A robust approach for solving a vehicle routing problem with time windows with uncertain service and travel times

Mehdi Nasri*, Abdelmoutalib Metranea,b, Imad Hafidi and Anouar Jamali

*National School of Applied Sciences Khouribga, Sultan Moulay Slimane University, Beni-Mellal, Morocco
bMohamed 6 Polytechnic University, Ben Guérir, Morocco

The main purpose of this paper is to study the vehicle routing problem with hard time windows where the main challenges is to include both sources of uncertainties, namely the travel and the service time that can arise due to multiple causes. We propose a new resolution approach for the robust problem based on the implementation of an adaptive large neighborhood search algorithm and the use of efficient mechanisms to derive the best robust solution that responds to all uncertainties with reduced running times. The computational experiments are performed and improve the objective function of a set of instances with different levels of the uncertainty polytope to obtain the best robust solutions that protect from the violation of time windows for different scenarios.

1. Introduction

Since the pioneer paper of Dantzig and Ramser (1959) on the truck dispatching problem appeared at the end of the fifties of the last century, work in the field of the vehicle routing problem (VRP) has increased exponentially. Using a method based on a linear programming formulation, the authors of this work produced by hand calculations a near optimal-solution with four routes of a fleet of gasoline delivery trucks between a bulk terminal and twelve service stations supplied by a terminal. Nowadays, vehicle routing problem is considered as one of the most outstanding research achievements in the story of operations research and particularly in practice. There are important advances and new challenges that have been raised during the last few years such as radio frequency identification, and parallel computing (e.g. Pillac et al., 2013; Montoya-Torres, 2015). The class of VRP problems involves minimizing a travel distance of vehicles, starting and ending from a depot, to serve some customers. Typically the solution has to obey several other constraints, such as the consideration of travel, service, and waiting times together with time-window restrictions. This variant is called in the literature vehicle routing problem with time windows (Bräysy & Gendreau, 2005; Kallehauge et al., 2005; Rincon-Garcia et al., 2015).
For instance, three types of solution approaches can be used to solve these types of problems. First, the exact methods assert that the optimal is found if the method is given sufficiently in time and space. We cannot expect to construct exact algorithms which solve NP-hard problems. Second, the heuristics are solution methods that can quickly achieve a feasible solution in a reasonable quality. A special class called metaheuristics provides a high quality solution (Labadie et al., 2016). The third class of solutions is also a special class of heuristic which provides a guarantee on the error combined with near-optimal solution. An interesting topic on solving VRP consists in considering parameters affected by uncertainty, making the problem more realistic.

Different approaches have been proposed to deal with uncertainties in a VRP problem either in demand, travel time and/or service time. Among them, the stochastic approaches of vehicle routing problem SVRP have been treated in series of papers (Dror et al., 1993; Dror & Trudeau, 1986; Gendreau et al., 1996). The aim of the SVRP methodology is to find a near-best solution of the objective function responding to all possible data uncertainty. An alternative approach to handle the uncertain parameters is the robust optimization in which one can optimize against the worst scenario that can be generated from the source of uncertainty by using bi-objective function (Yousefi et al., 2017) and is immunized against this uncertainty (Sungur et al., 2008). In this context, the literature coasts a large number of applications such as scheduling (Goren & Sabuncuoglu, 2008; Hazir et al., 2010), facility location (Minoux, 2010; Baron et al., 2011; Alumur et al., 2012; Gülpinar et al., 2013), inventory (Bienstock & Özbay, 2008; Ben-Tal et al., 2009), finance (Fabozzi et al., 2007; Gülpinar & Rustem, 2007; Pinar, 2007). In particular, the authors proposed a mathematical model for the robust optimization with uncertain demands (Moghaddam et al., 2012), heterogeneous fleet (Noorizadegan et al., 2012), routing with capacity (Sungur et al., 2008; Gounaris et al., 2013) and rail shuttle Routing problem (Rouky et al., 2018). For instance, this is equivalent to the deterministic case studied by Miller-Tucker-Zemlin formulation of the used VRP. We refer the reader to an excellent survey and tutorial of the robust vehicle routing proposed by Ordóñez (2010). We note that uncertainty in travel cost could be handled using the robust combinatorial optimization approach. Wu et al. (2017) proposed a linear model evaluated on a set of random instances for the vehicle routing problem with uncertain travel time to improve the robustness of the solution which enhances its quality compared with the worst case on a majority of scenarios. In the same spirit, Toklu et al. (2013) treated the VRP problem with capacity and uncertain travel costs based on a variant of the ant colony algorithm to generate sets of solutions of uncertainty levels and to analyze their effects on the problem.

