 target nodes.

VRP refers to the cases where multiple vehicles are available. Asahiro et al. (2006) analyzed a VRP in which multiple interceptor vehicles, robots in their case, are moving on straight track lines and targets are traveling at a fixed speed. Polynomial time algorithms and proof of NP-hardness are given for problem variants in which either the number of intercepted targets or the number of interceptor vehicles are variables to be optimized. Choubey (2013) considered the situation in which the moving targets are traveling at constant speed in a given direction and proposed a genetic algorithm and provided computational results on a random test set with a maximum of 10 target locations. More recently,

Stieber et al. (2015) addressed the moving-target VRP where the targets are moving at a fixed speed in a linear trajectory in a multi-dimensional space. The authors proposed a formulation for the problem in a time-extended graph and a time discretization step that leads to a very large number of variables is proposed to reformulate the problem as a mixed integer linear program. A heuristic is then proposed to produce solutions in a reasonable computational time and results are provided for instances with up to 36 targets and 3 vehicles.

Besides moving targets, other extensions of VRP have also been studied in the literature to model practical situations. For instance, time windows may be imposed on a vehicle reaching a particular location and several exact and heuristic solution approaches have been proposed (Dumas et al. 1995, Mingozzi et al. 1997, Gendreau et al. 1998). Other model variants include conditions that require a vehicle to visit a pick-up location before proceeding to the associated drop-off location (Baldacci et al. 2011). Such models are often used for passenger transportation such as the dial-a-ride problem which deals with the routing of the vehicles and the sequencing of passenger pick-up and drop-off (Baldacci et al. 2004, Cordeau 2006). Reyes et al. (2016) introduced another variant of VRP where the delivery points can be chosen from a discrete set of locations given capacity and time windows constraints with the objective of minimizing the vehicles travel time. Tailored heuristic algorithms outperform CPLEX especially as the size of the instance increases.

In this paper we address the vehicle routing problem with floating targets where the vehicles can meet the target clients at locations that may be away from their home. Particularly, we assume that each target can travel within a maximum speed limit to meet a vehicle. The vehicles are also assumed to be traveling within a maximum speed and we assume a common destination for all the targets. The VRPFT is formulated next.

3. Problem Formulation

This section presents the VRPFT problem formulation. We first propose the general case where the targets can move in any direction and then present a specialized formulation for the case where the targets can move according to a straight line.

3.1. General Case

Given a set of n targets to be picked up by a fleet of K vehicles, each with a capacity Q and a maximum vehicle speed S^V, the aim of the vehicle routing problem with floating targets

is to determine the minimum cost vehicle routes starting from the depot O and ending at the drop-off location D where all the targets are picked up by a vehicle at a convenient location which we refer to as a meeting point. At time $t=0$, each target j has a starting home location $H_j \in \mathbb{R}^2$, while the starting location of all vehicles is depot O. Furthermore, each target is allowed to move from the designated home location at a maximum speed S_j^T.

To formulate the mathematical model, the following binary decision variables are introduced:

$$x_{ij}^k = \begin{cases} 1 \text{ if target } j \text{ is the } i\text{-th target visited by vehicle } k, \\ 0 \text{ otherwise.} \end{cases}$$

$$y_k = \begin{cases} 1 \text{ if vehicle } k \text{ is used,} \\ 0 \text{ otherwise.} \end{cases}$$

The vehicles origin O is designated as target $j=0$ while the destination node D for the vehicles that are used is designated as target $j=n+1$. The following continuous decision variables are also introduced to determine the meeting points between the targets and the vehicles and the time required to reach the meeting points. The continuous variables are

m_{ki}^V : coordinates in $\mathbb{R}^2$ of the i-th meeting point for vehicle k,
m_j^T : coordinates in $\mathbb{R}^2$ of the pick-up point of target j,
t_{ki}^V : time required by vehicle k to reach the i-th meeting point from $i-1$,
t_j^T : time required by target j to reach the designated meeting point.

VRPFT can then be formulated as follows

$$[\text{OP}]: \min \quad \sum_{k=1}^{K}\sum_{i=1}^{Q+1} t_{ki}^V \tag{1}$$

$$\text{s.t} \quad \frac{\|m_{ki}^V - m_{k,i-1}^V\|}{S^V} \le t_{ki}^V \qquad \forall\, i=1,\dots,Q+1, \forall\, k=1,\dots,K \tag{2}$$

$$\frac{\|m_j^T - H_j\|}{S_j^T} \le t_j^T \qquad \forall\, j=1,\dots,n \tag{3}$$

$$m_{ki}^V \ge m_j^T - C^M(1-x_{ij}^k) \qquad \forall\, i=1,\dots,Q, \forall\, j=1,\dots,n+1, \quad \forall\, k=1,\dots,K \tag{4}$$

$$m_{ki}^V \le m_j^T + C^M(1-x_{ij}^k) \qquad \forall\, i=1,\dots,Q, \forall\, j=1,\dots,n+1, \quad \forall\, k=1,\dots,K \tag{5}$$

$$m^V_{k,Q+1} = y_k \cdot D + (1 - y_k) \cdot O \quad \forall\ k = 1, \ldots, K \tag{6}$$

$$\sum_{i'=1}^{i} t^V_{ki'} \geq t^T_j - C^T_j (1 - x^k_{ij}) \quad \forall\ i = 1, \ldots, Q, \forall\ j = 1, \ldots, n, \quad \forall\ k = 1, \ldots, K \tag{7}$$

$$\sum_{k=1}^{K} \sum_{i=1}^{Q} x^k_{i,j} = 1 \quad \forall\ j = 1, \ldots, n \tag{8}$$

$$\sum_{j'=1}^{n} x^k_{i,j'} \geq x^k_{i+1,j} \quad \forall\ i = 1, \ldots, Q-1, \forall\ j = 1, \ldots, n, \quad \forall\ k = 1, \ldots, K \tag{9}$$

$$y_k = \sum_{j=1}^{n} x^k_{1,j} \quad \forall\ k = 1, \ldots, K \tag{10}$$

$$\sum_{j=1}^{n+1} x^k_{i,j} = y_k \quad \forall\ i = 2, \ldots, Q, \forall\ k = 1, \ldots, K \tag{11}$$

$$m^V_{k0} = O \quad \forall\ k = 1, \ldots, K \tag{12}$$

$$m^T_{n+1} = D \tag{13}$$

$$t^T_j \geq 0 \quad \forall j = 1, \ldots, n \tag{14}$$

$$t^V_{ki} \geq 0 \quad \forall\ i = 1, \ldots, Q+1, \forall\ k = 1, \ldots, K \tag{15}$$

$$x^k_{i,j} \in \{0, 1\} \quad \forall\ i = 1, \ldots, Q, \forall\ j = 1, \ldots, n+1, \quad \forall\ k = 1, \ldots, K, (i,j) \neq (1, n+1) \tag{16}$$

$$y_k \in \{0, 1\} \quad \forall\ k = 1, \ldots, K. \tag{17}$$

The objective function (1) minimizes the total travel time of the vehicles. Constraints (2) ensure that the total travel time for a vehicle between two consecutive locations is no less than the travel time that is required given a maximum speed S^V. We note that the distances are computed via the Euclidean norm $\|\cdot\|$ ($\|\cdot\| := \|\cdot\|_2$). Constraints (3) ensure that the travel time for each target between the home location and the pick-up location is no less than the travel time that is required given a maximum speed S^T_j. Constraints (4) and (5) indicate that if target j is the i-th target to be visited by vehicle k then vehicle k and target j should meet at the same location $m^T_j = m^V_{ki}$. Hence Constraints (4) and (5) enforce the following logical conditions using the "big-M" coefficient C^M:

$$m^T_j = m^V_{ki} \ \text{ if } x^k_{ij} = 1 \ \forall\ i = 1, \ldots, Q, \forall\ k = 1, \ldots, K, \forall\ j = 1, \ldots, n+1.$$

