The Stochastic Vehicle Routing Problem

Research Paper
Vrije Universiteit Amsterdam
Faculty of Sciences
Study programme Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

Dimitry Erkin, email:dimitry.erkin@gmail.com
Supervisor Prof. G.M. Koole

February 12, 2017

Contents

1. Summary

The Vehicle Routing Problem (VRP) is one of the most famous combinatorial problems that has been studied by the mathematical community for many decades. In its classical case, called the Capacitated Vehicle Routing Problem (CVRP), this question is stated as a calculation of the optimal set of vehicle routes through the customers' placements that satisfies a number of constraints. All the routes have the same departure and finishing point. Only one vehicle visits a particular customer. Every customer has to be approached by at least one vehicle. All the vehicles have equal limited capacity. The quality of a route is measured as the amount of travelling costs that has to be minimized (Laporte, 2007). Initially this problem was introduced in 1959 as the generalization of the gasoline distribution issue from the fuel depot among the network of petrol-stations. Since that time this question has been analysed permanently not only because it is challenging from the mathematical perspective but also since it has a significant relevance for many practical applications such as transportation and logistic optimization (Hempsch & Irnich, 2014).

Pick-up and delivery problems (PDP) are an important subset of the VRP in which goods have to be collected at the source pick-up locations and carried to the destination delivery locations (Battarra et al., 2014). Another example of such problem is people transportation. In the case of dealing with people an additional constraint is introduced such as the maximum ride time which limits the time a passenger spends in a vehicle. Presence of this constraint turns the PDP into the Dial-a-Ride Problem (DARP) (Ropke & Cordeau, 2009). The PDP has many practical applications therefore we focus on this problem for the rest of this paper.

1.1 Uncertainty of input parameters

The VRP with random travelling times was initially stated as a constraint satisfaction problem such that the penalty of an arbitrary route is proportional to how much this route exceeds the time limit (Laporte et al, 1992). Later on, a number of studies have been carried out in this field such as those which consider the minimization of the time when the last vehicle returns to the depot and the maximization of the likelihood of the whole fleet having finished their routes by a certain moment in time (Kenyon & Morton, 2003), those that model the economic effect of the cases when a route time goes beyond a predefined time threshold (Campbell et al, 2011), and those that take into account the customers' preferences of when exactly it is beneficial for each of them to receive goods (Tas et al, 2012). This stochastic times formulation makes sense in many practical situations. For instance, when the drivers who deliver goods to customers have to be paid according to a higher tariff if they work after the normal working hours. From the other hand, those drivers' journeys among the clients' locations are in fact dependent not only on the travelling distances but also on the factors such as the weather conditions, the incidents and reconstruction on the roads, and so on (Laporte et al, 1992).

Another input parameter that is convenient to consider as uncertain is the customer demand (Gendreau et al, 2014). Under the presense of this kind of uncertainty a route can become unfeasible by the time of visiting a customer because of the vehicle capacity restrictions. For instance, Bertsimas (1992) studies the problem of money collection performed by a bank among its branches on a daily basis. The amount of money to be picked up at a branch and delivered to the bank headquarter is known up to its stochastic distribution parameters. From the other hand, the money carrying vehicle capacity is limited due to the safety reasons. Therefore, it is practically possible that a vehicle does not have enough room to load all the money when it arrives at a particular branch location.

Customers behaviour is also a stochastic parameter in a sence that an arbitrary customer could be absent by the time a vehicle arrives at their location. Sungur et al. (2010) analysed the case of a courier company pick-ups and deliveries in the dense urban area where travelling times are relatively short and a courier usually serves several client at the same location. Some of those clients could abandon the service so that the probabilistic nature of a client is also important to consider.

1.2 Availability of input parameters

The PDP problems with uncertain input parameters are closely related to another broader class that are the dynamic VRP problems. The latter ones not only regard the input parameters as being known up to the probability distribution parameters but also study the cases when those parameters are not known in advance by the time the generated optimal policy gets executed (Gendreau et al, 2014). Gendreau et al. (1999) investigates the international shipping service company example where the couriers have to pick up parcels at the customers locations and deliver them to a central office. What is essential is that new customer requests could take place dynamically during the day so that the fleet policy should be adjusted. The benefits of a method that takes into account the dynamic nature of travelling times are clearly demonstrated in Taniguchi & Shimamoto (2004). This study shows that by considering real time information on the roads instead of forecasting can reduce total costs and improve the quality of service, that is the number of times when a vehicle arrives at the customer location on time. Besides dynamic customer behaviour and the unpredictible nature of the environment there is another factor that is the dynamic vehicle availability which also comes to play a role (Gendreau et al, 2014). There are many situations such as car accidents, drivers illness and so on, that could affect the feasibility of a policy computed in advance.

1.3 Commodity handling patterns

Another classification of the PDP is based on a certain type of commodity handling. According to this classification the PDP can be subdivided into three categories such as many-to-many (M-M) problem, one-to-many-to-one (1-M-1) problem and one-to-one (1-1) problem (Battarra et al., 2014). For the (M-M) case each commodity item can be picked up from and delivered to many locations. An example of such problem is a car sharing service. For the (1-M-1) case there are two distinct commodity types: one is to be delivered from the depot to one of customer locations and another is to be picked-up at those locations and delivered back to the depot. For instance, the distribution of beverages and collection of empty bottles can be considered as the example of (1-M-1). Finally, for the (1-1) case each transportation request is to pick up at one origin location and to deliver this picked up customer to another destination location.

1.4 Classification of methods

The first approach that provides an exact solution of the VRP instance is based on the Branch-and-Bound algorithm combined with specific lower bound computation procedures (relaxations) and branching schemes (Christofides et al., 1981; Semet et al., 2014). The idea is to incrementally build a partial route by branching on the next locaton starting from the depot, passing through the depot $K-2$ times during the route (K is the number of vehicles) and finishing at the depot. Each branching on an arbitrary location j generates two child nodes of a decision tree: first one that corresponds to the decision to branch and includes arc from current location i to this arbitrary location j, and another one that is related to the decision not to branch and does not include arc (i, j). The branching decision about how to extend our current route (sub-tour) at any location except the depot is based on the combination of the maximal demand and the longest travelling distance. From the depot we usually go to a location with the maximal demand. The computation of the lower bound is based on the finding the cost of the Shortest Spanning Tree (SST) with fixed number of arcs connected to the depot vertex which is equal to K. The starting upper bound is usually computed by some heuristics. We stop when there are no tree nodes left to proceed.

The idea of the Branch-and-Bound algorithm is illustrated by the following example (Christofides et al., 1981). We consider $n = 10$ customers and a fleet of $K = 4$ vehicles. Each of the vehicles has capacity $Q = 24$, $i = 1$ is the depot. The cost matrix and the customers demand are represented in Table 1 and Table 2 respectively.

Table 1. Cost matrix

i	1	2	3	4	5	6	7	8	9	10	11
1	0.0	24.1	27.6	17.2	23.3	11.1	16.0	7.0	20.2	9.8	22.0

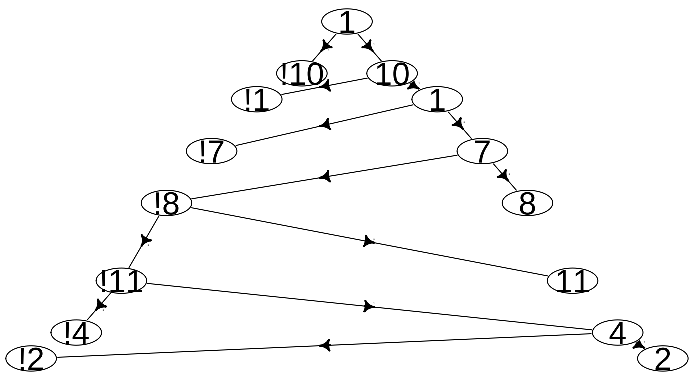
i	1	2	3	4	5	6	7	8	9	10	11
2	24.1	0.0	21.2	9.2	16.1	26.0	28.1	31.2	38.0	33.0	46.2
3	27.6	21.2	0.0	15.5	35.3	20.2	40.4	32.7	28.0	37.2	45.8
4	17.2	9.2	15.5	0.0	20.0	17.0	26.1	24.0	28.8	26.9	39.0
5	23.3	16.1	35.3	20.0	0.0	31.3	17.0	29.2	42.7	28.1	42.4
6	11.1	26.0	20.2	17.0	31.3	0.0	27.1	13.4	12.0	18.6	25.6
7	16.0	28.1	40.4	26.1	17.0	27.1	0.0	18.0	35.5	14.4	27.6
8	7.0	31.2	32.7	24.0	29.2	13.4	18.0	0.0	18.0	5.3	15.0
9	20.2	38.0	28.0	28.8	42.7	12.0	35.5	18.0	0.0	23.0	22.6
10	9.8	33.0	37.2	26.9	28.1	18.6	14.4	5.3	23.0	0.0	14.3
11	22.0	46.2	45.8	39.0	42.4	25.6	27.6	15.0	22.6	14.3	0.0

Table 2. Demand

	2	3	4	5	6	7	8	9	10	11
qi	1	5	6	12	13	13	3	9	21	10

In Figure 1 we see how this algorithm actually works. Our branching strategy is to consider all possible options from the current vertex and for each of those option apply a depth first procedure. Starting from the depo 1 we branch on customer 10 who has the highest demand of 21. After visiting this customer we have $24 - 21 = 3$ units of capacity left. So we have only two vertices left to consider that is $\{1, 7\}$. We first take 1, go to the depot and replenish our cargo so we have again full capacity available. Starting from 1 we branch on 7 because among all the remaining vertices $\{6, 7\}$ with the highest demand of 13 vertex 7 is located further from the depot. When we are at 7 we have $24 - 13 = 11$ capacity left so we consider vertices $\{2, 3, 4, 8, 9\}$ and first branch on vertex 8 because it is the nearest one. In the same way we proceed further until there are no more options to branch. All the nodes that correspond to the cases when we visited all the vertices and got back to the depot are the solutions. Among them there is one which is optimal.

Figure 1: Fragment of the Branch−and−Bound tree



The second method of finding an exact VRP instance solution is called the Branch-and-Price algorithm (Semet et al., 2014; Poggi et al., 2014). The idea is to apply so called Dantzig-Wolfe decomposition to the initial MIP problem and to split it into a master problem and a subproblem. The drawback is that the resulting set partitioning master problem formulation has in principle as many decision variables as many different feasible solutions (routes) has the initial problem. To cope with this drawback the Column Generation (CG) technique is applied. This technique allows to add to the linear relaxation of the master problem only the promising columns (routes) which have negative reduced cost. After that this relaxation is solved by a linear solver and a dual solution is used to generate new routes by solving the pricing subproblem. This process repeats until there are no new routes to add. Then we check the solution. If this solution satisfies the integrality constraints of the initial MIP then we solved the initial MIP problem. Otherwise, we branch either on vehicled or arcs, and repeat the above steps for child problems.

The third method that belongs to the family of exact VRP instance solvers is the Branch-and-Cut algorithm (Semet et al., 2014; Poggi et al., 2014). The idea is to relax the initial problem by removing integrality restrictions and some other constraints such as, for instance, the capacity limitations. Then the relaxed model is solved by a linear solver. If the obtained solution satisfies all the removed constrains then we found an optimal solution of the initial problem. Otherwise, the heuristic separation procedures are applied to find the violated removed constraints. After that we add those constraints to the relaxed model and solve it again. If there are no new violated removed constraints to add and the solution is not feasible for the initial problem then we branch on the fractional variable.

There is another dimension of the VRP that also has been studied for many years such as the computational efforts dependency on the number of vehicles and customers. Unfortunately, the VRP in general is a kind of problem that can be solved exactly only for the moderate size instances, that is where the number of customers is not greater than 100, given a reasonable amount of time. Therefore, what is needed in many real life application is to apply some heuristics that allow to find a sufficiently good suboptimal solution for larger cases but much faster. Those heuristics can be classified into several groups such as those which are tailored to find a beneficial starting route, those that are focused on the improvements of the initial route by some intelligent operations applied to the subsequent customers, and those whose aim is to efficiently explore the search space either iteratively or in a parallel manner, that is to say by evolving the population of solutions in time (Laporte et al, 2014).