The stochastic approach is also applicable for the vehicle routing problem with time window constraints (VRPTW). Errico et al. (2016) formulated the VRPTW with stochastic service times as a set partitioning problem and solve it by exact branch-cut-and-price algorithms. More precisely, they elaborated efficient algorithms by choosing label components, developing lower and upper bounds on partial route reduced the cost to be used in the column generation step. Unlike this approach, robust optimization seeks to get good solutions for the VRPTW problem by only considering nominal values and deviations possible uncertain data. Many works tackled the vehicle routing problem with time windows and uncertain travel times (Sungur et al., 2008). Agra et al. (2012) presented a general approach to the robust VRPTW problem with uncertain travel times. Travel times belong to a demand uncertainty polytope, which makes the problem more complex to solve than its deterministic equivalent. The advantage of the addition in complexity is that the model from Agra et al. (2012) is more useful than the one from Sungur et al. (2008) and leads to less conservative robust solutions. Toklu et al. (2013) adapted their approach to solve the problem of VRPTW with uncertain travel times, whose objective was to minimize window time violation penalties by providing the decision makers a group of solutions found over several degrees of uncertainties considered. Agra et al. (2013) studied the VRPTW with uncertain travel times and proposed two robust formulations of the problem. The first extends the formulation of inequalities of resources and the second generalizes the formulation of inequalities of way. Their results show that the solution times are similar for both formulations while being significantly faster than the solutions times of a layered formulation recently proposed for the problem. Rouky et al. (2018) considered the vehicle routing
problem with time windows (VRPTW) by introducing both uncertainties: the travel times of locomotives and the transfer times of shuttle as a model of the Rail Shuttle Routing Problem (RSRP) in the Le Havre port. In order to solve this problem with uncertainties, the authors proposed the Robust Ant Colony Optimization (RACO). The aim of this paper is to study the similar robust VRPTW including both uncertainties in travel times and service times as in Rouky et al. (2018). Our contribution to all previous works lies, first on the choice of the Adaptive Large Neighborhood Search (ALNS) metaheuristic. Moreover, the numerical results were tested on a set of small instances based on the reference of Solomon benchmark, and large instances of Gehring & Homberger's benchmark. The studied problem generated different scenarios and each scenario is performed by using the best known sampling method, Metropolis version algorithm of Monte Carlo simulation. It is worth mentioning that the ALNS approach used three different removal operators to maintain the diversity during the searching process namely: proximity operator, route portion operator and longest detour operator, and one repair operator based on the greedy insertion. For a complete description of this ALNS approach including construction and destruction operators, we refer the reader to the third section.

The remainder of this paper is structured in the following way. First of all, we define the problem, and we introduce its mathematical formulation. Next, we present our robust approach that is meant to solve the problem including a presentation of the ALNS preprocessing, destruction and insertion heuristics. Moreover, we evaluate the new approach using both the exact algorithm and the best-known heuristics. A detailed computational and comparative study is presented in Section 4 in order to provide perfectly robust conclusions. Finally, some concluding remarks are discussed.

2. Problem statement

This section is devoted to the statement of the vehicle routing problem with time windows under travel time and service time uncertainty. First, we introduce the deterministic model of the VRP problem which consists of an optimization of the total distance traveled by all vehicles under four constraints. Next, the service at any customer starts within a given time interval and it is not allowed to arrive late. Furthermore, if the vehicle arrives too early at a customer, it must wait until the time window opens. Taking into consideration these two constraints on time windows we transform the VRP problem to its VRPTW variant. To complete our statement of the problem, we introduce the source of uncertainties namely travel times and service times which makes the problem harder to solve than its deterministic counterpart. We suggest a new formulation of the uncertainty that was inspired by the recent work of Rouky et al. (2018) where the travel times and service times uncertainties belongs to a convex polytope in the travel time interval \([t_{ij}, t_{ij} + \Delta_{ij}]\) and service time interval \([P_i, P_i + \delta_i]\), where \(t_{ij}\) and \(P_i\) denote the nominal values, \(\Delta_{ij}\) and \(\delta_i\) are the maximum positive perturbations. The complexity of this problem leads us to look for robust solutions and therefore to min-max the objective function, this is the last part of the state of the art of our problem. Now, in order to describe our problem, let us denote the set of nodes by \(N\), using \(i\) and \(j\) to denote general nodes, the depot will be denoted by \(o\). The set of arcs is denoted as \(A\) and contains pairs of nodes, \((i, j)\). The set of vehicles is called \(V\) with elements \(k\). Now we can assign to each edge \((i, j)\) a cost \(t_{ij}\), and to each node \(i\) a time window \([a_i, b_i]\). Then \(x_{ij}^k\) are binary decision variables that take the value 1 if vehicle \(k\) uses the edge \((i, j)\) and 0, otherwise. The deterministic model of the VRP can be stated as follows:

\[
\min \sum_{k \in V} \sum_{(i,j) \in A} x_{ij}^k t_{ij}
\]

subject to

\[
\sum_{k \in V} \sum_{j \in N} x_{ij}^k = 1 \quad \forall (i \in N)
\]
Each customer must be visited once, which is ensured by the first constraint. The second constraint ensures that each tour passes through the depot. The constraint (3) is a flow conservation constraint. Finally, the last constraint guarantees that each tour ends at the depot. Since the service time $P_i$ at any client $i$ by vehicle $k$ begins inside a given time interval $[a_i, b_i]$, we require an additional constraint.

$$a_i \leq P_{i}^k \leq b_i \forall (i \in N) \forall (k \in V).$$

The time windows considered here is hard, i.e. they cannot be violated, if the vehicle arrives earlier than required at a client $i$, it must hold up until the time window $[a_i, b_i]$ opens and moreover it is not permitted to arrive late.

$$P_{i}^k + t_{ij} - P_{j}^k \leq M(1 - x_{ij}^k) \forall (i \in N) \forall (j \in N\{0\}) \forall (k \in V),$$

where $M$ is a great value. To model the uncertainty in travel times and service times in the presence of time windows, a step-wise (layered) formulation is used. As mentioned above, the sets of uncertainties related to these times are described as follows:

$$U_t = \left\{ \tilde{t} \in \mathbb{R}^{|A|} / \tilde{t}_{ij} = t_{ij} + \Delta t_{ij}, \sum_{(i,j) \in A} \epsilon_{ij} \leq \Gamma, 0 \leq \epsilon_{ij} \leq 1, \forall (i,j) \in A \right\},$$

and

$$U_p = \left\{ \tilde{P} \in \mathbb{R}^{|N|} / \tilde{P}_i = P_i + \delta_i \omega_i, \sum_{i \in N} \omega_i \leq \Lambda, 0 \leq \omega_i \leq 1, \forall (i \in N) \right\},$$

The uncertainty parameters $\Gamma$ and $\Lambda$ which vary respectively between 0 and $|N| + |V|$, and 0 and $|N|$, are called budgets of uncertainty. They are controlling the maximum number of travel times and service times.

- If $\Gamma=0$ and $\Lambda=0$, the robust case is equivalent to the deterministic case.
- If $\Gamma=|N| + |V|$ and $\Lambda=|N|$ then this is the worst case where all travel times and service times are supposed uncertain and take at once their maximum values.

Robust optimization seeks to obtain good solutions for all the possible realizations of the uncertainties without it being necessary to define the laws of probability and considering only the nominal values and the possible deviations of the uncertain data. The uncertainties are introduced in the cost function as follows:

$$\min \left( \sum_{k \in V} \sum_{(i,j) \in A} x_{ij}^k t_{ij} + \max_{\psi / |\psi| = \Gamma} \sum_{k \in V} \sum_{(i,j) \in \psi} x_{ij}^k \Delta_{ij} \right).$$

And the constraint (6) treating the time windows by this:

$$P_{i}^k + t_{ij} + \delta_i \nu_i^{\theta} + \Delta_{ij} \mu_{ij}^{\theta} - P_{j}^k \leq M(1 - x_{ij}^k) \forall (i \in N) \forall (j \in N\{0\}) \forall (k \in V), \forall (\theta \in N) |\theta| = \Lambda, \forall (\psi \subset A) |\psi| = \Gamma.$$
where $\nu_i^\theta$ and $\mu_j^\Psi$ are binary variables which only takes respectively two values: 1 if $i \in \theta$ and $(i, j) \in \Psi$ and 0 otherwise.