The constant $C^M = (C_x^M, C_y^M)$ in (4) and (5) can be safely defined as $C_x^M = X_{max} - X_{min}$, $C_y^M = Y_{max} - Y_{min}$, where $X_{max}, X_{min}, Y_{max}, Y_{min}$ are limits on the meeting point locations. Constraints (6) indicate that if a vehicle is used then it should end the tour at the drop-off location D otherwise the vehicle remains at the depot O. Constraints (7) impose the synchronization between the vehicles and targets by not allowing a vehicle to depart before the arrival of the target through enforcing the following logical conditions using the "big-M" coefficients C_j^T:

$$\sum_{i'=1}^{i} t_{ki'}^V \geq t_j^T \quad \text{if } x_{i,j}^k = 1 \ \forall \ i = 1, \ldots, Q, \forall \ k = 1, \ldots, K, \forall \ j = 1, \ldots, n.$$

The constant C_j^T in (7) can be safely defined as $C_j^T = \frac{\sqrt{(X_{max}-X_{min})^2-(Y_{max}-Y_{min})^2}}{S_j^T} + \bar{w}_j$ where $\bar{w}_j$ is the maximum allowable waiting time for target j. The requirement that each target is picked up by exactly one vehicle is expressed in Constraints (8). Constraints (9) impose the order of the targets that are intercepted by each vehicle and that $n+1$ is the last target visited by a used vehicle. We note that these constraints are a stronger version of the valid constraints:

$$\sum_{j=1}^{n} x_{i,j}^k \geq \sum_{j=1}^{n} x_{i+1,j}^k \ \forall \ i = 1, \ldots, Q-1, \forall \ k = 1, \ldots, K.$$

Constraints (10) follow from the definition of the y variables while Constraints (11) are necessary to express that a vehicle which left the depot should proceed to pick-up a target. Constraints (12) state that each vehicle is at the depot O at time $t = 0$ while Constraint (13) indicate that D is the final destination of each route. Finally, Constraints (14)–(17) define the decision variables of the problem. We note that variables $x_{1,n+1}^k$ are not defined in the model since no vehicle will leave the depot directly to the drop-off location. Furthermore, the definition of the x variables in terms of the pick-up sequence has the advantage of naturally enforcing the maximum capacity Q on the fleet of vehicle by only using the variables $x_{i,j}^k$ with $i \leq Q$.

Model (1)–(17) is a Mixed-Integer Second Order Cone Program (MISOCP). It consists of a linear objective function, linear constraints, and second order cone constraints ((2) and (3)). In order to solve the model with an optimization solver such as CPLEX, it is generally required to rewrite the second order cone constraints in the standard quadratic form

$$\|x\| \leq t \quad x \in \mathbb{R}^n, t \in \mathbb{R}.$$

For that, the auxiliary variables η^V_{ki}, τ^V_{ki}, η^T_j, and τ^T_j are introduced and Constraints (2) and (3) are replaced by

$$\begin{aligned}
&\eta^V_{ki} = m^V_{ki} - m^V_{k,i-1} && \forall\ i = 1,\dots,Q+1, \forall\ k = 1,\dots,K\\
&\tau^V_{ki} = S^V t^V_{ki} && \forall\ i = 1,\dots,Q+1, \forall\ k = 1,\dots,K\\
&\eta^T_j = m^T_j - H_j && \forall\ j = 1,\dots,n\\
&\tau^T_j = S^T_j t^T_j && \forall\ j = 1,\dots,n\\
&\|\eta^V_{ki}\| \le \tau^V_{ki} && \forall\ i = 1,\dots,Q+1, \forall\ k = 1,\dots,K\\
&\|\eta^T_j\| \le \tau^T_j && \forall\ j = 1,\dots,n.
\end{aligned}$$

Model (1)–(17) assumes the general VRPFT where the targets can meet the vehicle in any location that falls within the bounds $X_{max}, X_{min}, Y_{max}$, and Y_{min}. The next section presents the particular case where the targets are only allowed to move in a predefined direction.

3.2. Targets Moving along a Fixed Line

For the special case where the targets can only move according to a straight line, model (1)–(17) can be modified to include constraints to limit the trajectories of the targets. Particularly, the assumption is that each target j can move according to a direction vector d_j and a scalar λ_j parametrizes the travel trajectory. The limitations in determining the meeting points are then imposed using the following equations:

$$m^T_j - H_j = \lambda_j\ d_j\ \forall\ j = 1,\dots,n, \tag{18}$$

and Constraints (3) reduce to the linear conditions

$$\frac{\|d_j\|\lambda_j}{S^T_j} \le t^T_j\ \forall\ j = 1,\dots,n. \tag{19}$$

VRPFT with fixed line moving directions is then formulated as

$$\begin{aligned}
\min\quad & (1)\\
\text{s.t } & (2), (4)-(17),\\
& (18), (19)\\
& \lambda_j \ge 0 \qquad \forall j = 1,\dots,n
\end{aligned}$$

where the conic Constraints (3) in the general VRPFT are replaced by the linear Constraints (18) and (19). Since the resulting model contains less conic constraints, the VRPFT with fixed line moving directions is computationally easier to solve than the general VRPFT as demonstrated in the computational results section.

Finally we note that the proposed models make the assumption that the targets and the vehicles can travel freely in the Euclidean plane and are allowed to wait at the meeting points. Potentially constraints that limit such assumptions might be needed to limit the free travel of targets and vehicles in order to accommodate the specificities of the particular applications.

4. Valid Cuts

In this section, we propose a set of valid cuts for the VRPFT. These cuts are valid for model (1)–(17) but can strengthen the continuous relaxation and consequently speed up the solution of the problem.

4.1. Final Destination

In model (1)–(17), the final destination D for the vehicle routes is enforced using Constraints (9) and (11). This condition can be also enforced using the following constraints:

$$x^k_{i,n+1} \leq x^k_{i+1,n+1} \;\forall\; i = 2,\ldots,Q-1, \forall\; k = 1,\ldots K, \tag{20}$$

which indicate that if the i-th target for vehicle k is the destination node, then that vehicle will remain at the destination node by forcing all subsequent locations for vehicle k to be $n+1$. Furthermore the constraints

$$\sum_{i'=i+1}^{Q}\sum_{j=1}^{n} x^k_{i',j} + (Q-i)x^k_{i,n+1} \leq (Q-i)y_k \;\forall\; i = 2,\ldots,Q-1, \forall\; k = 1,\ldots K, \tag{21}$$

are also valid and indicate that if the i-th target for vehicle k is the destination node, then the vehicle will not serve any other target.

THEOREM 1. *Constraints* (21) *are valid for VRPFT.*

Proof: From Constraints (11), we have that the following equalities hold $\forall\; i = 2,\ldots,Q-1$, $\forall\; k = 1,\ldots K$:

$$\sum_{i'=i+1}^{Q}\sum_{j=1}^{n} x^k_{i',j} = \sum_{i'=i+1}^{Q}\left(y_k - x^k_{i',n+1}\right) = (Q-i)y_k - \sum_{i'=i+1}^{Q} x^k_{i',n+1}.$$

From (20) it follows that

$$(Q-i)x^k_{i,n+1} \leq \sum_{i'=i+1}^{Q} x^k_{i',n+1} \qquad \forall\ i=2,\dots,Q-1, \forall\ k=1,\dots K,$$

hence (21) are valid. □

4.2. Homogeneous Fleet

Taking advantage of the assumption of identical vehicles, valid inequalities can be added to model (1)–(17). Particularly, the following constrains are valid

$$y_1 = 1, \tag{22}$$

$$y_{k+1} \leq y_k\ \forall\ k=1,\dots,K-1. \tag{23}$$

Since at least one vehicle must be used, then Constraint (22) forces vehicle 1 to be used without loss of generality while Constraints (23) indicate that each vehicle k cannot be used unless vehicles $1,\dots,k-1$ are used as well.