1.5 Research question

The attention to the VRP resulted in a large variety of methods that have been developed in order to either exactly solve or approximate this problem. However, most of those techniques assume that all the input parameters which are required for the computation are deterministic and known upfront. This assumption limits the number of cases when the invented methods can be applied in practice, because in real life the factors such as demand, customers' presence, as well as both travelling and service times are usually stochastic. This observation is the main reason why in recent years the community has indicated a significant interest in solving the VRP with uncertain inputs (Gendreau et al, 2014).

In this paper we focus on the adaptation of classical techniques of solving determinstic cases of the VRP to the cases with nondeterministic parameters. More precisely, we study whether the celebrated Branch-and-Price algorithm can be effectively extended to solve the PDP instances with stochastic travel and service times. In particular, our aim is to determine the criteria of the solution acceptance in the presence of uncertainty. To make our problem more realistic we add the Time Windows constraint.

2. Methods

2.1 Notation

We use the notation of Cordeau (2006) with minor changes. To start with, we define n as the number of customers who have to be picked up. Next, we denote node set $V = \{1, ..., 2n + 2\}$ and arc set A. Nodes 1 and $2n + 2$ depict the origin and the destination depots while the subsets $P = \{2, ..., n + 1\}$ and $D = \{n + 2, ..., 2n + 1\}$ represent pick-up and delivery locations, respectively. Moreover, each customer has pick-up location $i \in P$ and corresponding delivery location $i + n \in D$. Arcs $(i, j) \in A | i \in V, j \in V$. After that, we define directed graph $G = (V, A)$.

For each node $i \in V$ we consider load q_i and service duration d_i such that $q_1 = q_{2n+2} = 0$; $q_i = -q_{i+n} = 1 | i \in P$; $d_1 = d_{2n+2} = 0$ and $d_i \geq 0 | i \in P \cup D$. Next for each node $i \in P \cup D$ we define time window $[e_i, l_i]$ where e_i

and l_i are the earliest and the latest time moments when serving a customer at this location can be started. The depot locations 1 and $2n+2$ also have corresponding time windows $[e_1, l_1]$ and $[e_{2n+2}, l_{2n+2}]$ which describe the time intervals of either leaving the origin or arriving to the destination. Furthermore, we define K as the number of vehicles having the same capacity Q. Moreover, we associate with each arc $(i,j) \in A$ routing cost c_{ij} and travel time t_{ij} and we assume that the triangle inequality holds both for routing costs and travel times.

2.2 Three-index formulation

Three-index formulation of the PDPTW is known as the following (Cordeau, 2006). For each arc $(i,j) \in A$ and each vehicle $k \in K$ let x_{ij}^k be binary variable which is equal to 1 if and only if vehicle k travels directly from i to j. For each location $i \in V$ and each vehicle $k \in K$ let u_i^k be time moment when vehicle k starts serving customer at location i, and w_i^k the load of vehicle k immediately after it leaves location i. For each location $i \in P$ and each vehicle $k \in K$ let r_i^k be the riding time of customer i on vehicle k. Furthermore, we define T as the maximal duration of route and L as the maximal customer riding time. Given these definitions, PDPTW can be formulated as the following:

$$\begin{array}{lll}
min \sum_{k\in K} \sum_{i\in V} \sum_{j\in V} c_{ij} x_{ij}^k & & (2.2.1) \\
\text{subject to} & & \\
\sum_{k\in K} \sum_{j\in V} x_{ij}^k = 1 & \forall i \in P & (2.2.2) \\
\sum_{j\in V} x_{1j}^k = \sum_{i\in V} x_{i,2n+2}^k = 1 & \forall k \in K & (2.2.3) \\
\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{i+n,j}^k = 0 & \forall i \in P, k \in K & (2.2.4) \\
\sum_{j\in V} x_{ji}^k - \sum_{j\in V} x_{ij}^k = 0 & \forall i \in P \cup D, k \in K & (2.2.5) \\
u_j^k \geq (u_i^k + d_i + t_{ij}) x_{ij}^k & \forall i \in V, j \in V, k \in K & (2.2.6) \\
w_j^k \geq (w_i^k + q_j) x_{ij}^k & \forall i \in V, j \in V, k \in K & (2.2.7) \\
r_i^k \geq u_{i+n}^k - (u_i^k + d_i) \geq 0 & \forall i \in P, k \in K & (2.2.8) \\
u_{2n+2}^k - u_1^k \leq T & \forall k \in K & (2.2.9) \\
e_i \leq u_i^k \leq l_i & \forall i \in V, k \in K & (2.2.10) \\
t_{i,i+n} \leq r_i^k \leq L & \forall i \in P, k \in K & (2.2.11) \\
max(0, q_i) \leq w_i^k \leq min(Q, Q + q_i) & \forall i \in V, k \in K & (2.2.12) \\
x_{ij}^k \in \{0, 1\} & \forall i \in V, j \in V, k \in K & (2.2.13)
\end{array}$$

In this formulation, constraint (2.2.2) ensures that each and every customer k is picked up only by one vehicle. Constraint (2.2.4) guarantees that if a customer i is picked up by vehicle k then he will be delivered by the same vehicle. Together constraints (2.2.2) and (2.2.4) ensure that each and every customer is served (picked up and delivered) exactly once and by the same vehicle. This means that there are no loops in the route.

The fact that each and every vehicle k leaves the origin and arrives to the destination depots exactly once is assured by constraint (2.2.3). Constraint (2.2.5) controls the continuity of any route that is the fact that if vehicle k arrives at either a pick-up or delivery location i then it will leave this location. Togeter constraints (2.2.3) and (2.2.5) guarantee that each and every vehicle starts its route at the origin depot and ends its route at the destination depot.

Constraint (2.2.6) ensures that vehicle k starts serving a customer at the next location j on its route not earlier than it finishes serving its customer at the current location i and travels from the current to the next location.

The fact that the total load w_j^k of vehicle k after leaving current location j is at least as large as that w_i^k of leaving the previous location i on its route plus the load q_j added/subtracted at this current location is guaranteed by constraint (2.2.7). Moreover, both the lower and upper bounds of the total load w_j^k of a vehicle k after leaving current location j are defined by constraint (2.2.12).

Constraint (2.2.8) sets up the lower bound of customer i riding time as the difference between the departure from his pick up location i and the time moment when this customer gets delivered to his destination $i+n$. Furthermore, both the lower and upper bounds of customer i riding time are defined by constraint (2.2.11) as the travelling time directly from his pick up location i to his delivery destination $i+n$ and fixed constant L respectively.

The maximal duration of route of each vehicle k is limited by constraint (2.2.9).

Constraint (2.2.10) guarantees that each and every location i is visited within its time window.

Given the maximal end time among all the time windows t_{max}, we rewrite constraint (2.2.6) as linear:

$$\begin{array}{lll} u_i^k - u_j^k + x_{ij}^k * (t_{max} + d_i + t_{ij}) \leq t_{max} & \forall i \in V, j \in V, k \in K & (2.2.14) \\ 0 \leq u_i^k \leq t_{max} & \forall i \in V, k \in K & (2.2.15) \end{array}$$

The aim of (2.2.14) is to guarantee that $u_i^k \leq u_j^k - d_i - t_{ij} | \forall i \in V, j \in V, k \in K$ if $x_{ij}^k = 1$. Furthermore, we have to ensure that u_i^k has no relation with u_j^k if $x_{ij}^k = 0$. In other words, time moment u_j^k when vehicle k starts serving a customer at location j has to be greater than or equal to time moment u_i^k when this vehicle starts serving a customer at location i plus time spent to service this customer at location i and to travel from i to j, given that j is the next location on the route of vehicle k after i. We can rewrite (2.2.14) as the following:

$$u_i^k - u_j^k + d_i + t_{ij} \leq (t_{max} + d_i + t_{ij}) * (1 - x_{ij}^k) \quad \forall i \in V, j \in V, k \in K \quad (2.2.16)$$

Looking at (2.2.16) we see that, indeed, if $x_{ij}^k = 1$ then $u_i^k \leq u_j^k - d_i - t_{ij}$. Moreover, if $x_{ij}^k = 0$ then $u_i^k - u_j^k + d_i + t_{ij} \leq t_{max} + d_i + t_{ij}$ thus $u_i^k \leq u_j^k + t_{max}$ but $u_i^k \leq t_{max} \wedge 0 \leq u_j^k$ so that u_i^k has no relation with u_j^k. Thus we can rewrite (2.2.16) as $u_i^k - u_j^k \leq (t_{max} + d_i + t_{ij}) - x_{ij}^k(t_{max} + d_i + t_{ij}) - d_i - t_{ij}$ thus $u_i^k - u_j^k + x_{ij}^k(t_{max} + d_i + tij) \leq (t_{max} + d_i + t_{ij}) - d_i - t_{ij}$ thus (2.2.14) holds.

While (2.2.7) and (2.2.12) provide the upper and the lower bounds of vehicle load, we study a way to find an exact relation. We do so because our objective is to formulate the PDPTW as a pure mixed integer problem and use a common MIP solver to get a solution. Let us define z_i as the absolute value of q_i. Given this definition, we need to guarantee that the load of vehicle k will be either increased or decreased exactly by z_i after visiting pick-up or delivery location j respectively, given that j is the next location on the route of vehicle k after i. We can do this if we prove the following theorem:

Theorem 1.

$$\begin{array}{lll} w_i^k q_j - w_j^k q_j + x_{ij}^k (Q + z_j) \leq Q & \forall i \in 1 \cup P \cup D, j \in P \cup D, k \in K & (2.2.17) \\ 0 \leq w_i^k \leq Q & \forall k \in K & (2.2.18) \\ w_1^k = w_{2n+2}^k = 0 & \forall k \in K & (2.2.19) \\ x_{ij}^k + h_{ij}^k = 1 & \forall i \in 1 \cup P \cup D, j \in P \cup D, k \in K & (2.2.20) \\ w_i^k q_j - w_j^k q_j + (Q+2) h_{ij}^k \geq -z_j & \forall i \in 1 \cup P \cup D, j \in P \cup D, k \in K & (2.2.21) \\ h_{ij}^k \in \{0, 1\} & \forall i \in 1 \cup P \cup D, j \in P \cup D, k \in K & (2.2.22) \end{array}$$

Proof:

Our aim is to guarantee that $\forall i \in 1 \cup P \cup D, j \in P \cup D, k \in K$ the following equalities hold:

$$\begin{array}{lll} w_j^k = w_i^k + z_j & x_{ij}^k = 1 \wedge j \in P & (2.2.23) \\ w_j^k = w_i^k - z_j & x_{ij}^k = 1 \wedge j \in D & (2.2.24) \end{array}$$

Furthermore, we have to ensure that w_i^k has no relation with w_j^k if $x_{ij}^k = 0$.