3. Robust approach for the VRPTW

Along these lines, we propose an adaptive large neighborhood search (ALNS) heuristic to integrate into our approach in order to deal with robust VRPTW. The proposed metaheuristic (ALNS) is an extension of the Large Neighborhood Search (LNS) heuristic, which was first introduced by Shaw (1998), ALNS is a metaheuristic proposed by Ropke and Pisinger (2006). It is a common technique used to enhance a locally optimal solution and can prevent getting stuck in premature convergence to local optima within tightly constrained search space. Given an initial solution obtained by a construction method, it is based on the idea of improving the initial solution by applying various destroy and repair operators to generate large neighborhoods through which the search space is explored (Palomo-Martínez Pamela et al., 2017). The ALNS has already been adapted to several transportation problems including vehicle routing (Ropke & Pisinger, 2006), arc routing (Angélica Salazar-Aguilar et al., 2012), inventory-routing (Coelho et al., 2012), and the reliable multiple allocation hub location problem (Chaharsooghi et al., 2017). The ALNS seems well-suited for the VRPTW, its power is manifested in the fact that each new solution is obtained by first removing a number of vertices, then re-inserting these vertices into the solution. ALNS was chosen because it outperforms other mono-objective algorithms applied to the same problem while keeping the simplicity and high performance that characterize local search algorithms.

3.1. Adaptive Large Neighborhood Search

We will now describe the ALNS that we have used in the present paper. We believe that ALNS can be applied to a large class of difficult optimization problems. In order to design an ALNS algorithm for a given optimization problem we need to:

- Choose a number of fast construction operators which are able to construct a full solution.
- Select a number of destruction operators. It might be sufficiently important to choose the destruction operators that are expected to work well with the construction operator, but it is unnecessary.

Here is the detailed algorithm:

<table>
<thead>
<tr>
<th>Algorithm 1 Adaptive Large Neighborhood Search</th>
</tr>
</thead>
</table>
| Construct a feasible solution $x$; set $x^* = x$
| Repeat
| Choose a destroy neighborhood $N^-$ and a repair neighborhood $N^+$ using roulette wheel
| selection based on previously obtained scores $\pi_j$
| Generate a new solution $x'$ from $x$ using the heuristics corresponding to the chosen destroy
| and repair neighborhoods
| If $x'$ can be accepted then
| $x = x'$
| End if
| If $f(x') < f(x)$ then
| $x^* = x'$
| End if
| Update scores $\pi_j$ of $N^-$ and $N^+$
| Until Stop criteria is met
| Return $x^*$ |
3.1.1 Initial solution generation

In order to deal with the initial solution, we apply a greedy algorithm, which will be used in the reconstruction phase of the ALNS. This operator aims to insert the non-inserted nodes by testing the different possible configurations and then giving a feasible solution. It is not necessary that the completion time of the initial solution be minimal, as this solution will be further enhanced using the ALNS method.

3.1.2 Solution destruction

During the destruction phase, we put forward three different removal methods to maintain the diversity during the searching process and to define the neighborhood to explore at each iteration. Each removal method aims to remove a predefined number of nodes. The first operator is known as proximity operator. Its objective is to select close clients according to a spatio-temporal measure (Shaw, 1998), and then remove clients engendering the higher value of this measure. Using the same technique, the route portion operator comes to give more flexibility than the proximity operator to change the routes. The principle consists in choosing a pivot client owned to a road and remove it as well as its adjacent clients. Then, we calculate the spatio-temporal measure, with the objective to select a second client belonging to another route but close to the initial pivot. The second pivot is removed from the solution as well as its adjacent clients and so on until all clients will be removed. The third operator is referred to as longest detour operator. The interest of this operator is to remove the customers that lead to the largest cost increases for servicing them. For more details, we refer the reader to (PrescottGagnon et al., 2009). The algorithms of the used destroy operators can be found in Annex.