4.3. Vehicles Order

In addition to Constraints (22)–(23), we propose families of cuts inspired by the conditions that are described in Fischetti et al. (1995) and used in Coelho and Laporte (2013). These cuts arise from the fact that given a feasible solution of VRPFT, it is always possible to construct an equivalent solution in which each vehicle k is allowed to pick-up target j only if vehicle $k-1$ picks up a target with an index that is smaller than j and thus the following cuts are valid:

$$\sum_{i=1}^{Q-1} x^k_{i,j} \leq \sum_{i'=1}^{Q-1}\sum_{j'<j} x^{k-1}_{i',j'}\ \forall\ j=2,\dots,n, \forall\ k=2,\dots,K. \tag{24}$$

THEOREM 2. *Constraints* (24) *are valid for VRPFT.*

Proof: Consider a feasible solution for VRPFT which also satisfies (22)–(23). Since the vehicles are identical, then there exists a permutation σ on the set of vehicles $\{1,\dots,K\}$ such that

$$t(\sigma(1)) \leq \cdots \leq t(\sigma(K)),$$

where $t(k) := \min\{j \in \{1,\dots,n\} \mid \exists\ i \in \{1,\dots,n\} : x^k_{ij} = 1\}$ for $k \in \{1,\dots,K\}$. Without loss of generality, we can reorder the vehicles according to permutation σ such that

$$t(1) \leq \cdots \leq t(K)$$

and Constraints (24) are therefore valid. □

4.4. Unserved Targets

Valid cuts can also be derived based on the number of targets not served by any vehicle with index not greater than k which is given by

$$L_k = \underbrace{n - \Big(Q - \sum_{i=2}^{Q} x^1_{i,n+1}\Big)}_{\text{targets not served by vehicle 1}} - \sum_{k'=2}^{k} \underbrace{\Big(Qy_{k'} - \sum_{i=2}^{Q} x^{k'}_{i,n+1}\Big)}_{\text{targets served by vehicle } k'} \qquad \forall\ k = 1, \ldots, K-1.$$

Particularly, the following cuts are valid

$$n - \Big(Q - \sum_{i=2}^{Q} x^1_{i,n+1}\Big) - \sum_{k'=2}^{k} \Big(Qy_{k'} - \sum_{i=2}^{Q} x^{k'}_{i,n+1}\Big) \leq (n-k)y_{k+1} \quad \forall\ k = 1, \ldots, K-1, \tag{25}$$

$$n - \Big(Q - \sum_{i=2}^{Q} x^1_{i,n+1}\Big) - \sum_{k'=2}^{k} \Big(Qy_{k'} - \sum_{i=2}^{Q} x^{k'}_{i,n+1}\Big) \geq y_{k+1} \quad \forall\ k = 1, \ldots, K-1. \tag{26}$$

THEOREM 3. *Constraints* (25)–(26) *are valid for VRPFT.*

Proof: L_k varies between 0 and $n-k$. If L_k is strictly greater than 0 then k vehicles are not sufficient for serving all targets and therefore vehicle $k+1$ has to be used which can be imposed by setting $L_k \leq (n-k)\ y_{k+1}\ \forall\ k = 1, \ldots, K-1$ and thus (25) are valid. Otherwise, if $L_k = 0$ then all targets can be picked up by the first k vehicles and hence vehicle $k+1$ is not needed. This can be imposed by setting $L_k \geq y_{k+1}\ \forall\ k = 1, \ldots, K-1$ and thus (26) are valid. □

As discussed in Section 6.5, the addition of of the proposed valid cuts reduces the computational time for solving VRPFT. However, the problem remains challenging to solve. The following section presents a branch-and-price approach that exploits the structure of the problem to achieve a better computational performance.

5. Branch-and-Price

In this section, we describe the different components of the branch-and-price solution approach. We start first by applying a Lagrangian relaxation to [OP].

5.1. Lagrangian Relaxation

Lagrangian relaxation is a well known algorithm that has been used as a solution approach for several complex optimization problems (Fisher 2004, Frangioni 2005). Particularly, Lagrangian relaxation has been used to address various variants of the vehicle routing problem (Desrochers et al. 1992, Fisher et al. 1997, Kohl and Madsen 1997). The common

approach is to relax the assignment constraints of targets to vehicles. For VRPFT, the Lagrangian relaxation is applied to Constraints (8) using multipliers μ_j to obtain the following subproblem:

$$\text{[SP]: } v_{SP(\mu)} = \min \sum_{k=1}^{K}\sum_{i=1}^{Q+1} t_{ki}^V - \sum_{k=1}^{K}\sum_{i=1}^{Q}\sum_{j=1}^{n} \mu_j x_{i,j}^k + \Big[\sum_{j=1}^{n} \mu_j\Big]$$
$$\text{s.t} (2)-(7), (9)-(17).$$

Problem [SP] is decomposable into K identical single vehicle subproblems

$$\text{[SSP]: } v_{SSP(\mu)} = \min \quad \sum_{i=1}^{Q+1} t_i^V - \sum_{j=1}^{n}\sum_{i=1}^{Q} \mu_j x_{i,j} \tag{27}$$
$$\text{s.t} \quad \frac{\|m_i^V - m_{i-1}^V\|}{S^V} \le t_i^V \qquad \forall\ i=1,\dots,Q+1 \tag{28}$$
$$\frac{\|m_j^T - H_j\|}{S_j^T} \le t_j^T \qquad \forall\ j=1,\dots,n \tag{29}$$
$$m_i^V \ge m_j^T - C_M(1-x_{ij}) \quad \forall\ i=1,\dots,Q,\ \forall\ j=1,\dots,n+1 \tag{30}$$
$$m_i^V \le m_j^T + C_M(1-x_{ij}) \quad \forall\ i=1,\dots,Q,\ \forall\ j=1,\dots,n+1 \tag{31}$$
$$m_{Q+1}^V = y\cdot D + (1-y)\cdot O \tag{32}$$
$$\sum_{i'=1}^{i} t_{i'}^V \ge t_j^T - C_T(1-x_{ij}) \quad \forall\ i=1,\dots,Q,\ \forall\ j=1,\dots,n \tag{33}$$
$$\sum_{j'=1}^{n} x_{i,j'} \ge x_{i+1,j} \qquad \forall\ j=1,\dots,n,\ \forall\ i=1,\dots,Q-1 \tag{34}$$
$$y = \sum_{j=1}^{n} x_{1,j} \tag{35}$$
$$\sum_{j=1}^{n+1} x_{i,j} = y \qquad \forall\ i=2,\dots,Q \tag{36}$$
$$m_0^V = O \tag{37}$$
$$m_{n+1}^T = D \tag{38}$$
$$t_j^T \ge 0 \qquad \forall j=1,\dots,n \tag{39}$$
$$t_i^V \ge 0 \qquad \forall\ i=1,\dots,Q+1 \tag{40}$$
$$x_{i,j} \in \{0,1\} \qquad \forall\ i=1,\dots,Q, \forall\ j=1,\dots,n+1,\ (i,j)\neq(1,n+1) \tag{41}$$
$$y \in \{0,1\}. \tag{42}$$

Note that if the fixed line moving direction case of VRPFT is assumed, then in that case, Constraints (29) are replaced by Constraints (18) and (19).