In order to do that we first notice that (2.2.23) holds if and only if both of the following inequalities hold:

$$\begin{array}{lll} w_j^k \leq w_i^k + z_j & x_{ij}^k = 1 \wedge j \in P & (2.2.25) \\ w_j^k => w_i^k + z_j & x_{ij}^k = 1 \wedge j \in P & (2.2.26) \end{array}$$

Similarly, for (2.2.24) we have:

$$\begin{array}{lll} w_j^k \leq w_i^k - z_j & x_{ij}^k = 1 \wedge j \in D & (2.2.27) \\ w_j^k => w_i^k - z_j & x_{ij}^k = 1 \wedge j \in D & (2.2.28) \end{array}$$

So to prove (2.2.23), (2.2.24) we have to prove (2.2.25)-(2.2.28). We can do this if we ensure that all the following statements hold:

$$\begin{array}{lll} (w_i^k - w_j^k)q_j + z_j \leq (Q + z_j)(1 - x_{ij}^k) & \forall i \in 1 \cup P \cup D, j \in P \cup D, k \in K & (2.2.29) \\ 0 \leq q_j(w_i^k - w_j^k) + z_j & (1/2 - x_{ij}^k) < 0 & (2.2.30) \end{array}$$

Indeed, if $x_{ij}^k = 1 \wedge j \in P$ then from (2.2.29) $w_i^k \leq w_j^k - z_j$ thus $w_j^k \geq w_i^k + z_j$ thus (2.2.26) holds. If $x_{ij}^k = 1 \wedge j \in D$ then from (2.2.29) $-w_i^k + w_j^k + z_j \leq 0$ thus $w_j^k \leq w_i^k - z_j$ thus (2.2.27) holds. If $x_{ij}^k = 0 \wedge j \in P$ then from (2.2.29) $w_i^k - w_j^k \leq Q$ thus $w_i^k \leq w_j^k + Q$ but $w_i^k \leq Q \wedge 0 \leq w_j^k$ so that w_i^k has no relation with w_j^k. If $x_{ij}^k = 1 \wedge j \in D$ then from (2.2.29) $-w_i^k + w_j^k \leq Q$ thus $-w_i^k \leq Q - w_j^k$ thus $w_i^k \geq w_j^k - Q$ but $w_i^k \geq 0 \wedge w_j^k \leq Q$ so that w_i^k has no relation with w_j^k. Consequently, based on (2.2.29) we prove (2.2.26) and (2.2.27) and demonstrate that regardless of the fact whether location j is pick-up or delivery w_i^k has no relation with w_j^k if $x_{ij}^k = 0$.

What is left is to prove (2.2.25) and (2.2.28). The idea behind (2.2.30) is based on the next observation. We rewrite both (2.2.25) and (2.2.28) as following:

$$\begin{array}{lll} 0 \leq w_i^k - w_j^k + z_j & q_j = 1 & (2.2.25) \\ 0 \leq -w_i^k + w_j^k + z_j & q_j = -1 & (2.2.28) \end{array}$$

Then we notice that (2.2.25) and (2.2.28) are actually the same if we consider q_j as the coefficient of w_i^k and w_j^k that is:

$$0 \leq q_j w_i^k - q_j w_j^k + z_j = B \qquad (2.2.30)$$

Furthermore, we notice that (2.2.29) should hold only if $x_{ij}^k = 1$ but not if $x_{ij}^k = 0$ because the latter case we study separately. Thus we arrive to the condition for (2.2.30) such as $A = (1/2 - x_{ij}^k) < 0$.

Next, we apply the linearization of if-then statement devised in Miller (2007) as folloving:

$$\begin{array}{ll} \text{IF}(A < 0)\text{THEN}(B \geq 0) & \\ N_a \geq max(A) & (2.2.30) \\ N_b \geq max(B) & (2.2.31) \\ h \in \{0,1\} & (2.2.32) \\ N_a h \geq A & (2.2.33) \\ A + (1-h)N_a \geq 0 & (2.2.34) \\ B \geq -hN_b & (2.2.35) \end{array}$$

In our case $max(A) = 1$ thus $N_a = 1$, $max(B) = (Q+1)$ thus $N_b = (Q+2)$. Therefore, from (2.2.33) we get $h_{ij}^k \geq 1/2 - x_{ij}^k$, from (34) we get $(1/2 - x_{ij}^k) + (1 - h_{ij}^k) \geq 0$, and from (2.2.35) we get $((w_i^k + z_j q_j) - w_j^k)q_j \geq -h_{ij}^k(Q+2)$. Thus, the following inequalities hold:

$$\begin{array}{ll} x_{ij}^k + h_{ij}^k \geq 1/2 & (2.2.36) \\ x_{ij}^k + h_{ij}^k \leq 3/2 & (2.2.37) \\ w_i^k q_j - w_j^k q_j + h_{ij}^k(Q+2) \geq -z_j & (2.2.38) \end{array}$$

From (2.2.36) if $x_{ij}^k = 1$ then $h_{ij}^k \geq -1/2$ but $h_{ij}^k in\{0,1\}$ thus $h_{ij}^k = 0$. From (2.2.37) if $x_{ij}^k = 1$ then $1/2 \geq h_{ij}^k$ but $h_{ij}^k in\{0,1\}$ thus $h_{ij}^k = 0$. From (2.2.38) if $h_{ij}^k = 0 \wedge j \in P$ then $(w_i^k + z_j) - w_j^k \geq 0$ thus $w_j^k \leq w_i^k + z_j$ thus (2.2.25) holds. From (38) if $h_{ij}^k = 0 \wedge j \in D$ then $(w_i^k - z_j) - w_j^k \leq 0$ thus $w_j^k \geq w_i^k - z_j$ thus (2.2.28) holds.

Therefore, all inequalities (2.2.25)-(2.2.28) hold. Thus, we prove that equalities (2.2.23) and (2.2.24) are valid. In addition, the linearization of (2.2.25) and (2.2.28) results in the inequalities (2.2.20)-(2.2.22).

Next, we notice that from (2.2.13),(2.2.20) and (2.2.22) $h_{ij}^k = 1 - x_{ij}^k$. Thus, we can exclude h_{ij}^k and rewrite (2.2.21) as the following:

$$w_i^k q_j - w_j^k q_j + (Q+2)(1 - x_{ij}^k) \geq -z_j \quad \forall i \in 1 \cup P \cup D, j \in P \cup D, k \in K \qquad (2.2.39)$$

Thus, we demonstrate that Theorem 1 holds. □

As a result, we get a pure mixed integer formulation of the PDPTW as the following:

$$
\begin{array}{lll}
min \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij}^k & & (2.2.1)\\
\text{subject to} & & \\
\sum_{k\in K}\sum_{j\in V} x_{ij}^k = 1 & \forall i\in P & (2.2.2)\\
\sum_{j\in V} x_{1j}^k = \sum_{i\in V} x_{i,2n+2}^k = 1 & \forall k\in K & (2.2.3)\\
\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{i+n,j}^k = 0 & \forall i\in P, k\in K & (2.2.4)\\
\sum_{j\in V} x_{ji}^k - \sum_{j\in V} x_{ij}^k = 0 & \forall i\in P\cup D, k\in K & (2.2.5)\\
r_i^k \geq u_{i+n}^k - (u_i^k + d_i) \geq 0 & \forall i\in P, k\in K & (2.2.8)\\
u_{2n+2}^k - u_1^k \leq T & \forall k\in K & (2.2.9)\\
e_i \leq u_i^k \leq l_i & \forall i\in V, k\in K & (2.2.10)\\
t_{i,i+n} \leq r_i^k \leq L & \forall i\in P, k\in K & (2.2.11)\\
max(0,q_i) \leq w_i^k \leq min(Q, Q+q_i) & \forall i\in V, k\in K & (2.2.12)\\
x_{ij}^k \in \{0,1\} & \forall i\in V, j\in V, k\in K & (2.2.13)\\
u_i^k - u_j^k + x_{ij}^k * (t_{max} + d_i + t_{ij}) \leq t_{max} & \forall i\in V, j\in V, k\in K & (2.2.14)\\
0 \leq u_i^k \leq t_{max} & \forall i\in V, k\in K & (2.2.15)\\
w_i^k q_j - w_j^k q_j + x_{ij}^k(Q+z_j) \leq Q & \forall i\in 1\cup P\cup D, j\in P\cup D, k\in K & (2.2.17)\\
0 \leq w_i^k \leq Q & \forall k\in K & (2.2.18)\\
w_1^k = w_{2n+2}^k = 0 & \forall k\in K & (2.2.19)\\
w_i^k q_j - w_j^k q_j + (Q+2)(1-x_{ij}^k) \geq -z_j & \forall i\in 1\cup P\cup D, j\in P\cup D, k\in K & (2.2.39)
\end{array}
$$

This formulation is for the deterministic case. It can be solved by a common MIP solver which is not extended to cope with VRP that is does not have a tailored for the VRP Branch-and-Cut algorithm implementation. It has no stochasticity component which we add later while considering the dynamic programming formulation. Furthermore, this formulation results in $2|K| + 3|P||K| + 3|V||K| + 3|V|^2|K|$ constraints so it is too big to solve efficiently for instances of a moderate size. However, it is a good option to test our algorithms on small instances.

2.3 Preprocessing

The VRP is the generalization of the TSP problem which is NP-hard even for the instances with Euclidean distances. Indeed, if we have n vertices to visit then at an arbitrary vertex i in order to go to a next vertex we need to know which vertices we already visited, thus our state space is all subsets of a set comprised of n elements. The cardinality of such state space is 2^n which is a huge number even for moderate n values. Thus, we want to remove as many arcs as it is possible in order to reduce the computational complexity of this problem. We do so by preprocessing problem parameters values as it is proposed in Cordeau (2006).

Time window tightening

To start with, we classify all customers as being either inbound or outbound. For the former case a customer wants to be picked up within a certain time window but his delivery preferences are not specified. Thus, for inbound customer i we have certain time window $[e_i, l_i]$ at pick up location i and wide time window $[0, t_{max}]$ at delivery location $i+n$. On the contrary, for the latter outbound case a customer wants to be delivered to his destination within a certain time window but his delivery preferences are not specified. Thus, for outbound customer i we have wide time window $[0, t_{max}]$ at pick up location i and certain time window $[e_{i+n}, l_{i+n}]$ at delivery location $i+n$.

In the case of an inbound user with pick-up location i, the time window at delivery location $i+n$ can be tightened as the following:

$$\begin{array}{ll} e_{i+n} = max(0, e_i + d_i + t_{i,i+n}) & (2.3.1) \\ l_{i+n} = min(t_{max}, l_i + d_i + L) & (2.3.2) \end{array}$$

In the case of an outbound user with delivery location j, the time window at pick-up location $j - n$ can be tightened as the following:

$$\begin{array}{ll} e_{j-n} = max(0, e_j - d_{j-n} - L) & (2.3.3) \\ l_{j-n} = min(t_{max}, l_j - d_{j-n} - t_{j-n,j}) & (2.3.4) \end{array}$$

The time windows at locations 1 and 2n + 2 can also be tightened as the following:

$$\begin{array}{ll} e_1 = e_{2n+2} = min_{i \in P \cup D} max(0, e_i - t_{1,i}) & (2.3.5) \\ l_1 = l_{2n+2} = max_{i \in P \cup D} l_i + d_i + t_{i,2n+2} & (2.3.6) \end{array}$$

Infeasible routes elimination

Given the PDPTW formulation provided in (2.2) several paths are not feasible due to time windows constraint (2.2.10), pairing and ride time constrains (2.2.8),(2.2.14) and (2.2.15). Therefore, some arcs can not belong to any feasible route and can be removed from G as the following:

$$\begin{array}{lll} (i, 1) & \forall i \in V & (2.3.7) \\ (i, i) & \forall i \in V & (2.3.8) \\ (1, n + i) & \forall i \in P & (2.3.9) \\ (i, 2n + 2) & \forall i \in P & (2.3.10) \\ (n + i, i) & \forall i \in P & (2.3.11) \\ (i, j) | e_i + d_i + t_{ij} > l_j & i \in V, j \in V & (2.3.12) \\ (i, j) | t_{ij} + d_j + t_{j,i+n} > L & i \in P, j \in V & (2.3.13) \\ (i, j) | t_{j-n,i} + d_i + t_{ij} > L & i \in V, j \in D & (2.3.14) \end{array}$$

Furthermore, a route through locations $\{n_1, n_2, n_3, n_4\}$ is not feasible if the following inequality holds:

$$max\{max(e_{n1} + d_{n1} + t_{n1,n2}, e_{n2}) + d_{n2} + t_{n2,n3}, e_{n3}\} + d_{n3} + t_{n3,n4} > l_{n4} \qquad (2.3.15)$$

Let us define I as the set of all infeasible routes through 4 locations according to (2.3.15). Given this definition we can eliminate the following arcs:

$$\begin{array}{lll}
(i,j)|\{j-n,i,j,i+n\} \in I & i \in P, j \in D & (2.3.16)\\
(i,j)|\{i-n,i,j,j+n\} \in I & i \in D, j \in P & (2.3.17)\\
(i,j)|\{i,j,i+n,j+n\} \in I, \{i,j,j+n,i+n\} \in I & i \in P, j \in P & (2.3.18)\\
(i,j)|\{i-n,j-n,i,j\} \in I, \{j-n,i-n,i,j\} \in I & i \in D, j \in D & (2.3.19)
\end{array}$$

As a result of preprocessing we get arc set $A' \subset A$ populated with only arcs that can belong to a feasible solution.