3.1.3 Solution reconstruction

Solomon's insertion heuristic (1987) presented a technique for choosing the new customer to be inserted into a route using two criteria \( c_1(i, u, j) \) and \( c_2(i, u, j) \) to select customer \( u \) for insertion between adjacent clients \( i \) and \( j \) in the current partial route. The primary criterion \( c_1 \) calculate the best feasible insertion place in the current route for each unrouted client \( u \) as

\[
c_1(i(u), u, j(u)) = \min_{\rho=1,...,m} c_1(i_{\rho-1}, u, i_{\rho}).
\]

The second criterion \( c_2 \) selects the new inserted customer:

\[
c_2(i(u^*), u^*, j(u^*)) = \max_u \{ c_2(i(u), u, j(u)) \mid u \text{ is unrouted and route is feasible} \}.
\]

Customer \( u^* \) is then inserted into the route between \( i(u^*) \) and \( j(u^*) \). The measurement of an insertion place \( c_1 \) depends on factors: the increase in total distance of the current route after the insertion, and the delay of service start time of the customer following the new inserted customer. To be more precise, \( c_1(i, u, j) \) is calculated as:

\[
c_1(i, u, j) = \alpha_1 \left( d_{iu} + d_{uj} - \mu d_{ij} \right) + \alpha_2 \left( b_{ju} - b_j \right),
\]

where \( d_{iu} + d_{uj} \) is the new distance between two nodes \( i \) and \( j \) after the insertion, \( b_j \) is the previous service start time, \( d_{ij} \) is the old distance between \( i \) and \( j \) and \( b_{ju} \) is the new service start time of customer \( j \) after the insertion of customer \( u \). The criterion \( c_2(i, u, j) \) is calculated as following

\[
c_2(i, u, j) = \lambda d_{0u} - c_1(i, u, j), \lambda \geq 0
\]

where the parameter \( \lambda \) is used to define how much the best insertion place for an unrouted customer depends on its distance from the depot and extra time required to visit the customer by the current vehicle.
3.1.3 Roulette wheel
For each iteration of the destruction phase, a roulette-wheel procedure is applied to select a method for generating the neighborhood (nodes to be removed). During the search process, the ALNS maintains a score $\phi_j$ which measures the best performance of an heuristic $j$ in the past iterations. The roulette wheel selection consists in selecting an heuristic $j$ with a probability $\frac{\phi_j}{\sum \phi_i}$. During $M$ iterations, the score $\phi_i$ is reset and the probabilities of choosing an heuristic are recalculated (Pisinger and Ropke (2007)).

3.2 ALNS applied to the robust VRPTW
In this section, we apply the adaptive large neighborhood search (ALNS) to solve the VRPTW, assuming that the displacement and the service times are both objects of uncertainty. The robustness of this approach has been tested on several scenarios generated by the Monte Carlo tool of simulation. We will now describe how we have adapted the general ALNS to the robust VRPTW.

We consider that the uncertainties in travel times and service times are independent and are budgeted by a pair of degrees of robustness $(\Lambda, \Gamma)$. In other words, $\Gamma$ and $\Lambda$ are interpreted as the maximum number of travel times and service times that can deviated from their nominal values, and are bounded respectively in the intervals $[0, |N| + |V|]$ and $[0, |N|]$. Therefore, our objective is to find for each $(\Lambda, \Gamma)$ scenario a robust solution which minimize the total delays, or immunize against the violation of time windows.

Our algorithm is presented in detail in the subsections: (3.2.1), (3.2.2) and (3.2.3). Here are a few notations used in our Algorithm:

$R^A,\Gamma_N$: A possible realization.
$S_{best}$: Best robust solution.
$S_N$: Solutions achieved at the $N^{th}$ realization.
TotalCost(.): A function used to calculate the total time of displacement of a solution
WorstEval$^\Gamma(.)$: A function used to calculate the worst evaluation of a solution
$Tr_k = (c_1 = o, c_2, ..., c_n = o)$: The tour of the vehicle $k$
$\sigma_l = (c_1, c_2, ..., c_l)$: A path of the tour $Tr_k$
$ArcSet^{\Gamma}$: The whole of the arcs which have the more large deviations of travel time
$NodeSet^{\Lambda}$: The whole of the nodes which have the $\Lambda$ more large deviations of service time
$\xi(\sigma_l) = \{c_1, c_2, ..., c_l\}$: All of the nodes which constitute $\sigma_l$
$Arc(\sigma_l) = \{y_1 = (c_1, c_2), y_2 = (c_2, c_3), ..., y_{l-1} = (c_{l-1}, c_l)\}$: The whole of the arcs which constitute the path $\sigma_l$
$(s_l^{k-})$: The maximum date of arrival of the vehicle $k$ at customer $c_l$