The value of the Lagrangian bound $v_{LR(\mu)}$ is

$$v(LR(\mu)) = K \cdot v(SSP(\mu)) + \sum_{j=1}^{n} \mu_j,$$

and the best Lagrangian bound is given by $v_{LR} = \max_{\mu \in \mathbb{R}^n} v_{LR(\mu)}$. Given H, the set of feasible solutions of [SSP], the best Lagrangian bound can be found by solving

$$\max_{\mu} \Big\{ K \cdot \min_{h=1,\ldots,H} \Big\{ \sum_{i=1}^{Q+1} t^V_{i,(h)} - \sum_{j=1}^{n} \sum_{i=1}^{Q} \mu_j x_{i,j,(h)} \Big\} + \sum_{j=1}^{n} \mu_j \Big\},$$

which is equivalent to the Lagrangian Master Problem:

$$[\text{LMP}] : \max \sum_{j=1}^{n} \mu_j + K\theta \tag{43}$$

$$\text{s.t} \sum_{j=1}^{n} \sum_{i=1}^{Q} \mu_j x_{i,j,(h)} + \theta \leq \sum_{i=1}^{Q+1} t^V_{i,(h)} \qquad \forall\ h \in H. \tag{44}$$

Denoting with α_h the dual variables associated with Constraints (44), the dual of [LMP] is the Dantzig-Wolfe master problem:

$$\begin{aligned}
[\text{DMP}] : \min & \sum_{h \in H} \alpha_h \sum_{i=1}^{Q+1} t^V_{i,(h)} \\
\text{s.t.} & \sum_{h \in H} \alpha_h = K \\
& \sum_{h \in H} \alpha_h \sum_{i=1}^{Q} x_{i,j,(h)} = 1, \quad \forall\ j = 1, \ldots, n \\
& \alpha_h \geq 0, \quad \forall\ h \in H.
\end{aligned}$$

Since the set H is not known beforehand, the Lagrangian bound v_{LR} is calculated iteratively where a relaxed version of [LMP] is solved to obtain the Lagrange multipliers μ_j and then [SSP] is solved leading to a new cut in [LMP]. Note that a new cut in [LMP] is equivalent to a new column in [DMP]. Solving the relaxed version of [LMP] provides an upper bound on the optimal Lagrangian bound v_{LR} while the optimal solution of [SSP] provides a lower bound $v_{LR(\mu)}$. The iterative approach is stopped when the upper and lower bounds are equivalent.

5.2. Tightening the Lagrangian bound

A main advantage for the application of the presented Lagrangian relaxation is the decomposition of the problem into [SSP] which is computationally less challenging to solve than [OP]. However, due to the relaxation, the resulting Lagrangian bound is a lower bound to [OP]. The Lagrangian bound can be tightened by the addition of constraints that are redundant to [OP] but cut a part of the feasible solutions of [SSP]. Particularly the inequalities

$$\sum_{i=1}^{Q} x_{i,j} \leq 1 \quad \forall\ j = 1, \ldots, n \tag{45}$$

are valid to [OP] and improve the Lagrangian bound when added to [SSP]. Constraints (45) indicate that each target j should be visited by a vehicle at most once.

The Lagrangian bounds are used in a branch-and-bound framework to obtain an optimal solution to [OP]. As discussed next, constraints that maintain the decomposable structure of [SP] are used for branching.

5.3. Branching strategy

At each node of the branch-and-bound tree, the columns matrix corresponding to [DMP] is

$$\mathcal{X} = \begin{bmatrix} \sum_{i=1}^{Q} x_{i,1,1} & \sum_{i=1}^{Q} x_{i,1,2} & \ldots & \sum_{i=1}^{Q} x_{i,1,|H|} \\ \sum_{i=1}^{Q} x_{i,2,1} & \sum_{i=1}^{Q} x_{i,2,2} & \ldots & \sum_{i=1}^{Q} x_{i,2,|H|} \\ \ldots & \ldots & \ldots & \ldots \\ \sum_{i=1}^{Q} x_{i,n,1} & \sum_{i=1}^{Q} x_{i,n,2} & \ldots & \sum_{i=1}^{Q} x_{i,n,|H|} \end{bmatrix}$$

where each column h is associated with a value $\overline{\alpha}_h$ that is given by the optimal solution of [DMP] at that node. Branching is needed when two columns h_1 and h_2 are associated with a fractional α_h and contain the pattern

	$\ldots$	h_1	$\ldots$	h_2	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
j_1	$\ldots$	1	$\ldots$	1	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
j_2	$\ldots$	1	$\ldots$	0	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

in the columns matrix $\mathcal{X}$. The challenge is in devising branching constraints that can maintain the decomposable structure of the subproblem when added. We thus propose branching constraints inspired from the Ryan-Foster branching rules (Ryan and Foster 1981) which have been successfully used in branch-and-price algorithms with separable subproblems (Mehrotra and Trick 1996).

The branching constraints are based on forcing two targets to be assigned to the same vehicle versus two different vehicles and can be written as follows

$$\sum_{i=1}^{Q} x_{i,j_1}^k = \sum_{i=1}^{Q} x_{i,j_2}^k \ \forall\ k = 1, \ldots, K, \tag{46}$$

$$\sum_{i=1}^{Q} x_{i,j_1}^k + \sum_{i=1}^{Q} x_{i,j_2}^k \leq 1 \ \forall\ k = 1, \ldots, K. \tag{47}$$

Constraints (46) impose that if targets j_1 and j_2 should be picked up by the same vehicle while Constraints (47) forbid the two targets from being picked up by the same vehicle.

In our implementation, the columns in $\mathcal{X}$ are ordered with respect to the value $\overline{\alpha}_h$ and those with the highest value are checked first. If a branching pattern is found, then $\mathcal{X}$ is split into two parts; the first containing all the columns that satisfy Constraints (46) while the second containing all the columns that satisfy Constraints (47). The children nodes are then created and the master problem [DMP] at each node is warm started with the columns that satisfy the branching constraints that are added to the corresponding subproblem [SSP].

6. Computational Testing

6.1. Implementation Details

In this section, we evaluate the computational performance of the proposed branch-and-price algorithm which we implemented in C. Each optimization problem is solved using CPLEX 12.6.1 with default settings. The branch-and-price tree is explored using a depth-first search strategy where the left node is associated to Constraints (46) and the right node is associated to Constraints (47). When solving the subproblem [SSP], Constraints (30), (31), (33), and (34) are declared as lazy constraints since they constitute a relatively large set of constraints with the majority of them likely not to be binding at optimality. The branch-and-price approach is compared to the direct solution of problem [OP] with Constraints (4), (5), (7), and (9) declared as lazy constraints. The results are conducted on a Xeon E3-1220 processor clocked at 3.10 GHz with 16 GB RAM. A 7200 seconds time limit is imposed after which the best available solution is reported.

6.2. Test Instances

A set of 18 randomly generated instances with varying characteristics is used for the evaluation (the instances can be obtained from `http://www.vrp-rep.org/datasets/item/2016-0012.html` and `http://www.vrp-rep.org/datasets/item/2016-0013.html`). In particular, the number of targets is varied between 10 and 20 and the number of vehicles is varied between 3 and 5. The vehicle capacities are set to $\lceil \frac{n}{K} \rceil + 2$. The home locations of the targets are randomly generated in the Euclidean space centered at $(0, 0)$ and with width 50 and height 100. The initial locations of the vehicles is set at $(-20, 0)$ and the common destination/drop-off location is set to $(20, 0)$. The speed of the vehicles is also randomly generated from the uniform distribution $\mathcal{U}[2, 3]$ while the speed of the targets is randomly generated from the uniform distribution $\mathcal{U}[0.1, 1]$. The VRPFT with fixed line moving directions is tested on the same 18 instances, where a random direction vector is introduced for each target.

6.3. Initial Greedy Heuristic

At the start of the branch-and-price algorithm, a greedy heuristic is used to initialize the incumbent. For that, a feasible solution is easily obtained by assuming that the pick-up order is greedily assigned to the vehicle with the closest target ordered first. A sketch of the greedy heuristic is as follows.

1: **Initialization:** Assume that all the vehicles are currently located at the starting location O and are empty;
2: Let S be the set of unassigned targets and initialize S as the set of all targets;
3: Let D be a distance vector where $D(j)$ denotes the distance between O and the home location of an unassigned target j and initialize D accordingly;
4: Let k be the current vehicle and initialize k to 1;
5: Let i be the number of targets served by vehicle k and initialize i to 0.
6: **while** S is not empty **do**
7: Select target j with the minimum $D(j)$;
8: Fix $x^k_{i+1,j} = 1$ in the VRPFT formulation;
9: Update $S = S - \{j\}$;
10: Eliminate target j from D;
11: Update $i = i + 1$;

12: **if** the capacity limit of vehicle k is reached (i.e., $i = Q$) **then**
13: Update $k = k + 1$;
14: Set $i = 0$.
15: **end if**
16: **end while**
17: Solve the resulting VRPFT with all the binary variables fixed to obtain the meeting locations.