2.4 Set partitioning formulation

Set partitioning formulation of the PDPTW is based on the idea of Dantzig-Wolfe decomposition (Dantzig & Thapa, 2003; Rader, 2010). To illustrate this idea, we consider a linear programming problem of the form:

$$\begin{array}{ll}
min(c^T x) & (2.4.1)\\
\text{subject to} & \\
A_0 x = b_0 & (2.4.2)\\
A_1 x = b_1 & (2.4.3)\\
x \geq 0 & (2.4.4)
\end{array}$$

In this formulation, A_0 and A_1 are coefficients matrixes of size [m_0 x n] and [m_1 x n], respectively; $m_0+m_1 = m$; b_0 and b_1 are right-hand side vectors of size [m_0 x 1] and [m_1 x 1], respectively; c is the objective function coefficients vector of size [n x 1]; and x is the decision variables vector of size [n x 1].

We consider the problem:

$$\begin{array}{ll}
min(c^T x) & (2.4.1)\\
\text{subject to} & \\
A_0 x = b_0 & (2.4.2)
\end{array}$$

subject to the additional constraints:

$$\begin{array}{ll}
A_1 x = b_1 & (2.4.3)\\
x \geq 0 & (2.4.4)
\end{array}$$

Next, we consider the feasible region X for constraints (2.4.3) and (2.4.4). If we have the set Ω of all the extreme points $r \in \Omega$ of this region X, such that r is a vector of size [n x 1] corresponding to an arbitrary extreme point of X, then any solution $x \in X$ to

$$min(c^T x) \qquad (2.4.1)$$

$$\begin{array}{ll}
\text{subject to} & \\
A_0 x = b_0 & (2.4.2) \\
A_1 x = b_1 & (2.4.3) \\
x \geq 0 & (2.4.4)
\end{array}$$

can be written as the weighted by scalar coefficients α_r sum of these extreme points $r \in \Omega$:

$$\begin{array}{lll}
x = \sum_{r\in\Omega} \alpha_r r & & (2.4.5) \\
\sum_{r\in\Omega} \alpha_r = 1 & & (2.4.6) \\
\alpha_r \geq 0 & \forall r \in \Omega & (2.4.7)
\end{array}$$

Thus, if we know all those extreme points then we can substitute (2.4.5) into (2.4.1), (2.4.2) and reformulate this problem in terms of coefficients α_r:

$$\begin{array}{lll}
min(c^T \sum_{r\in\Omega} \alpha_r r) & & (2.4.8) \\
\text{subject to} & & \\
A_0 \sum_{r\in\Omega} \alpha_r r = b_0 & & (2.4.9) \\
\sum_{r\in\Omega} \alpha_r = 1 & & (2.4.6) \\
\alpha_r \geq 0 & \forall r \in \Omega & (2.4.7)
\end{array}$$

The resulting linear programming problem has only $m_0 + 1$ rows but $|\Omega|$ columns. Furthermore, if we solve it then by means of (2.4.5)-(2.4.7) we get the optimal solution for our initial problem (2.4.1)-(2.4.4).

Dantzig-Wolfe decomposition is in particular useful if the technological coefficients matrix A has a specific block-angular structure that is can be separated into two or more rows subsets. The first connecting subset A_0 corresponds to the constraints which together have non-zero coefficients for all decision variables. The rest of rows is divided into K subsets such that each and every decision variable has non-zero coefficient only in one of the subsets $A_k | k \in K$. In such case, the calculation of the solutions to subproblem k:

$$\begin{array}{ll}
min(c^T x) & (2.4.1) \\
\text{subject to} & \\
A_k x = b_k & (2.4.10) \\
x \geq 0 & (2.4.4)
\end{array}$$

can be performed only on rows of this subset A_k.

Given these insights, we apply Dantzig-Wolfe decomposition to the Three-index formulation of the PDPTW (2.2.1)-(2.2.13). To start with, we observe that all the constraints except (2.2.2) have separate rows for each and every vehicle $k \in K$. These rows corresponding to each vehicle k have zero coefficients for all other vehicles except k. Thus, we can distinguish K independent subsets $A_k | k \in K$ corresponding to each vehicle $k \in K$ (they are identical actually). Furthermore, constraint (2.2.2) has as many rows as there are pick-up locations P. Each row of that constraint guarantees that corresponding customer i is picked-up exactly once among all K routes. In other words, this row i has value 1 for each outgoing arc $(i, j)|\forall j \in V$. Thus, together all the (2.2.2) constraint rows have non-zero (actually 1) coefficients for all decision variables (arcs). Therefore, these rows correspond to connecting subset A_0.

After that, we notice that all the subproblems $k|k \in K$ for the PDPTW are the same because the objective function coefficients vector c has regular structure that is compised of K concatenated equal subvectors for each vehicle. Thus, we fix index $k = 1$ and define the PDPTW subproblem as the following:

$$\begin{array}{ll} min(c^T x) & (2.4.1) \\ \text{subject to} & \\ A_1 x = b_1 & (2.4.3) \\ x \geq 0 & (2.4.4) \end{array}$$

Next, we define the scalar cost vector of route r as $\tilde{c}_r = c^T r$. Given this definition, we rewrite (2.4.8) as the following:

$$min(\textstyle\sum_{r \in \Omega} \tilde{c}_r \alpha_r) \qquad (2.4.11)$$

After that, we recall that the number of independent subsets that we distinguished, or equivalently, the number of subproblems, is equal to the number of vehicles K. Thus, we rewrite (2.4.6) as the following:

$$0 \leq \textstyle\sum_{r \in \Omega} \alpha_r \leq K \qquad (2.4.12)$$

Next, we notice that in (2.4.9) each and every row of $A_0 r$ has only one value 1 and the remaining values 0. This follows from the fact that $r \in \Omega$ is an extreme point (solution) to subproblem. Thus, r has only one outgoing arc (i, j) with value 1 for any location $i \in V$. As it was already mentioned, A_0 in every row i has value 1 for each outgoing arc $(i, j)|\forall j \in V$. Being multiplied as $A_0 r$ they result in a matrix of size $[m_0$ x $1]$ such that each row of this matrix represents whether customer i is picked up by route r or not. Thus, being summed up over all routes $\sum_{r \in \Omega} \alpha_r A_0 r$ this row gives weighted by master problem coefficients α_r sum of a number of times customer i is picked up. If we restrict α_r as binary variable $\alpha_r \in \{0, 1\}$ then this weighted sum becomes the sum of a number of times customer i is picked up among all the routes that are part of the solution of:

$$\begin{array}{lll} min(\sum_{r \in \Omega} \tilde{c}_r \alpha_r) & & (2.4.11) \\ \text{subject to} & & \\ A_0 \sum_{r \in \Omega} \alpha_r r = b_0 & & (2.4.9) \\ 0 \leq \sum_{r \in \Omega} \alpha_r \leq K & & (2.4.12) \\ \alpha_r \in \{0, 1\} & \forall r \in \Omega & (2.4.13) \end{array}$$

Next, we introduce $a = A_0\Omega$, denote the column r of a as a_r and rewrite (2.4.9) as $\sum_{r\in\Omega} \alpha_r a_r = b_0$. According to (2.2.2) the sum of a number of times customer i is picked up among all the routes that are part of the solution have to be equal to 1. We notice that in order to satisfy this condition b_0 has to be the vector of values 1 for each row i. This observation allows us to rewrite $\sum_{r\in\Omega} \alpha_r a_r = b_0$ as i separate equalities:

$$\sum_{r\in\Omega} a_{ir}\alpha_r = 1 \quad \forall i \in P \quad (2.4.14)$$

As a result, we get the following set partitioning formulation of the PDPTW:

$$\begin{array}{lll} min(\sum_{r\in\Omega} \tilde{c}_r\alpha_r) & & (2.4.11) \\ \text{subject to} & & \\ \sum_{r\in\Omega} a_{ir}\alpha_r = 1 & \forall i \in P & (2.4.14) \\ 0 \leq \sum_{r\in\Omega} \alpha_r \leq K & & (2.4.12) \\ \alpha_r \in \{0,1\} & \forall r \in \Omega & (2.4.13) \end{array}$$

This formulation is called the integer programming master problem (IMP) and it is similar to the set partitioning formulation of the PDPTW (Ropke & Cordeau, 2009) except the additional constraint (2.4.12) that we introduce to limit the number of vehicles. However, Parragh & Cordeau (2015) introduced the same constraint for the set partitioning formulation of the Track and Trailer routing problem which has similar block-angular structure.

Let us notice that without (2.2.2) constraints (2.2.1)-(2.2.13) do not guarantee the absence of loop of length greater than 2 so to assure this condition we add to subproblem the following constraint (Ropke & Cordeau, 2009):

$$u_i^k + t_{i,i+n} \leq u_{i+n}^k \quad \forall i \in P, k \in K \quad (2.4.14)$$

2.5 Branch-and-price algorithm

The idea of the Branch-and-price algorithm is known as the following (Ropke & Cordeau, 2009; Parragh & Cordeau, 2015). First, the linear programming relaxation of the IMP is formulated which is called the linear programming master problem (LMP). The optimal solution of the LMP gives us the lower bound on the optimal solution of the IMP. Due to the large size of Ω it is usually computationally difficult to solve the LMP by considering all columns $r \in \Omega$. This is why a technique called Column Generation is applied to populate $\Omega' \in \Omega$ only with those columns that could potentially belong to optimal solution, and to solve the LMP on Ω'.

The idea of Column Generation is to start with some set of initial columns and calculate the optimal solution of the LMP first time. After that, dual information from this solution is used to obtain new promising columns that could improve current solution. Next, those new columns are added to Ω' and the LMP is solved again. This process repeats until there are no new promising columns that can be identified. If the solution which we have by that time satisfies integrality constraint (2.4.13) then we have the optimal solution of the IMP. Otherwise, the branching procedure is applied in order to generate two new instances of the LMP. Then the algorith is applied to both of these instances in a row according to some priority criterion.

Following this idea we formulate the LMP as the following:

$$\begin{array}{lll} min(\sum_{r\in\Omega}\tilde{c}_r\alpha_r) & & (2.4.11)\\ \text{subject to} & & \\ \sum_{r\in\Omega}a_{ir}\alpha_r = 1 & \forall i\in P & (2.4.14)\\ 0\le\sum_{r\in\Omega}\alpha_r\le K & & (2.4.12)\\ \alpha_r\ge 0 & \forall r\in\Omega & (2.4.7)\end{array}$$

Next, we add the initial set of columns (routes) to Ω. This initial set is filled by considering the simplest one customer routes plus a dummy route visiting all the customers using all arcs and not consuming the vehicle resourse in (2.4.12) (Parragh & Cordeau, 2015). Later on we extend this set by applying some construction heuristic.

After that we solve the LMP by the Simplex method first time and get vector β of the dual variables of solution α. This vector has m_0+2 rows such that first m_0 rows correspond to constraints (2.4.14) for each customer i while the remaining two rows m_0+1 and m_0+2 are related with (2.4.12). Let us notice that β_{m_0+1} and β_{m_0+2} always have values 1.

Given vector β, we search for new promising columns. These columns being added to Ω should have potential to improve the objective function value. In terms of the Simplex method it means that each of them should have negative reduced cost, that is the amount by which the objective value is changed if we change the value of the corresponding to this column r decision variable α_r by 1. Thus, column with more negative reduced cost could more effectively improve (decrease) the objective function value.