Here is the detailed algorithm of our approach:
Algorithm 2 The robust approach Algorithm

Parameters: Set Solutions, Set realizations

Outputs: Solution solution
realizations ← MonteCarlo()

For each realization ∈ realizations do
  solution ← ALNS(realization)
  solutions.add(solution)
End for

For each solution ∈ solutions do
  If checkRobustness(solution) ≠ True then
    solutions.remove(solution)
    Return NULL
  End if
  If EvalWorstCase(solution) ≠ True then
    solutions.remove(solution)
    Return NULL
  End if
End for

solution ← MinObjective(solutions)

Return solution

3.2.1 Identification of scenarios

At this step, we performed the Metropolis Monte-Carlo sampling in order to generate a reduced set of possible realizations $R_N^{\Gamma}$ from uniform distribution. For the sake of clarity, a realization $R_N^{\Gamma}$ represents a possible scenario in which $\Gamma$ displacement times take their maximum values $t_{ij} + \Delta_{ij}$, and $\Gamma$ service times take their maximum values $P_i + \delta_i$. While the other arcs and the other nodes take respectively their nominal values $t_{ij}$ and $P_i$. For more details about Metropolis-Monte-Carlo techniques, we refer the reader to the work of Mosegaard et al. (1995) and more recent version of Rubinstein et al. (2016).

3.2.2 Research of solution

For each realization $R_N^{\Gamma}$, we apply the Adaptive large neighborhood search in order to obtain a feasible solution noted $S_N$ satisfies each scenario that we have already generalized by Monte-Carlo.

3.2.3 Robustness

We draw up different mechanisms for the study of the robustness of the solution obtained by using ALNS approach in the previous step. The first mechanism of robustness allowed to verify the feasibility of our solution by investigating the related time windows of each visited customer. It is not required at this level to test all possible realizations. The pseudo code of this method is shown in the algorithm 3. The second mechanism concerns the evaluation of robustness of the solution according to a defined robust criterion. The best case criterion is considered as a min-min strategy that tries to find the best solution overall scenarios. At the opposite, the worst case criterion provides a solution minimizing the worst deviation among all considered scenarios (e.g. Wu 2017). In between, an alternative robust criterion is related to as min-max deviation; more detail can be found in the paper of Aissi et al. (2009).

In the present paper, we are more interested in finding the best solution in the presence of the worst case scenario for VRPTW under uncertain travel and service times (see algorithm 4). For the full details of different steps of algorithm 3 and 4, we refer the reader to the work of Rouky et al., (2018).
Algorithm 3 Check of the robustness

\begin{align*}
\text{feasible} & \leftarrow True \\
\text{For } k & \leftarrow 1 \text{ to } |V| \text{ do} \\
& \text{For } i \leftarrow \xi(T_i) \text{ to } |\Gamma_i(T_i)| \\
& \quad \text{Calculate } \text{ArcSet}^\Gamma \text{ and } \text{NodeSet}^A \\
& \quad \text{For } \lambda \leftarrow \text{ to } l - 1 \text{ do} \\
& \quad \quad \text{If } l \leq \Gamma + 1 \text{ or } y_\lambda \in \text{ArcSet}^\Gamma \text{ then} \\
& \quad \quad \quad t_{y_\lambda} \leftarrow t_{y_\lambda} + \Delta y_\lambda \\
& \quad \text{End if} \\
& \text{End for} \\
& \text{For } i \leftarrow \text{ to } l - 1 \text{ do} \\
& \quad \text{If } l \leq \Lambda \text{ or } c_i \in \text{NodeSet}^{A_{i-h-1}} \text{ then} \\
& \quad \quad p_{c_i} \leftarrow p_{c_i} + \delta c_i \\
& \quad \text{End if} \\
& \text{End for} \\
& (s_i^{1-k})^{\Gamma_A} \leftarrow \\
& \text{For } i \leftarrow \text{ to } l \text{ do} \\
& \quad (s_i^{1-k})^{\Gamma_A} \leftarrow \max ((s_i^{1-k})^{\Gamma_A} + t_{y_{i-1}} + p_{c_{i-1}} + a_{c_i}) \\
& \text{End for} \\
& \text{If } (s_i^{1-k})^{\Gamma_A} > b_{c_i} \text{ then} \\
& \quad \text{feasible takes false and the algorithm ends} \\
& \text{End if} \\
& \text{End for} \\
& \text{End for} \\
\end{align*}