Because the heuristic assigns all targets to vehicles, the resulting VRPFT model does not contain any binary variable and is thus a second order cone problem. The greedy heuristic is therefore used to initialize the incumbent with a feasible solution. The incumbent is then updated when a better feasible solution is found in the branch-and-price tree.

6.4. Results

This section presents two sets of results, the first comparing the performance of CPLEX to the proposed branch-and-price algorithm in solving the general VRPFT (1)–(17) while the second set of results presents the same comparison for the VRPFT with fixed line moving directions that is presented in Section 3.2. We note that the results that are presented in this section do not include the valid inequalities that are discussed in Section 4. The impact of the valid inequalities is evaluated next in Section 6.5.

Tables 1 and 2 illustrate the performance of the proposed branch-and-price compared to the performance of CPLEX for VRPFT and VRPFT with fixed line moving directions, while Tables 3 and 4 provide the details of the performance of each of the components of the branch-and-price algorithm. Particularly, the following details are provided:

- Instance Name: p_t_v_c indicates an instance with t targets, v vehicles, and c vehicle capacity.
- Upper Bound: Best upper bound found by the algorithm.
- Lower Bound: Best lower bound found by the algorithm.
- Gap (%): Percentage gap calculated as $\frac{\text{Upper Bound}-\text{Lower Bound}}{\text{Upper Bound}}$.
- CPU Time (s): Total computational CPU time in seconds.
- Number of Nodes: Total number of nodes explored by the branch-and-price algorithm.
- Iterations: Total number of iterations at the root node.
- CPU Master Problem: Total CPU time spent on solving the master problems at the root node.

- CPU Subproblem: Total CPU time spent on solving the subproblems at the root node.
- Avg. Iterations: Average number of iterations at each of the nodes of the branch-and-price tree excluding the root node.
- Avg. CPU Master Problem: Average Total CPU time spent on solving the master problems at each of the nodes of the branch-and-price tree excluding the root node.
- Avg. CPU Subproblem: Average Total CPU time spent on solving the subproblems at each of the nodes of the branch-and-price tree excluding the root node.

As shown in Tables 1 and 2, the proposed branch-and-price algorithm outperforms CPLEX for the majority of the tested instances. For VRPFT (Table 1), branch-and-price solved 12 instances to optimality within the time limit while CPLEX solved 5 instances. For the VRPFT with fixed line moving directions (Table 2), branch-and-price solved 14 instances to optimality while CPLEX solved 7 instances. For the instances where the time limit is reached, we notice that branch-and-price tends to find a better lower bound than CPLEX thus highlighting the advantage of column generation (p_18_3_8, p_18_4_7, p_20_4_7 and p_20_5_6 in Table 1 and p_18_3_8, p_18_4_7, p_20_3_9 and p_20_5_6 in Table 2). On the other hand, CPLEX found a better upper bound than the proposed branch-and-price for 5 instances (p_16_3_8, p_18_3_8, p_20_3_9, and p_20_4_7 in Table 1 and p_20_3_9 in Table 2). The better upper bound in such cases is expected due to the capabilities of CPLEX in finding good feasible solutions through heuristics. Branch-and-price also found a better

	Branch-and-Price				CPLEX			
Instance Name	Upper Bound	Lower Bound	Gap(%)	CPU Time (s)	Upper Bound	Lower Bound	Gap(%)	CPU Time (s)
p_10_3_6	72.69	72.69	0%	**183.84**	72.69	72.69	0%	229.61
p_10_4_5	62.14	62.14	0%	**70.38**	62.14	62.14	0%	449.27
p_10_5_4	71.71	71.71	0%	**161.28**	71.71	71.71	0%	542.51
p_12_3_6	76.91	76.91	0%	**377.43**	76.91	76.91	0%	579.70
p_12_4_5	91.36	91.36	0%	**1209.43**	91.36	84.05	*8%	>7200
p_12_5_5	80.08	80.08	0%	**532.05**	80.08	80.08	0%	4107.24
p_14_3_7	84.47	84.47	0%	**3841.94**	84.47	84.47	0%	6961.36
p_14_4_6	97.62	97.62	0%	**5131.43**	103.97	90.45	13%	>7200
p_14_5_5	80.18	80.18	0%	**377.12**	80.18	73.77	*8%	>7200
p_16_3_8	181.58	78.08	57%	>7200	**85.84**	**78.97**	8%	>7200
p_16_4_6	75.85	75.85	0%	**6635.41**	76.71	75.18	2%	>7200
p_16_5_6	91.68	91.68	0%	**6084.35**	96.7	64.79	33%	>7200
p_18_3_8	217.47	**97.86**	55%	>7200	**132.58**	86.18	35%	>7200
p_18_4_7	**80.68**	**75.03**	7%	>7200	87.64	60.47	31%	>7200
p_18_5_6	76.18	76.18	0%	**1132.36**	78.73	59.05	25%	>7200
p_20_3_9	193.88	50.41	74%	>7200	**104.74**	**67.03**	36%	>7200
p_20_4_7	216.63	**106.15**	51%	>7200	**127.02**	83.83	34%	>7200
p_20_5_6	**128.35**	**114.23**	11%	>7200	133.24	97.27	27%	>7200

* indicates that the upper bound is the optimal solution however the gap is due to the lower bound.

Table 1 Computational results comparing branch-and-price and CPLEX for the general VRPFT (1)–(17).

	Branch-and-Price				CPLEX			
Instance Name	Upper Bound	Lower Bound	Gap(%)	CPU Time (s)	Upper Bound	Lower Bound	Gap(%)	CPU Time (s)
p_10_3_6	85.51	85.51	0%	**31.40**	85.51	85.51	0%	154.09
p_10_4_5	72.06	72.06	0%	**20.56**	72.06	72.06	0%	104.20
p_10_5_4	83.33	83.33	0%	**25.95**	83.33	83.33	0%	99.32
p_12_3_6	94.55	94.55	0%	**60.57**	94.55	94.55	0%	111.76
p_12_4_5	109.34	109.34	0%	**137.00**	109.34	109.34	0%	3510.18
p_12_5_5	91.55	91.55	0%	**79.52**	91.55	91.55	0%	2669.57
p_14_3_7	101.43	101.43	0%	**361.97**	101.44	101.44	0%	2676.76
p_14_4_6	105.19	105.19	0%	**1109.85**	113.99	101.45	11%	>7200
p_14_5_5	89.31	89.31	0%	**104.39**	89.97	82.77	8%	>7200
p_16_3_8	100.63	100.63	0%	**3035.48**	100.63	93.59	*7%	>7200
p_16_4_6	97.91	97.91	0%	**1577.55**	111.85	82.77	26%	>7200
p_16_5_6	105.37	105.37	0%	**401.00**	124.07	98.02	21%	>7200
p_18_3_8	**167.57**	**140.76**	16%	>7200	198.62	101.3	49%	>7200
p_18_4_7	**85.96**	**84.24**	2%	>7200	99.84	69.89	30%	>7200
p_18_5_6	97.93	97.93	0%	**1333.93**	110.27	77.19	30%	>7200
p_20_3_9	248.08	**81.87**	67%	>7200	**159.38**	74.91	53%	>7200
p_20_4_7	128.94	128.94	0%	**4750.29**	169.41	94.87	44%	>7200
p_20_5_6	**137.36**	**127.74**	7%	>7200	187.12	104.79	44%	>7200

* indicates that the upper bound is the optimal solution however the gap is due to the lower bound.

Table 2 Computational results comparing branch-and-price and CPLEX for VRPFT with fixed line direction.