Let us denote by $\bar{A}$ the subset of A_0 and by $\bar{\beta}$ the subset of β that both correspond to (2.4.14). Next, we define the row i of $\bar{A}$ as $\bar{A}_i$, and the scalar reduced cost of pick-up location i as $\bar{\beta}_i$, respectively. Given this definition, the reduced cost $\hat{c}_r$ of column (route) r is computed as the following (Rader, 2010):

$$\hat{c}_r = \tilde{c}_r - \beta^T a_r = c^T r - \bar{\beta}^T(\bar{A}r) - \beta_{m_0+1} - \beta_{m_0+2} = (c - \bar{A}^T\bar{\beta})^T r - \beta_{m_0+1} - \beta_{m_0+2} \qquad (2.5.1)$$

We use (2.5.1) to formulate the problem of searching column (route) r with minimal reduced cost as the following (Rader, 2010):

$$\begin{array}{ll} min_{r\in\Omega}((c-\bar{A}^T\bar{\beta})^T r - \beta_{m_0+1} - \beta_{m_0+2}) & (2.5.2)\\ \text{subject to} & \\ A_1 r = b_1 & (2.5.3)\\ r\ge 0 & (2.5.4)\end{array}$$

This problem is called the pricing subproblem. For the PDPTW by analysis of (2.5.1) we get the reduced cost formula as:

$$\hat{c}_r = \tilde{c}_r - \sum_{i\in P}(\beta_i a_{ir}) - \beta_{m_0+1} - \beta_{m_0+2} \qquad (2.5.5)$$

As a result, we define the pricing subproblem for the PDPTW as the following:

$$min_{x\in\Omega}(\tilde{c}_x - \textstyle\sum_{i\in P}(\beta_i a_{ix}) - \beta_{m_0+1} - \beta_{m_0+2}) \quad (2.5.6)$$
subject to (2.2.3)-(2.2.13) with $k = 1$ (or dropping index k)

This formulation is similar to the pricing subproblem formulation for the PDPTW (Ropke & Cordeau, 2009) except the additional terms β_{m_0+1} and β_{m_0+2} that we introduce to limit the number of vehicles due to constraint (2.4.12). Let us notice that this problem is integer due to (2.2.13).

Dynamic programming algorithm for the pricing subproblem

We solve the PDPTW pricing subproblem by dynamic programming algorithm based on Ropke & Cordeau (2009). First, we define the state at any location $i \in V$ as the following:

t_i	the arrival time at i
$\hat{c}_i$	the reduced cost of reaching i
$\mathcal{V}_i$	the set of pick-up locations that have been visited before arriving at i
$\mathcal{O}_i$	the set of picked up customers that are on board at i
$\mathcal{S}_i$	the set of picking-up times for $\mathcal{V}_i$
s	the route starting time

Given there definitions, we formulate the value fuction as the following:

$$V_i(t_i, \hat{c}_i, \mathcal{V}_i, \mathcal{O}_i, \mathcal{S}_i, s) = \quad min_{(i,j)\in A'}(\hat{c}_i \wedge j = (2n+2) \vee V_j(t_j, \hat{c}_j, \mathcal{V}_j, \mathcal{O}_j, \mathcal{S}_j, s) \wedge j \in V \setminus \{2n+2\}) \quad (2.5.7)$$

subject to

$$t_j = \quad max(t_i + d_i + t_{ij}, e_j) \quad (2.5.8)$$

$$\hat{c}_j = \hat{c}_i + \quad \begin{cases} \tilde{c}_j - \beta_j | j \in P \\ \tilde{c}_j | j \in V \setminus P \end{cases} \quad (2.5.9)$$

$$\mathcal{V}_j = \quad \begin{cases} \mathcal{V}_i \cup \{j\} | j \in P \\ \mathcal{V}_i | j \in V \setminus P \end{cases} \quad (2.5.10)$$

$$\mathcal{O}_j = \quad \begin{cases} \mathcal{O}_i \cup \{j\} | j \in P \\ \mathcal{O}_i \setminus \{j-n\} | j \in D \\ \mathcal{O}_i | j \in V \setminus P \cup D \end{cases} \quad (2.5.11)$$

$$\mathcal{S}_j = \quad \begin{cases} \mathcal{S}_i \cup \{t_j + d_j\} | j \in P \\ \mathcal{S}_i | j \in V \setminus P \end{cases} \quad (2.5.12)$$

$$s = \quad \begin{cases} t_j - t_{ij} | i = 1 \\ s | i \neq 1 \end{cases} \quad (2.5.13)$$

stopping criteria

$$t_j > l_j \quad (2.5.14)$$

$$|\mathcal{O}_j| > Q \quad (2.5.15)$$

$$j \in P \wedge \mathcal{V}_i \cap \{j\} \neq 0 \quad (2.5.16)$$

$$
\begin{aligned}
&j \in D \wedge \{j-n\} \notin \mathcal{O}_i && (2.5.17)\\
&j \in D \wedge (t_j - \mathcal{S}_{j-n}) > L && (2.5.18)\\
&j = (2n+2) \wedge \mathcal{O}_j \neq 0 && (2.5.19)\\
&j = (2n+2) \wedge (t_j - s) > T && (2.5.20)
\end{aligned}
$$

In this formulation, $V_i(t_i, \hat{c}_i, \mathcal{V}_i, \mathcal{O}_i, s)$ is the minimal reduced cost of reaching location $2n+2$ from i. Thus, $V_1(0,0,\{\},\{\},0)$ gives us the minimal reduced cost of travelling from the origin depot 1 to the destination depot $2n+2$ while our route satisfies (2.2.3)-(2.2.13) and (2.4.14). Constraint (2.5.8) is used to compute arrival time to the next location according to its time window. The set of all picked up customers is maintaned by (2.5.10) such that each new pick up location is added to it upon arriving. Similarly, (2.5.11) takes care of the set of all customers who are currently on board, that is adds new one while getting to pick-up node and removes him as soon as he is delivered. The start riding time of each picked up customer is collected by (2.5.12) while (2.5.13) memorizes the time when vehicle leaves the origin depot.

As far as constraints satisfaction is concerned, if customer i is picked up on a route then this route will not end up at the destination untill this customer will be delivered because of (2.5.19). From the other hand, customer i will be delivered only if he is picked-up (2.5.17). Thus (2.2.4) is satisfied. Furthermore, We consider only routes from the origin depot and according to (2.5.7) the only one location where route can finish is $2n+2$ so that origin-to-destination constraints (2.2.3) and (2.2.5) are ensured. Pick-up-before-delivery constraint (2.2.6) is guaranteed because of (2.5.8). Constraint (2.5.15) takes care of the load related constraints (2.2.7) and (2.2.12). Riding time constraints (2.2.8) and (2.2.11) are ensured by (2.5.18) while the maximal duration of route is guaranteed not to exceed its threshold (2.2.9) by (2.5.20). Time window constraint (2.2.10) is satisfied by (2.5.14). In the Three-index formulation of the PDPTW without (2.2.2) the absence of loops is assured by (2.2.3)-(2.2.6) among with additional constraint (2.4.14). In order to fulfil this requirement we add (2.5.16).

Dominance rules

The above formulation being applied as is results in the evaluation of all feasible paths. In order to reduce the number of such evaluations we introduce so called dominance criterion similar to what is proposed in Ropke & Cordeau (2009), as the following:

$$
\begin{aligned}
&t_i^1 \leq t_i^2 \wedge \hat{c}_i^1 \leq \hat{c}_i^2 \wedge \mathcal{O}_i^1 = \mathcal{O}_i^2 && \mathcal{V}_i^1 = \mathcal{V}_i^2 \quad (2.5.21)\\
&t_i^1 \leq t_i^2 \wedge \hat{c}_i^1 \leq \hat{c}_i^2 - \textstyle\sum_{j \in (\mathcal{V}_i^2 \setminus \mathcal{V}_i^1)} \beta_j \wedge \mathcal{O}_i^1 = \mathcal{O}_i^2 && \mathcal{V}_i^1 \subset \mathcal{V}_i^2 \quad (2.5.22)
\end{aligned}
$$

If one of the above statements is true for two arbitrary arrivals at the same node i then arrival 2 is defined as being dominated by arrival 1, and, if this is the case then there is no need to continue dynamic recursion further from this arrival 2. Formula (2.5.21) relates to the case when the sets of visited customers are equal for both arrivals. Formula (2.5.22) models the situation when arrival 1 has to visit some customers that arrival 2 has already visited, thus, we need to deduct reduced costs related to the corresponding to those customers pick-up locations.

Local TSP

In order to reduce the number of evaluated paths we stop recursion at any state if all the combinations of being currently on board customers up to a certain number of people (up to 2 in the current implementation) can not be delivered on time due to the time windows restriction. Thus, we solve a number of small instances of the Travelling Salesman problem (Ropke & Cordeau, 2009).

NG-relaxation

In order to simplify the pricing subproblem we apply NG-relaxation (Baldacci et al, 2011). To start with, we define for each vertex j the set of a number of its closest neighbours N_j (10 in the current implementation) along with this vertex j itself. After that, we specify for vertex j its predecessor vertex i which has outgoing arc to j. Then, instead of the set of all visited pick-up locations $\mathcal{V}_j$ we consider the set NG_j which is comprised of vertex j itself and all the vertices of the set NG_i that remain being elements of NG_j because they are in N_j. The NG-set of the origin NG_1 is empty. In other words, the NG-set of each vertex is comprised of this vertex and all its predecessors that become and remain being included in the (cumulative) neighbourhood of this route. This (cumulative) neighbourhood is initialy formed as empty at the origin and then updated according to the described above rules each time we go to a next vertex j.

This technique helps us to identify an optimal substructure of the relaxed pricing subproblem and use this to narrow down the number or recursion steps. Indeed, two different arrivals to vertex j at the same time t_j with the same set of customers on board $\mathcal{O}_j$ and the same NG-set NG_j are not distinguishable in terms of reduced cost, that is the second and all the following arrivals will give the same reduced cost as the first one, thus, we can stop recursion and use cached reduced cost value. Furthermore, less the size of NG_j is more chance we have to get into cached value. Moreover, in practice many arrivals take place at the beginning of the vertex time window so they have the same t_j, thus, the chance to land into cached value in this particular case is even higher. As a result of applying NG-relaxation we get less computationaly difficult problem to solve.

Branching strategy

If the solution to the LMP is not integer then we apply so called branching that is we split current problem into two child problems while tightening those variables that violate integrality constrains. Following this idea we first consider the number of vehicles $\sum_{r\in\Omega}\alpha_r$ in (2.4.12) (Parragh & Cordeau, 2015). If this number if fractional then we set the upper bound for one child node and the lower bound for another to this fractional value. Otherwise, we split on arcs. In the latter case we sum up over all the columns (routes) in current solution the corresponding matrixes of arcs that form each route. These matrixes have only 0 or 1 values and are summed by element-wise summation so that resulting matrix has the same cardinality as all the inputs. Furthermore, before the summation each matrix is multiplied by the corresponding scalar value α_r. As a result, for each arc we get a fractional value from 0 to 1. Then we branch on the arc (i, j) which has the closest to 0.5 value. In the first child problem we remove this arc and all the columns (routes) that use it. In the second child problem we, on the contrary, reinforce this arc, that is we remove all the arcs to j but not from i and all arcs from i but not to j, and we also remove all the corresponding columns (routes) as well.

2.6 Neighborhood search

In order to start with a pool of columns that could potentially belong to a good solution, as well as for the improvement of the best obtained route in case the algorithm gets stuck in a local optimun, we apply a neighbourhood search algorithm.