Algorithm 4 Evaluation on the worst case

\begin{align*}
\text{WorstEval}^\Gamma(S_N) & \leftarrow \\
& \text{Put in descending order all the arcs of } \gamma(S_N) \text{ according to their maximum deviations.} \\
\text{For } i & \leftarrow 1 \text{ to } |\Gamma(S_N)| \text{ do} \\
& \quad \text{WorstEval}^\Gamma(S_N) \leftarrow \text{WorstEval}^\Gamma(S_N) + t_{y_i} + \Delta y_i \\
\text{End for} \\
\text{For } i & \leftarrow \Gamma + 1 \text{ to } |\gamma(S_N)| \text{ do} \\
& \quad \text{WorstEval}^\Gamma(S_N) \leftarrow \text{WorstEval}^\Gamma(S_N) + t_{y_i} \\
\text{End for} \\
\text{Return } \text{WorstEval}^\Gamma(S_N) \\
\end{align*}

4. Computational experiment

Since VRPTW and RVRPTW are both NP-Hard, so to provide perfect conclusions and comparative results, we considered several kinds of instances. The robust approach examined was tested on a set of small instances based on the reference of Solomon benchmark (1987) (Solomon, 1987), and large instances of Gehring & Homberger's benchmark. Since the uncertainty of RVRPTW is simulated by discrete scenarios using Monte-Carlo Simulation, the uncertain travel time of each arc and the uncertain service time at each node are generated randomly between 0 and 10, with (\(\Gamma, \Lambda\)) is the degree of robustness which represents the number of service times and the number of travel times assumed to be uncertain. The used instances are noted as follow Gr_\(\Gamma_A\_i\), where \(Gr = \{C1, C2, R1, R2, RC1, RC2\}\) corresponds to the class name of the benchmark of Solomon and Gehring & Homberger, and \(\Lambda\) represent the number of travel times and service times supposed uncertain. \(i = \{100,200,400,600,800,1000\}\) is an index that represents the size of the instance.
Table 1 shows the results obtained for small instances (100 customers) by using Cplex for the deterministic VRPTW and the results obtained by our robust approach based on ALNS that deals with the VRPTW considering that travel times and service times are both uncertain. The column “Instance” displays the notation of the instance. The column “Initial solution” presents the initial solution with which the robust approach starts. The column “best” states the best values found by the robust approach with 10 runs while the column “mean” shows the average values found by the robust approach over 10 trials. The column “Optimal” displays the optimal solution for the deterministic VRPTW calculated by Cplex.

Table 1

<table>
<thead>
<tr>
<th>Instance</th>
<th>Initial solution</th>
<th>Best solution</th>
<th>Mean solution</th>
<th>Optimal solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>R101_10_10_100</td>
<td>2176.39</td>
<td>1918.56</td>
<td>1981.45</td>
<td>1637.7</td>
</tr>
<tr>
<td>R106_10_10_100</td>
<td>1794.75</td>
<td>1570.49</td>
<td>1603.24</td>
<td>1234.6</td>
</tr>
<tr>
<td>R112_10_10_100</td>
<td>1292.24</td>
<td>1135.25</td>
<td>1165.39</td>
<td>978.7</td>
</tr>
<tr>
<td>R201_10_10_100</td>
<td>1751.13</td>
<td>1534.99</td>
<td>1544.19</td>
<td>1143.2</td>
</tr>
<tr>
<td>C101_10_10_100</td>
<td>917.76</td>
<td>870.46</td>
<td>881.17</td>
<td>827.3</td>
</tr>
<tr>
<td>C105_10_10_100</td>
<td>1016.49</td>
<td>872.35</td>
<td>906.45</td>
<td>827.3</td>
</tr>
<tr>
<td>C108_10_10_100</td>
<td>1172.46</td>
<td>982.18</td>
<td>1016.33</td>
<td>827.3</td>
</tr>
<tr>
<td>C201_10_10_100</td>
<td>782.39</td>
<td>606.44</td>
<td>645.1</td>
<td>599.1</td>
</tr>
<tr>
<td>C205_10_10_100</td>
<td>805.15</td>
<td>662.08</td>
<td>713.74</td>
<td>586.4</td>
</tr>
<tr>
<td>C208_10_10_100</td>
<td>831.3</td>
<td>727.28</td>
<td>779.43</td>
<td>585.8</td>
</tr>
<tr>
<td>RC101_10_10_100</td>
<td>2193.94</td>
<td>1882.76</td>
<td>1994.25</td>
<td>1619.8</td>
</tr>
<tr>
<td>RC105_10_10_100</td>
<td>1915.73</td>
<td>1713.6</td>
<td>1589.1</td>
<td>1513.7</td>
</tr>
<tr>
<td>RC108_10_10_100</td>
<td>1710.58</td>
<td>1474.84</td>
<td>1562.7</td>
<td>1114.2</td>
</tr>
<tr>
<td>RC201_10_10_100</td>
<td>1659.86</td>
<td>1463.38</td>
<td>1509.13</td>
<td>1261.8</td>
</tr>
<tr>
<td>RC202_10_10_100</td>
<td>1566.17</td>
<td>1398.14</td>
<td>1448.56</td>
<td>1092.3</td>
</tr>
<tr>
<td>RC205_10_10_100</td>
<td>1518.52</td>
<td>1418.02</td>
<td>1453.03</td>
<td>1154</td>
</tr>
</tbody>
</table>