		Root Node			Other Nodes†		
Instance Name	Number of Nodes	Iterations	CPU Master Problem	CPU Subproblem	Avg. Iterations	Avg. CPU Master Problem	Avg. CPU Subproblem
p_10_3_6	17	29	<0.01	60.18	5	0.02	7.71
p_10_4_5	1	37	0.10	70.26	-	-	-
p_10_5_4	41	30	<0.01	25.26	4	<0.01	3.40
p_12_3_6	1	63	<0.01	377.39	-	-	-
p_12_4_5	147	27	<0.01	62.71	4	<0.01	7.85
p_12_5_5	67	39	<0.01	88.7	5	<0.01	6.72
p_14_3_7	1	109	0.02	3841.8	-	-	-
p_14_4_6	157	51	<0.01	532.1	6	<0.01	29.48
p_14_5_5	15	50	<0.01	143.99	7	<0.01	16.65
p_16_3_8	1	97	0.11	>7200	-	-	-
p_16_4_6	83	66	<0.01	1182.2	7	<0.01	66.49
p_16_5_6	119	51	0.03	484.03	7	<0.01	47.45
p_18_3_8	1	67	0.01	>7200	-	-	-
p_18_4_7	39	62	<0.01	2275.28	7	<0.01	134.43
p_18_5_6	1	81	0.01	1132.23	-	-	-
p_20_3_9	1	39	<0.01	>7200	-	-	-
p_20_4_7	1	103	0.02	>7200	-	-	-
p_20_5_6	187	77	0.01	2254.2	5	<0.01	27.35

† indicates the average results over all the nodes except the root node in the branch-and-price tree.
- indicates that the branch-and-price stopped at the root node.

Table 3 Details of the branch-and-price performance for the general VRPFT (1)–(17).

upper bound than CPLEX for 5 instances (p_18_4_7 and p_20_5_6 in Table 1 and p_18_3_8, p_18_4_7, and p_20_5_6 in Table 2).

The comparison between the results that are shown in Tables 1 and 2 also reveal that the VRPFT with fixed line direction is computationally less challenging to solve than the general VRPFT. As indicated in Section 3.2, this is due to the need of additional conic constraints to model the general VRPFT as compared to the fixed line moving direction

		Root Node			Other Nodes†		
Instance Name	Number of Nodes	Iterations	CPU Master Problem	CPU Subproblem	Avg. Iterations	Avg. CPU Master Problem	Avg. CPU Subproblem
p_10_3_6	1	34	<0.01	31.39	-	-	-
p_10_4_5	1	27	<0.01	20.55	-	-	-
p_10_5_4	21	22	<0.01	6.91	4	<0.01	0.95
p_12_3_6	1	40	<0.01	60.55	-	-	-
p_12_4_5	39	30	<0.01	27.42	5	<0.01	2.88
p_12_5_5	9	34	<0.01	30.26	9	<0.01	6.15
p_14_3_7	1	56	<0.01	361.91	-	-	-
p_14_4_6	45	57	<0.01	230.12	7	<0.01	19.99
p_14_5_5	7	38	<0.01	52.75	8	<0.01	8.60
p_16_3_8	1	83	0.01	3035.33	-	-	-
p_16_4_6	41	62	<0.01	353.82	7	<0.01	30.58
p_16_5_6	3	63	<0.01	314.19	12	<0.01	43.37
p_18_3_8	48	83	0.01	2665.69	5	<0.01	97.05
p_18_4_7	45	76	0.01	1409.48	12	<0.01	144.74
p_18_5_6	21	68	0.03	536.00	9	<0.01	39.89
p_20_3_9	1	58	0.01	7199.86	-	-	-
p_20_4_7	7	95	0.39	2549.12	16	0.07	366.69
p_20_5_6	349	73	0.01	589.63	6	<0.01	19.00

† indicates the average results over all the nodes except the root node in the branch-and-price tree.
- indicates that the branch-and-price stopped at the root node.

Table 4 Details of the branch-and-price performance for VRPFT with fixed line direction.

case. Besides instance p_18_5_6, solving VRPFT with fixed line direction using branch-and-price required less computational time than solving the general VRPFT.

The details of the computational performance of the various components of the proposed branch-and-price algorithm are presented in Tables 3 and 4. We notice that branch-and-price explored a limited number of nodes in the tree to find the optimal solutions. This is due to the relatively tight bounds that are obtained at each node of the tree through the column generation. Furthermore, the results show that the solution of the root node consumes the majority of the computational time as compared to the other nodes in the tree. This is mainly due to the fact that each node is warm started with a set of columns that were generated at the parent node as highlighted in Section 5.3. Warm starting each node with a set of columns leads to a decrease in the number of iterations that are required to solve the problem at each node and thus, as shown the results, the average number of iterations that are required at each of the descendant nodes of the tree are significantly less than the number of iterations that are required at the root node. Furthermore, the results show that the majority of the computational time is spent on solving the subproblem while solving the master problem requires a fraction of a second. The high computational time for solving the subproblem is mainly due to the fact that the subproblem which is solved repeatedly is a single vehicle routing with floating targets which is still challenging to solve. Thus investigating approaches to efficiently solve the single vehicle routing with floating

targets problem may highly improve the performance of the presented branch-and-price approach.

Next, we evaluate the impact of the addition of valid inequalities to the performance of the branch-and-price and CPLEX.

6.5. Impact of the Valid Cuts

In this section, we evaluate the effect of the valid cuts on the computational performance. We note that only Cuts (20) and (21) can be used in the branch-and-price approach as they preserve the decomposable structure of the problem into identical subproblems while

Instance Name	No Valid Cuts		Cuts (20)		Cuts (21)		Cuts (22)–(23)		Cuts (24)		Cuts (25)–(26)		All Cuts	
	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)
p_10_3_6	0%	229.61	0%	231.39	0%	166.58	0%	**114.68**	0%	120.62	0%	133.69	0%	114.94
p_10_4_5	0%	449.27	0%	446.52	0%	446.11	0%	38.90	0%	56.95	0%	124.17	0%	**38.88**
p_10_5_4	0%	542.51	0%	545.09	0%	545.17	0%	132.83	0%	180.62	0%	307.50	0%	**132.37**
p_12_3_6	0%	579.7	0%	578.85	0%	576.19	0%	191.80	0%	**170.54**	0%	1330.81	0%	192.77
p_12_4_5	8%	>7200	8%	>7200	8%	>7200	0%	**2925.95**	0%	3060.27	4%	>7200	0%	6874.47
p_12_5_5	0%	4107.24	0%	4008.89	0%	4028.73	0%	3295.91	0%	**1301.30**	0%	2794.89	0%	3447.58
p_14_3_7	0%	6961.36	0%	7033.35	0%	7073.75	0%	1202.05	0%	2446.03	0%	1226.49	0%	**1197.30**
p_14_4_6	13%	>7200	8%	>7200	12%	>7200	1%	>7200	0%	**3172.38**	10%	>7200	0%	4132.75
p_14_5_5	8%	>7200	8%	>7200	8%	>7200	0%	>7200	0%	**6844.89**	14%	>7200	1%	>7200
p_16_3_8	8%	>7200	9%	>7200	8%	>7200	**5%**	>7200	8%	>7200	8%	>7200	**5%**	>7200
p_16_4_6	2%	>7200	14%	>7200	2%	>7200	1%	>7200	0%	**5676.89**	2%	>7200	**1%**	>7200
p_16_5_6	33%	>7200	9%	>7200	16%	>7200	**5%**	>7200	10%	>7200	11%	>7200	28%	>7200
p_18_3_8	35%	>7200	30%	>7200	27%	>7200	22%	>7200	34%	>7200	28%	>7200	**17%**	>7200
p_18_4_7	31%	>7200	31%	>7200	30%	>7200	17%	>7200	37%	>7200	**9%**	>7200	17%	>7200
p_18_5_6	25%	>7200	23%	>7200	26%	>7200	20%	>7200	**10%**	>7200	19%	>7200	12%	>7200
p_20_3_9	36%	>7200	37%	>7200	37%	>7200	**31%**	>7200	**31%**	>7200	37%	>7200	**31%**	>7200
p_20_4_7	34%	>7200	35%	>7200	34%	>7200	28%	>7200	33%	>7200	**25%**	>7200	33%	>7200
p_20_5_6	27%	>7200	26%	>7200	26%	>7200	23%	>7200	24%	>7200	14%	>7200	**8%**	>7200
Avg. Speedup		1		1		0.98		0.75		0.70		0.92		0.76

Table 5 Impact of the valid inequalities for VRPFT.