K-Regret construction heuristic

In the general case when we have to complete a given route which has a set of unrouted customer locations, we apply the to this route K-Regret construction heuristic. To start with, we define a vertex of the current route which does not have outgoing arc as c_l, the next vertex that we consider to add to the current route after c_l as v, and we specify K as the number of intermediate vertices that we try to add to the current route before adding vertex v in order to compute insertion regret value I_r. After that, we define insertion regret value I_r as the sum of the costs of travelling from vertex c_l through all added intermediate vertices to vertex v. Next, we iterate through all unrouted vertices and for each of them compute insertion regret value I_r. Finally, we select vertex v' with the maximal I_r because this value is exacltly the measure of how much we would regret if we did not add v' after c_l.

Initially, we start with $K = 1$. If at some point we see that we can not add vertex v' because the resulting route is not feasible then we make one step back, that is we remove last added vertex and try to add v' again. We repeat this reverse movement until we either add v' and get a feasible route or we end up with all vertices removed. In the latter case we conclude that it is not possible to get a feasible route.

While moving backward to get a feasible route we collect all the vertices that we removed. We do this for the sake of identifying the reverse patterns. If for the current route we experience the same reverse pattern more than once then we increase the value of K as $K = K + 1$ making our Regret algorithm adaptive until we get K equal to 3. In the latter case we conclude that it is not possible to get a feasible route.

If there are more than one vertices of the current route which do not have outgoing arc then we apply the described algorith to each of them within the same iteration and get v' along with one of the c_l to extend the current route.

Initial route construction

In case if the initial route is not given, that is we have to construct a whole route through all the customer locatios for the sake of populating the initial column pool Ω', we apply K-Regret construction heuristic to the empty route.

Shaking

If we see no improvements of the Branch-and-price algorithm result during two consecutive iteration, that is no new promising columns which can be identified, then we apply a shaking procedure to the best feasible route we have by that moment. In this procedure some vertices are removed from this route with the following application of the K-Regret construction heuristic to each of the resulting incomplete routes.

We use three removal patterns. To start with, we remove a delivery vertex and try to move this vertex toward the begining of the current route until it immediately follows the corresponding pick-up vertex. Next, we iteratively remove up to $min(n, 5)$ consecutive vertices starting from each position $i \in 2..2n - 2$. Finally, we iteratively remove up to $min(n, 3)$ consecutive vertices starting from two different non-overlapping positions. The last strategy is the most destructive so it increases the chances of escaping from current local optimum. All the new constructed routes are collected in a sorted by route cost queue from which a route with the minimal cost is selested as the new best feasible route.

We repeat this shaking three times in a row for each its application, that is we potentially start from at most three new best feasible routes.

2.7 Uncertain travel and service times

Because of the probabilistic nature of real life transportation it is no longer possible to simply classify an arbitrary route as being either feasible or not. The reason for this is because travel and service times are uncertain so that the time of vising a particular customer is in reality a random variable. Thus, we need another criteria of the acceptance of a route. Let us denote the arrival time at location i following route r as Y_r^i, and the set of all locations of route r as N_r. Given these definitions, we formulate that route r is acceptable if the probability of visiting each of its locations i before the end of corresponding time window is greater that service level $\mathcal{Y}$ (Jula et al., 2006):

$$P(Y_r^i \leq l_i) \geq \mathcal{Y} \quad \forall i \in N_r \qquad (2.7.1)$$

We consider the case when the distributions of travel and service times are known, do not depend on time (stationary) and have only mean and variance as the corresponding probability density function parameters. Let us define the function that incorporates the time window constraint as the following:

$$g(y_j) = \begin{cases} e_j | y_j < e_j \\ y_j | e_j \leq y_j \leq l_j \\ t_{max} | y_j > l_j \end{cases} \qquad (2.7.2)$$

Next, we denote the service time at location i as S_i; the travel time along arc (i, j) as X_{ij}; the arrival times at location j from another location i with and without time window restrictions as $\hat{Y}_j$ and Y_j as the following:

$$Y_j = \hat{Y}_i + S_i + X_{ij} \qquad (2.7.3)$$
$$\hat{Y}_j = g(Y_j) \qquad (2.7.4)$$

Provided with these definitions, for each location j we compute the mean $E[\hat{Y}_j]$ and variance $var(\hat{Y}_j)$ of the (actual) arrival time at this location j from location i as the following:

$$E[\hat{Y}_j] = \begin{cases} e_j | P(Y_j < e_j) \\ g(E[\hat{Y}_i] + E[S_i] + E[X_{ij}]) = E[\hat{Y}_i] + E[S_i] + E[X_{ij}] | P(e_j \leq Y_j \leq l_j) \\ t_{max} | P(Y_j > l_j) \end{cases} \qquad (2.7.5)$$

$$var(\hat{Y}_j) = \begin{cases} 0 | P(Y_j < e_j) \\ var(g(Y_j)) = var(Y_j) = var(\hat{Y}_i) + var(S_i) + var(X_{ij}) | P(e_j \leq Y_j \leq l_j) \\ 0 | P(Y_j > l_j) \end{cases} \qquad (2.7.6)$$

In formulas (2.7.5),(2.7.6) we assume that $\hat{Y}_i, S_i, X_{ij}$ are independent, and use the fact that $var(g(Y_j)) \approx (g^{'}(E[Y_j]))^2 var(Y_j)$ (Papanicolaou, 2009). Based on these definitions we extend the dynamic programming formulation provided for the deterministic case. To start with, we redefine the state at any location $i \in V$ as the following:

m_i	the mean value $E[\hat{Y}_j]$ of the arrival time at i
v_i	the variance $var(\hat{Y}_j)$ of the arrival time at i
$\hat{c}_i$	the reduced cost of reaching i
$\mathcal{V}_i$	the set of pick-up locations that have been visited before arriving at i
$\mathcal{O}_i$	the set of picked up customers that are on board at i
$\mathcal{S}_i$	the set of picking-up times for $\mathcal{V}_i$
s	the route starting time

After that, we extend the value fuction as the following:

$$V_i(m_i, v_i, \hat{c}_i, \mathcal{V}_i, \mathcal{O}_i, \mathcal{S}_i, s) =$$
$$min_{(i,j)\in A'}$$

$$\begin{aligned}&(j = (2n+2) \wedge \hat{c}_i \vee \\ &j \in V \setminus \{2n+2\} \wedge \\ &\quad [p^1 V_j(m_j^1, v_j^1, \hat{c}_j, \mathcal{V}_j, \mathcal{O}_j, \mathcal{S}_j^1, s^1) + \\ &\quad p^2 V_j(m_j^2, v_j^2, \hat{c}_j, \mathcal{V}_j, \mathcal{O}_j, \mathcal{S}_j^2, s^2) + \\ &\quad p^3 V_j(m_j^3, v_j^3, \hat{c}_j, \mathcal{V}_j, \mathcal{O}_j, \mathcal{S}_j^3, s^3)] \\ &)\end{aligned} \tag{2.7.7}$$

subject to

$$p^1 = P(Y_j < e_j) \tag{2.7.8}$$
$$m_j^1 = e_j \tag{2.7.9}$$
$$v_j^1 = 0 \tag{2.7.10}$$
$$p^2 = P(e_j \le Y_j \le l_j) \tag{2.7.11}$$
$$m_j^2 = m_i + E[S_i] + E[X_{ij}] \tag{2.7.12}$$
$$v_j^2 = v_i + var(S_i) + var(X_{ij}) \tag{2.7.13}$$
$$p^3 = P(Y_j > l_j) \tag{2.7.14}$$
$$m_j^3 = t_{max} \tag{2.7.15}$$
$$v_j^3 = 0 \tag{2.7.16}$$
$$\hat{c}_j = \hat{c}_i + \begin{cases} \tilde{c}_j - \beta_j | j \in P \\ \tilde{c}_j | j \in V \setminus P \end{cases} \tag{2.5.9}$$
$$\mathcal{V}_j = \begin{cases} \mathcal{V}_i \cup \{j\} | j \in P \\ \mathcal{V}_i | j \in V \setminus P \end{cases} \tag{2.5.10}$$
$$\mathcal{O}_j = \begin{cases} \mathcal{O}_i \cup \{j\} | j \in P \\ \mathcal{O}_i \setminus \{j-n\} | j \in D \\ \mathcal{O}_i | j \in V \setminus P \cup D \end{cases} \tag{2.5.11}$$
$$\mathcal{S}_j^k = \begin{cases} \mathcal{S}_i \cup \{m_j^k + E[S_j]\} | j \in P \\ \mathcal{S}_i | j \in V \setminus P \end{cases} \quad \forall k \in \{1,2,3\} \tag{2.7.17}$$
$$s^k = \begin{cases} m_j^k - E[X_{ij}] | i = 1 \\ s | i \neq 1 \end{cases} \quad \forall k \in \{1,2,3\} \tag{2.7.18}$$

stopping criteria

$$|\mathcal{O}_j| > Q \tag{2.5.15}$$
$$j \in P \wedge \mathcal{V}_i \cap \{j\} \neq 0 \tag{2.5.16}$$
$$j \in D \wedge \{j-n\} \notin \mathcal{O}_i \tag{2.5.17}$$
$$j \in D \wedge (\textstyle\sum_{k\in\{1,2,3\}}(p^k m_j^k) - \mathcal{S}_{j-n}) > L \tag{2.7.19}$$
$$j = (2n+2) \wedge \mathcal{O}_j \neq 0 \tag{2.5.19}$$
$$j = (2n+2) \wedge (\textstyle\sum_{k\in\{1,2,3\}}(p^k m_j^k) - s) > T \tag{2.7.20}$$
$$P(Y_j \le l_i) < \mathcal{Y} \tag{2.7.21}$$

Next, we adjust the dominance criterion for the stochastic case as the following:

$$m_{i_i}^1 \le m_{i_i}^2 \wedge v_{i_i}^1 \le v_{i_i}^2 \wedge \hat{c}_i^1 \le \hat{c}_i^2 \wedge \mathcal{O}_i^1 = \mathcal{O}_i^2 \qquad \mathcal{V}_i^1 = \mathcal{V}_i^2 \tag{2.7.22}$$
$$m_{i_i}^1 \le m_{i_i}^2 \wedge v_{i_i}^1 \le v_{i_i}^2 \wedge \hat{c}_i^1 \le \hat{c}_i^2 - \textstyle\sum_{j\in(\mathcal{V}_i^2 \setminus \mathcal{V}_i^1)} \beta_j \wedge \mathcal{O}_i^1 = \mathcal{O}_i^2 \qquad \mathcal{V}_i^1 \subset \mathcal{V}_i^2 \tag{2.7.23}$$

3. Results

The proposed algorithm is prototyped in Python and Gurobi is used as a linear solver. All the tests are performed on a PC with 4x Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz and 4Gb RAM.

Given a limited time frame to complete this research, we focus on the deterministic cases only, and we do not implement the full branch-and-price method but we couple the column generation procedure with local search. In order words, within each iteration we try to add new columns by solving the pricing subproblem and, if we are stuck with no improvements in two consecutive iterations, we apply shaking.

3.1 Random test PDPTW instance

First, we use a random PDPTW instance with 1 vehicle and 6 customers without capacity restrictions which is represented in Table 3. Columns x and y denote the spacial location of a vertex, d defines an absolute value of demand, q indicates whether a vertex is a pick-up(1), delivery(-1) or depot(0) location, while e and l are the earliest and the latest time moments when it is allowed to visit this vertex.