Instance Name	No Valid Cuts		Cuts (20)		Cuts (21)		Cuts (22)–(23)		Cuts (24)		Cuts (25)–(26)		All Cuts	
	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)
p_10_3_6	0%	154.09	0%	96.79	0%	127.84	0%	58.66	0%	97.73	0%	**54.74**	0%	71.00
p_10_4_5	0%	104.2	0%	103.26	0%	103.26	0%	26.03	0%	**19.11**	0%	56.41	0%	20.38
p_10_5_4	0%	99.32	0%	99.23	0%	99.11	0%	**43.15**	0%	135.02	0%	43.34	0%	47.92
p_12_3_6	0%	111.76	0%	111.48	0%	112.1	0%	**89.93**	0%	284.16	0%	76.29	0%	127.43
p_12_4_5	0%	3510.18	0%	2425.39	0%	2355.82	0%	1595.86	0%	2305.95	0%	3488.67	0%	**1432.01**
p_12_5_5	0%	2669.57	0%	2353.36	0%	1050.18	0%	489.88	0%	3468.17	0%	610.79	0%	**216.71**
p_14_3_7	0%	2676.76	0%	2662.66	0%	2673.79	0%	**727.29**	0%	764.03	0%	1078.46	0%	1133.00
p_14_4_6	11%	>7200	4%	<7200	10%	<7200	20%	<7200	0%	**3407.89**	35%	>7200	0%	5855.59
p_14_5_5	8%	>7200	8%	<7200	8%	<7200	0%	**3973.01**	0%	4514.05	0%	4600.9	0%	6253.56
p_16_3_8	7%	>7200	12%	<7200	10%	<7200	5%	<7200	9%	>7200	8%	>7200	**2%**	<7200
p_16_4_6	26%	>7200	20%	<7200	25%	<7200	14%	<7200	**10%**	>7200	18%	>7200	24%	<7200
p_16_5_6	21%	>7200	21%	<7200	21%	<7200	0%	**6444.95**	30%	>7200	20%	>7200	14%	<7200
p_18_3_8	49%	>7200	47%	<7200	48%	<7200	**36%**	<7200	54%	>7200	**36%**	>7200	50%	<7200
p_18_4_7	30%	>7200	27%	<7200	35%	<7200	**20%**	<7200	29%	>7200	33%	>7200	30%	<7200
p_18_5_6	30%	>7200	31%	<7200	35%	<7200	**19%**	<7200	**19%**	>7200	26%	>7200	36%	<7200
p_20_3_9	53%	>7200	53%	<7200	54%	<7200	50%	<7200	**43%**	>7200	48%	>7200	49%	<7200
p_20_4_7	44%	>7200	45%	<7200	43%	<7200	44%	<7200	**40%**	>7200	**40%**	>7200	47%	<7200
p_20_5_6	44%	>7200	43%	<7200	40%	<7200	31%	<7200	41%	>7200	**30%**	>7200	42%	<7200
Avg. Speedup		1		0.95		0.94		0.73		0.95		0.79		0.77

Table 6 Impact of the valid inequalities for VRPFT with fixed line direction.

Instance	Without Valid Cuts		Cuts (20)		Cuts (21)		All Cuts	
Name	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)
p_10_3_6	0%	183.84	0%	183.36	0%	**157.11**	0%	177.07
p_10_4_5	0%	70.38	0%	61.52	0%	**51.35**	0%	62.98
p_10_5_4	0%	161.28	0%	172.78	0%	**117.37**	0%	140.14
p_12_3_6	0%	377.43	0%	416.68	0%	375.78	0%	**372.61**
p_12_4_5	0%	1209.43	0%	1152.71	0%	**1098.71**	0%	1115.19
p_12_5_5	0%	532.05	0%	485.62	0%	605.36	0%	**470.94**
p_14_3_7	0%	3841.94	0%	1046.25	0%	1115.42	0%	**974.05**
p_14_4_6	0%	5131.43	0%	6829.27	0%	4052.53	0%	**4043.21**
p_14_5_5	0%	377.12	0%	372.61	0%	376.33	0%	**361.07**
p_16_3_8	57%	>7200	56%	>7200	57%	>7200	**54%**	>7200
p_16_4_6	0%	**6635.41**	0%	6668.42	0%	6651.75	0%	6680.76
p_16_5_6	0%	6084.35	0%	**5792.52**	0%	6397.58	0%	6174.77
p_18_3_8	55%	>7200	55%	>7200	**53%**	>7200	55%	>7200
p_18_4_7	**7%**	>7200	**7%**	>7200	**7%**	>7200	**7%**	>7200
p_18_5_6	0%	1132.36	0%	1145.51	0%	**1057.68**	0%	1144.65
p_20_3_9	74%	>7200	**73%**	>7200	**73%**	>7200	74%	>7200
p_20_4_7	51%	>7200	**50%**	>7200	**50%**	>7200	51%	>7200
p_20_5_6	11%	>7200	12%	>7200	**10%**	>7200	11%	>7200
Avg. Speedup		1		0.97		0.91		0.92

Table 7 Impact of the valid inequalities on the branch-and-price for VRPFT.

Instance	Without Valid Cuts		Cuts (20)		Cuts (21)		All Cuts	
Name	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU(s)
p_10_3_6	0%	31.40	0%	30.61	0%	**30.26**	0%	31.98
p_10_4_5	0%	20.56	0%	**20.40**	0%	20.54	0%	20.44
p_10_5_4	0%	25.95	0%	25.91	0%	25.87	0%	**25.79**
p_12_3_6	0%	60.57	0%	60.50	0%	**60.45**	0%	60.49
p_12_4_5	0%	**137.00**	0%	137.79	0%	137.74	0%	137.25
p_12_5_5	0%	79.52	0%	78.80	0%	80.71	0%	**78.74**
p_14_3_7	0%	361.97	0%	**313.58**	0%	354.53	0%	312.41
p_14_4_6	0%	1109.85	0%	1112.98	0%	**1108.45**	0%	1113.28
p_14_5_5	0%	104.39	0%	106.56	0%	95.17	0%	**91.76**
p_16_3_8	0%	3035.48	0%	**2862.25**	0%	3472.14	0%	3210.63
p_16_4_6	0%	1577.55	0%	**1061.73**	0%	1167.65	0%	1087.60
p_16_5_6	0%	**401.00**	0%	452.38	0%	501.65	0%	414.71
p_18_3_8	16%	>7200	14%	>7200	**12%**	>7200	14%	>7200
p_18_4_7	**2%**	>7200	4%	>7200	4%	>7200	4%	>7200
p_18_5_6	0%	1333.93	0%	**1252.57**	0%	1396.91	0%	1283.96
p_20_3_9	67%	>7200	**64%**	>7200	67%	>7200	**64%**	>7200
p_20_4_7	0%	4750.29	0%	4509.42	0%	4616.61	0%	**4438.09**
p_20_5_6	**7%**	>7200	**7%**	>7200	**7%**	>7200	**7%**	>7200
Avg. Speedup		1		0.97		1		0.97

Table 8 Impact of the valid inequalities on the branch-and-price for VRPFT with fixed line direction.

all the valid cuts can be used within CPLEX. Tables 5 and 6 show the computational performance of CPLEX in solving VRPFT and VRPFT with fixed line line directions using various sets of valid cuts.