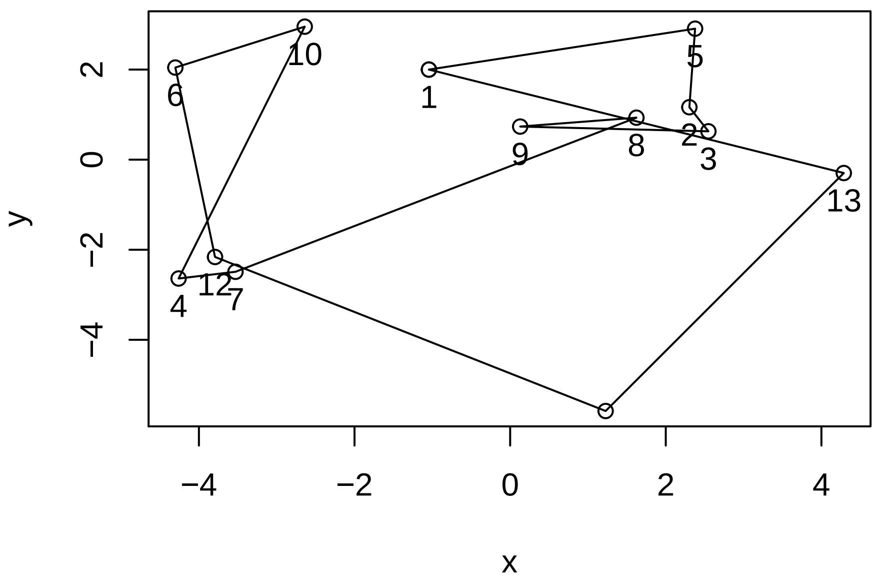
Table 3. Test instance data

	x	y	d	q	e	l
1	-1.044	2.000	0	0	0	1440
2	2.303	1.164	10	1	0	1440
3	2.548	0.629	10	1	0	1440
4	-4.261	-2.639	10	1	0	1440
5	2.377	2.908	10	1	147	177
6	-4.303	2.045	10	1	471	499
7	-3.530	-2.490	10	1	321	346
8	1.623	0.932	10	-1	260	276
9	0.129	0.735	10	-1	178	215
10	-2.640	2.953	10	-1	381	397
11	1.227	-5.581	10	-1	0	1440
12	-3.793	-2.161	10	-1	0	1440
13	4.288	-0.297	10	-1	0	1440
14	-1.044	2.000	0	0	0	1440

The aim of using this instance is to compare the results obtained from the three index pure MIP model (2.2) and the column generation procedure implementation at the early stages of development when we can not apply this implementation to the well known benchmark instances.

Both models give us a solution of 46.69 in less than a minute (Figure 2). We collect up to 50 new promising columns per iteration.

Figure 2: Test instance, solution 46.69(1)



3.2 Li & Lim's PDPTW lr101s instance

Next, we consider the lr101s PDPTW benchmark instance generated by Li & Lim (2001). This instance has 106 customer destinations and the best known solution is 1650.80(19 vehicles). The construction heuristic described in 2.6 gives us already a good starting solution of 2213.34(27 vehicles) (Figure 3) so that we get the best known solution (Figure 4) in 9 iterations (Figure 6). We see that the average reduced cost starts from around -30 and converges to 0 with some fluctuations in iterations $4-6$ (Figure 5). Similar to 3.1, we collect up to 50 new promising columns per iteration and the experiment takes about 15 minutes.

Figure 3: lr101s instance, initial solution 2213.34(27)

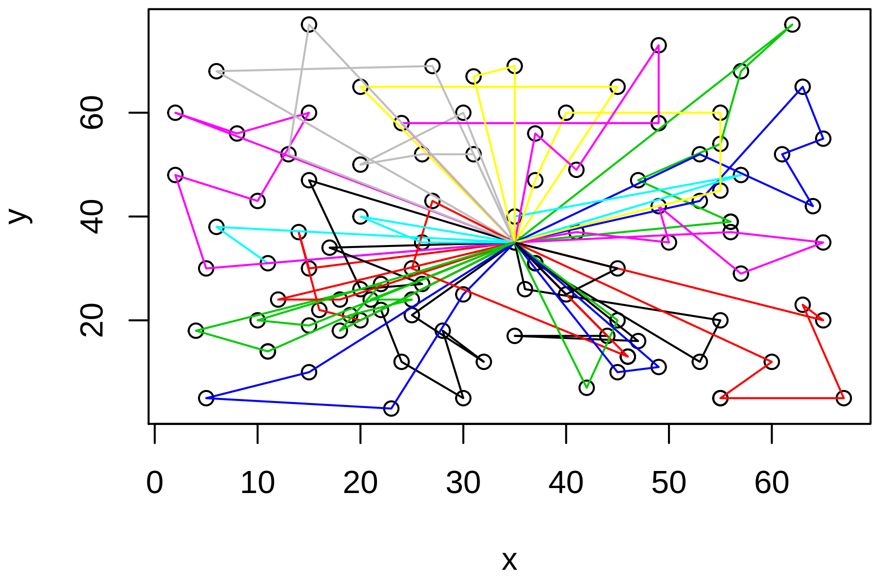


Figure 4: lr101s instance, solution 1650.80(19)

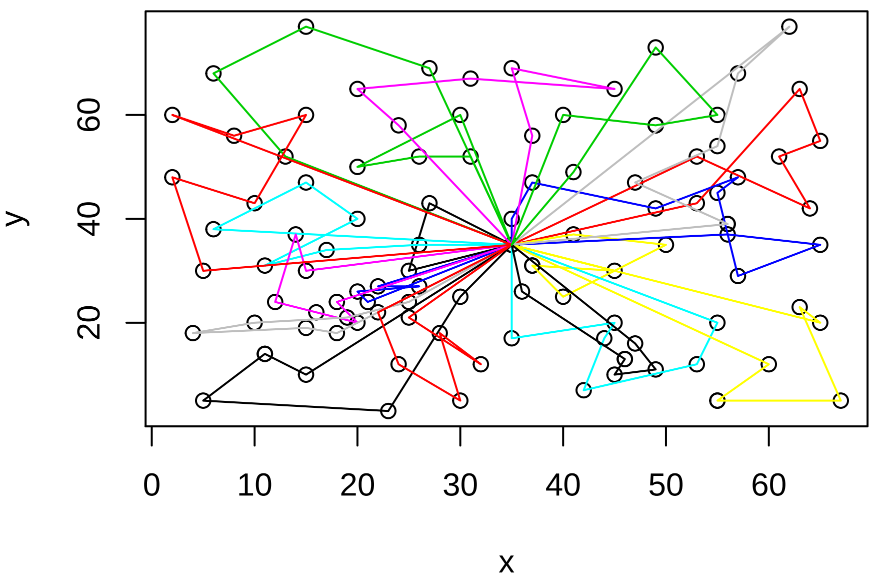


Figure 5: lr101s instance, reduced costs per iteration

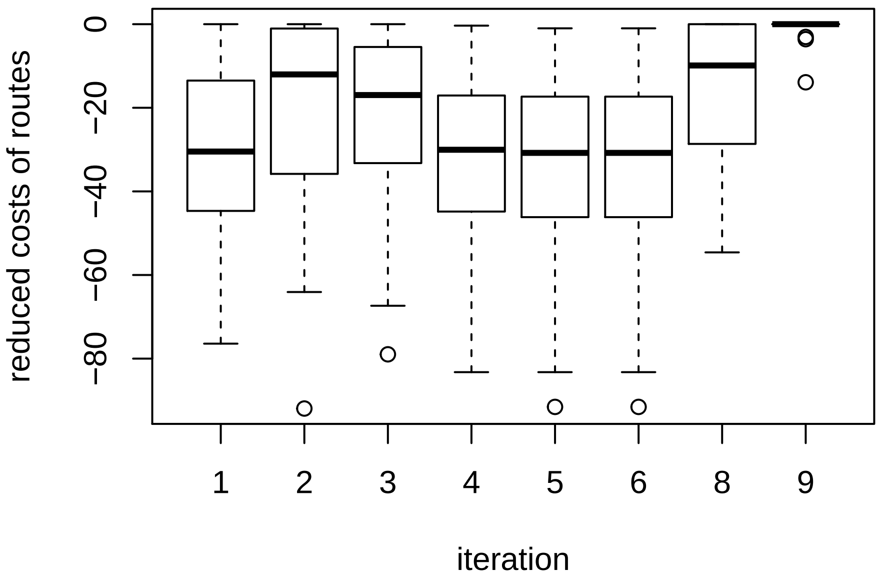
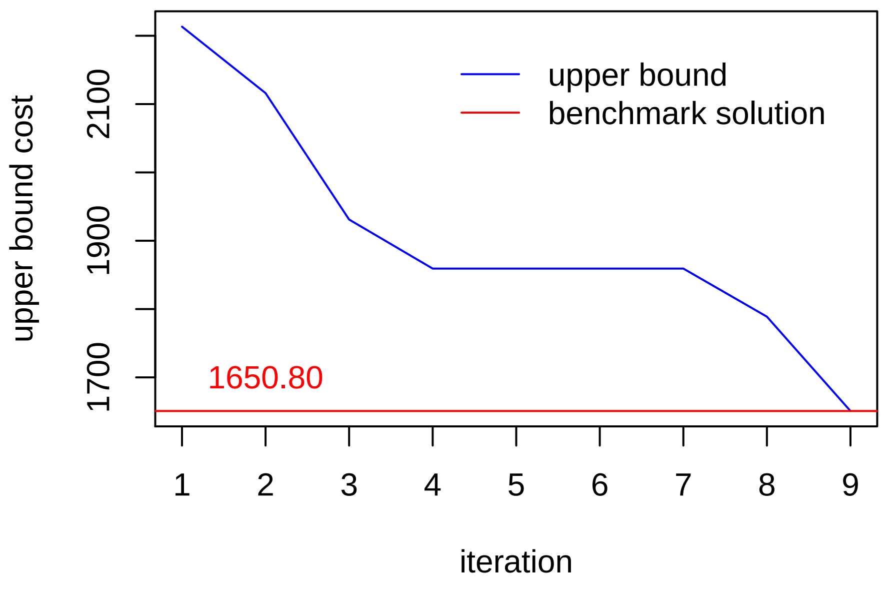


Figure 6: lr101s instance, upper bound per iteration



3.3 Li & Lim's PDPTW lc101s instance

After that, we consider another Li & Lim (2001) lc101s PDPTW benchmark instance. This instance also has 106 customer destinations and the best known solution is 828.94(10 vehicles). In contrast to the previous case it is clustered and, since our construction heuristic is not tailored to deal with such type of instances, this construction gives us a starting solution of just a moderate quality (Figure 7) which has almost three times larger cost (2094.97) compared to that one of the best known solution. We get the best known solution (Figure 8) in 11 iterations (Figure 10), and we observe that the average reduced cost starts from a much lower value around -200 with significant variance, but converges to 0 relatively fast (Figure 9). The fact that the initial solution is worse leads to that, in contrast to 3.1 and 3.2, we increase the number of gathered new promising columns per iteration up to 280 and the experiment takes about 30 minutes.