The results that are shown in Table 5 reveal that the addition of cuts (20) and (21) has limited impact on the computational time while the addition of (22)–(23), (24), and (25)–(26) decreases the computational time by up to 30% on average. For the instances that are not solved to optimality within the time limit, tighter bounds are obtained when valid cuts are added. Furthermore as shown in Table 5, optimal solutions for instances p_12_4_5, p_14_4_6, p_14_5_5, and p_16_4_6 are obtained within the time limit after the addition of

valid cuts. As shown in Table 6, similar results are also obtained for the VRPFT with fixed line directions. The addition of cuts (20) and (21) leads to a decrease of 5% and 6% respectively in the average computational time. The addition of (22)–(23), (24), and (25)–(26) decreases the computational time by up to 27% on average with tighter bounds obtained for the instances that are not solved within the time limit. Furthermore, optimal solutions are obtained within the time limit for p_14_4_6, p_14_5_5, and p_16_5_6 with the addition of the valid cuts.

As shown in Tables 7 and 8, the addition of valid inequalities also improves the performance of the branch-and-price algorithm. For VRPFT, the addition of cuts (20) leads to an average decrease of 3% in the computational time while the addition of cuts (21) leads to an average decrease of 9%. Tighter bounds are also obtained for the instances that are not solved within the time limit to optimality. For the VRPFT with fixed direction, the computational time of branch-and-price also decreases by 3% on average with the addition of cuts (20) while the addition of cuts (21) has limited impact on the performance.

7. Conclusions

This paper presented the vehicle routing problem with floating targets and proposed a branch-and-price algorithm as a solution approach. The presented problem is a challenging optimization problem that has important applications in practice and is also interesting from the theoretical point of view as it generalizes the vehicle routing problem which is known to be difficult. The branch-and-price solution approach that is presented is based on exploiting the structure of the problem through a Lagrangian decomposition while a rule that preserves the decomposable structure of the problem is used for branching. Valid cuts are also proposed to improve the computational performance of the branch-and-price as well as the direct solution of the problem using CPLEX. Results on several instances with varying properties showcase the advantages of the proposed branch-and-price. In particular, the proposed branch-and-price is capable in solving instances in less computational time than CPLEX and optimal solutions are obtained for instances where CPLEX fails in solving the problem within the 2 hours computational time limit.

Future work will focus on further improving the computational performance of the solution approach to be able to solve larger instances. Particularly, since the computational testing revealed that the majority of the computational time is spent on solving the

subproblem, investigating solution approaches for the subproblem may potentially have a significant impact in improving the computational performance and in solving larger instances. Furthermore, additional extensions for practical applications such as the inclusion of time windows and uncertainty may also be investigated.

References

Aïvodji UM, Gambs S, Huguet MJ, Killijian MO (2015) Privacy preserving carpooling. *Odysseus 2015 - 6th International Workshop on Freight Transportation and Logistics, Ajaccio.*

Asahiro Y, Horiyama T, Makino K, Ono H, Sakuma T, Yamashita M (2006) How to collect balls moving in the Euclidean plane. *Discrete Applied Mathematics* 154(16):2247 – 2262, ISSN 0166-218X.

Baldacci R, Bartolini E, Mingozzi A (2011) An exact algorithm for the pickup and delivery problem with time windows. *Operations Research* 59(2):414–426.

Baldacci R, Maniezzo V, Mingozzi A (2004) An exact method for the car pooling problem based on Lagrangean column generation. *Operations Research* 52(3):422–439.

Barnes JW, Wiley VD, Moore JT, Ryer DM (2004) Solving the aerial fleet refueling problem using group theoretic tabu search. *Mathematical and Computer Modelling* 39(6):617–640.

Bit-Monnot A, Artigues C, Huguet MJ, Killijian MO (2013) Carpooling: the 2 Synchronization Points Shortest Paths Problem. Frigioni D, Stiller S, eds., *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 33 of *OpenAccess Series in Informatics (OASIcs)*, 150–163 (Dagstuhl, Germany), ISBN 978-3-939897-58-3, ISSN 2190-6807.

Bruck BP, Incerti V, Iori M, Vignoli M (2015) On the development of a practical carpooling application. *Odysseus 2015 - 6th International Workshop on Freight Transportation and Logistics, Ajaccio.*

Choubey NS (2013) Moving target travelling salesman problem using genetic algorithm. *International Journal of Computer Applications* (70):975–8887.

Coelho LC, Laporte G (2013) The exact solution of several classes of inventory-routing problems. *Computers & Operations Research* 40(2):558–565.

Cordeau JF (2006) A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 54(3):573–586.

Dantzig GB, Ramser JH (1959) The truck dispatching problem. *Management Science* 6(1):80–91.

Desrochers M, Desrosiers J, Solomon M (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Operations research* 40(2):342–354.

Dumas Y, Desrosiers J, Gelinas E, Solomon MM (1995) An optimal algorithm for the traveling salesman problem with time windows. *Operations Research* 43(2):367–371.

Fischetti M, González JJS, Toth P (1995) Experiments with a multi-commodity formulation for the symmetric capacitated vehicle routing problem. *in Proceedings of the 3rd Meeting of the EURO Working Group on Transportation, 169–173*.

Fisher ML (2004) The Lagrangian relaxation method for solving integer programming problems. *Management science* 50(12_supplement):1861–1871.

Fisher ML, Jörnsten KO, Madsen OB (1997) Vehicle routing with time windows: Two optimization algorithms. *Operations Research* 45(3):488–492.

Frangioni A (2005) About Lagrangian methods in integer optimization. *Annals of Operations Research* 139(1):163–193, ISSN 0254-5330.

Gendreau M, Hertz A, Laporte G, Stan M (1998) A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research* 46(3):330–335.

Gutin G, Punnen AP (2002) *The traveling salesman problem and its variations*, volume 12 (Springer Science & Business Media).

Hammar, Nilsson (2002) Approximation results for kinetic variants of TSP. *Discrete & Computational Geometry* 27(4):635–651, ISSN 0179-5376.

Helvig CS, Robins G, Zelikovsky A (2003) The moving-target traveling salesman problem. *Journal of Algorithms* 49(1):153–174.

Jiang Q, Sarker R, Abbass H (2005) Tracking moving targets in the non-stationary travelling salesman problem. *Complexity International* 171–179.

Kohl N, Madsen OB (1997) An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. *Operations Research* 45(3):395–406.

Laporte G (1992) The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* 59(2):231–247.

Mallick M, Vo BN, Kirubarajan T, Arulampalam S (2013) Introduction to the issue on multitarget tracking. *IEEE Journal of Selected Topics in Signal Processing* 7(3):373–375.

Mehrotra A, Trick MA (1996) A column generation approach for graph coloring. *Informs Journal on Computing* 8(4):344–354.

Mingozzi A, Bianco L, Ricciardelli S (1997) Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints. *Operations Research* 45(3):365–377.

Reyes D, Savelsbergh M, Toriello A (2016) Vehicle routing with roaming delivery locations. *Optimization Online* (2016-01-5281).

Ryan DM, Foster BA (1981) An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling* 269–280.

Stieber A, Fügenschuh A, Epp M, Knapp M, Rothe H (2015) The multiple traveling salesmen problem with moving targets. *Optimization Letters* 9(8):1569–1583.

Sundar K, Rathinam S (2013) Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. *CoRR* abs/1304.0494, URL `http://arxiv.org/abs/1304.0494`.

Thomas PR, Bhandari U, Bullock S, Richardson TS, du Bois JL (2014) Advances in air to air refuelling. *Progress in Aerospace Sciences* 71:14 – 35, ISSN 0376-0421.

Toth P, Vigo D (2014) *Vehicle Routing: Problems, Methods, and Applications, Second Edition* (SIAM).

Varone S, Aissat K (2015) Multi-modal transportation with public transport and ride-sharing. *ICEIS 2015*.