Figure 7: lc101s instance, initial solution 2094.97(17)

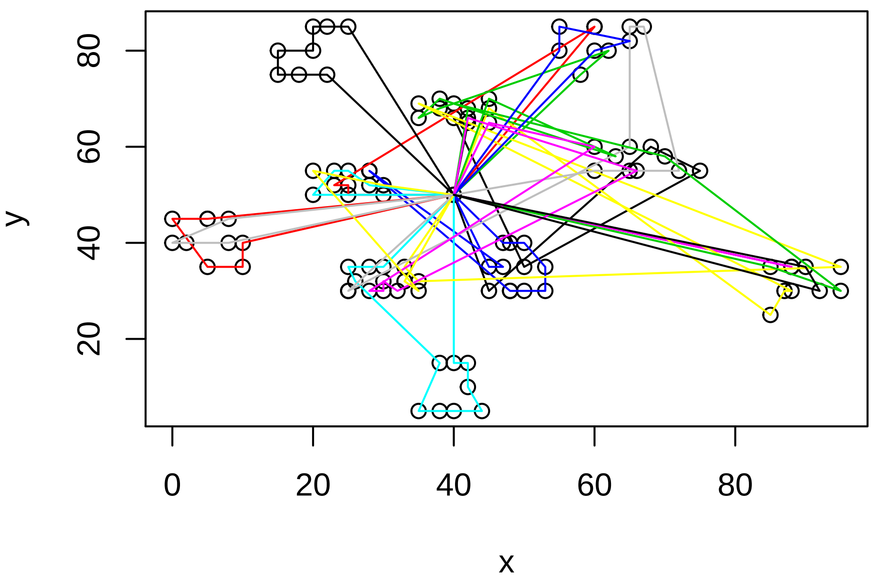


Figure 8: lc101s instance, solution 828.94(10)

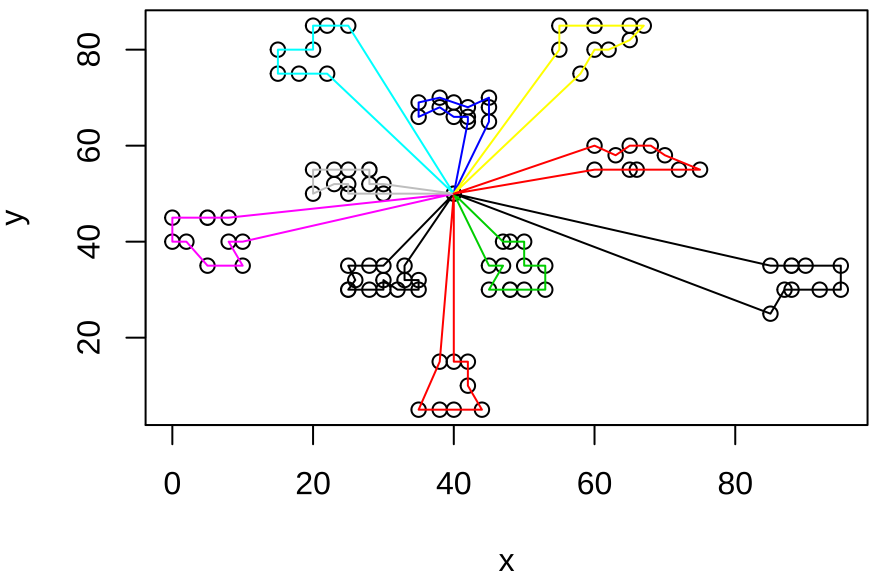


Figure 9: lc101s instance, reduced costs per iteration

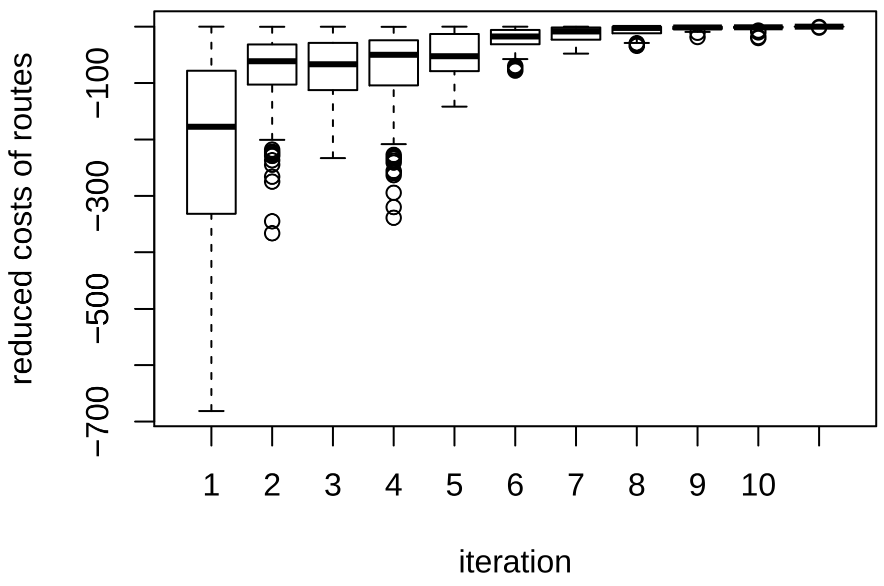
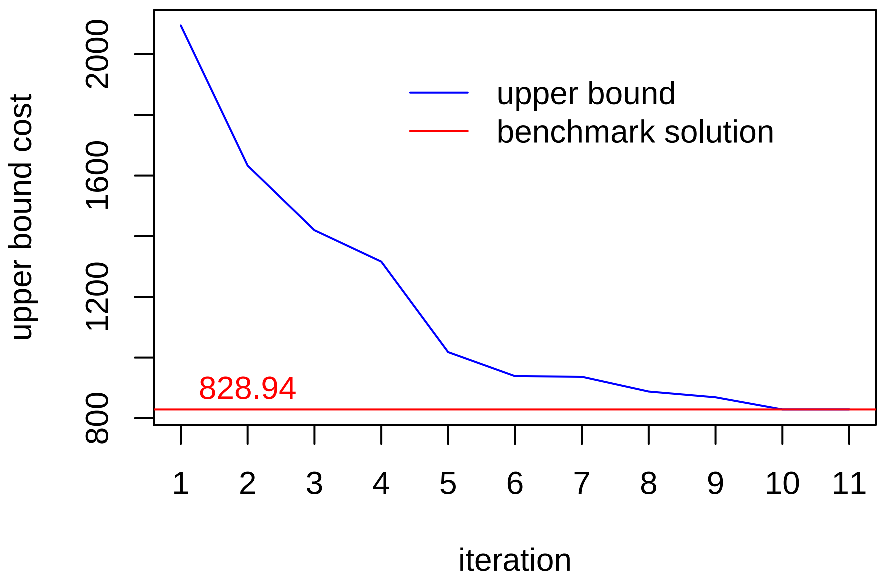


Figure 10: lc101s instance, upper bound per iteration



4. Conclusion

From this study we conclude that, indeed, the branch-and-price algorithm in general, and in particular the column generating procedure has its potential to be applied to the deterministic cases of the VRP(PDPTW). Despite the fact that we only prototyped this method and there are many ways to improve this implementation, we see that it is capable of solving instances of a moderate size in a reasonable amount of time.

Next to that, we report that both construction and improvement heuristics play an essential role in this method. Indeed, the quality of initial solution affects considerably the following computational effort that we need to get a good result. Furthermore, in some cases it is even impossible to get a solution of a good quality without decent starting point. Moreover, the branching part does not work well enough, that is does not significantly improve a solution, and, therefore, it is replaced by a local search heuristic in order to escape from local optimum. As a result, we come to the conclusion that the hybrid approach based on the combination of column generation and local search is the most promising way to proceed further.

As far as robustness is concerned, this method is to a high extent parameter dependent. In fact, there are many ways to tune it for a particular instance such as the number of new promising columns collected per column generation iteration, the number of Local TSP combinations, the size of NG-neighbourhood, the way the vertices are ordered during both the initial construction and pricing, and the shaking parameters. Therefore, a separate study is required to determine the correlation between instance type/size and the most appropriate parameters values. This study might result in an implementation of meta-algorithm on top of the current method to make the whole approach more adaptive.

We studied the theory related to the application of this method to the instances with uncertain travelling and service times. Furthermore, we defined the acceptance criteria of a route in the presence of such uncertainty. We think that the proposed dynamic programming formulation can be extended to work with such routes. However, due to the limited time we did not implement this in our prototype so further research is required.

References

Azi, N., Gendreau, M. & Potvin, J.Y. (2012). A dynamic vehicle routing problem with multiple delivery routes, Annals of Operations Research, 199, 103–112.

Baldacci, R., Mingozzi, A. & Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. Operations Research, 59(5), 1269–1283.

Battarra, M, Cordeau, J.F. & Iori, M. (2014). Pickup-and-Delivery Problems for Goods Transportation. In Vehicle Routing Problems, Methods, and Applications (2nd ed). Toth, P. & Vigo, D. (eds.). Philadelphia: SIAM, 161-192.

Bertsimas, D. (1992). A vehicle routing problem with stochastic demand. Operations Research, 40, 574–585.

Campbell, A., Gendreau, M. & Thomas, B. (2011). The orienteering problem with stochastic travel and service times. Annals of Operations Research. 186(1). 61-81.

Christofides, N., Mingozzi, A. & Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical Programming 20, 255-282.

Cordeau, J-F. (2006). A Branch-and-cut algorithm for the dial-a-ride problem. Operations Research, 54, 573–586.

Dantzig, G.B. & Thapa, M. (2003). Linear Programming 2: Theory and Extensions. Springer-Verlag, 448.

Doerner, K.F. & Salazar-González, J.J. (2014). Pickup-and-Delivery Problems for People Transportation. In Vehicle Routing Problems, Methods, and Applications (2nd ed). Toth, P. & Vigo, D. (eds.). Philadelphia: SIAM, 193-212.

Gendreau, M., Guertin, F., Potvin, J.Y. and Taillard, É. (1999). Parallel tabu search for real-time vehicle routing and dispatching, Transportation Science, 33, 381–390.

Gendreau, M., Jabali, O. & Rei, W. (2014). Stochastic Vehicle Routing Problems. In Vehicle Routing Problems, Methods, and Applications (2nd ed). Toth, P. & Vigo, D. (eds.). Philadelphia: SIAM, 213-240.

Hempsch, C. & Irnich, S. (2008). Vehicle routing problems with inter-tour resource constraints. In The Vehicle Routing Problem: Latest Advances and New Challenges. Golden, B., Raghavan, S. & Wasil, E. (eds.). New York: Springer, 421-444.

Jula, H., Dessouky, M. & Ioannou, P.A. (2006). Truck route planning in nonstationary stochastic networks with time windows at customer locations. IEEE Transactions on Intelligent Transportation Systems, 7, 51–62.

Kenyon, A. & Morton, D. (2003). Stochastic Vehicle Routing with Random Travel Times. Transportation Science. 37(1). 69-82.

Laporte, G., Louveaux, F. & Mercure, H. (1992). The vehicle routing problem with stochastic travel times. Transportation Science. 26(3). 161-170.

Laporte, G. (2007). What you should know about the vehicle routing problem. Naval Research Logistics. 54(8). 811-819.

Laporte, G., Ropke, S. & Vidal, T. (2014). Heuristics for the Vehicle Routing Problem. In Vehicle Routing Problems, Methods, and Applications (2nd ed). Toth, P. & Vigo, D. (eds.). Philadelphia: SIAM, 87-118.

Lee, C., Lee, K. & Park, S. (2012). Robust vehicle routing problem with deadlines and travel time/demand uncertainty. Journal of Operational Research Society, 63(9), 1294-1306.

Li & Lim. (2001). Best Known Results for PDPTW 100-cases. accessed 12 February 2017, https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/100-customers/

Miller, S.J. (2007). An Introduction to Linear Programming. lecture notes, Brown University Mathematics Dept.

Papanicolaou, A. (2009). Taylor Approximation and the Delta Method. accessed 16 January 2017, http://web.stanford.edu/class/cme308/OldWebsite/notes/TaylorAppDeltaMethod.pdf

Parragh, S.N. & Cordeau, J-F. (2015). Branch-and-price for the truck and trailer routing problem with time windows. CIRRELT report 2015-54, submitted for publication.

Poggi, M., & Uchoa, E. (2014). New exact algorithms for the capacitated vehicle routing problem. In Vehicle Routing Problems, Methods, and Applications (2nd ed). Toth, P. & Vigo, D. (eds.). Philadelphia: SIAM, 59-86.

Rader, D.J. (2010). Deterministic Operations Research: Models and Methods in Linear Optimization. Wiley, 632.

Ropke, S. & Cordeau, J-F. (2009). Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows. Transportation Science, 43(3), 267-286.

Semet, F., Toth, P. & Vigo, D. (2014). Classical Exact Algorithms for the Capacitated Vehicle Routing Problem. In Vehicle Routing Problems, Methods, and Applications (2nd ed). Toth, P. & Vigo, D. (eds.). Philadelphia: SIAM, 37-58.

Sungur, I., Ren, Y., Ordóñez, F. & Dessouky, M. (2010). A model and algorithm for the courier delivery problem with uncertainty, Transportation Science, 44, 193–205.

Taniguchi, E. & Shimamoto, H. (2004). Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times, Transportation Research Part C: Emerging Technologies, 12, 235–250.

Tas, D., Dellaert, N., van Woensel, T. & de Kok, T. (2013). Vehicle routing problem with stochastic travel times including soft time windows and service costs. Computers & Operations Research. 40. 214-224.