Table 2 shows the results obtained for large instances by comparing the best-known results for the deterministic VRPTW to the results found by our robust approach based on ALNS that deals with the VRPTW considering that travel times and service times are both uncertain.

Table 2

<table>
<thead>
<tr>
<th>Instance</th>
<th>Initial solution</th>
<th>Best solution</th>
<th>Mean solution</th>
<th>Best known</th>
</tr>
</thead>
<tbody>
<tr>
<td>R121_25_25_200</td>
<td>6610.35</td>
<td>5696.41</td>
<td>5911.84</td>
<td>4784.11</td>
</tr>
<tr>
<td>R141_25_25_400</td>
<td>12526.98</td>
<td>10939.12</td>
<td>11082.53</td>
<td>10372.31</td>
</tr>
<tr>
<td>R161_50_50_600</td>
<td>26176.96</td>
<td>24277.04</td>
<td>24910.6</td>
<td>21311.09</td>
</tr>
<tr>
<td>R181_50_50_800</td>
<td>41205.13</td>
<td>39011.14</td>
<td>39643.9</td>
<td>36852.06</td>
</tr>
<tr>
<td>R1101_100_100_1000</td>
<td>63493.03</td>
<td>60506.88</td>
<td>61702.03</td>
<td>53473.26</td>
</tr>
<tr>
<td>R125_25_25_200</td>
<td>5156.38</td>
<td>4667.38</td>
<td>4910.81</td>
<td>4107.86</td>
</tr>
<tr>
<td>R145_25_25_400</td>
<td>11714.54</td>
<td>11250</td>
<td>11665.48</td>
<td>9226.21</td>
</tr>
<tr>
<td>R165_50_50_600</td>
<td>22959.22</td>
<td>21231.41</td>
<td>21802.25</td>
<td>19588.89</td>
</tr>
<tr>
<td>R185_50_50_800</td>
<td>37521.62</td>
<td>36848.37</td>
<td>36983.03</td>
<td>33723.77</td>
</tr>
<tr>
<td>R1105_100_100_1000</td>
<td>58238.81</td>
<td>56015.9</td>
<td>56877.48</td>
<td>50876.21</td>
</tr>
<tr>
<td>R210_25_25_200</td>
<td>3919.05</td>
<td>3676.6</td>
<td>3731.17</td>
<td>3301.18</td>
</tr>
<tr>
<td>R1410_25_25_400</td>
<td>10201.39</td>
<td>9647.68</td>
<td>10121.67</td>
<td>8094.1</td>
</tr>
<tr>
<td>R1610_50_50_600</td>
<td>21602.23</td>
<td>19874.36</td>
<td>20530.31</td>
<td>17748.83</td>
</tr>
<tr>
<td>R1810_50_50_800</td>
<td>36258.1</td>
<td>34792.04</td>
<td>35587.88</td>
<td>31086.85</td>
</tr>
<tr>
<td>R11010_100_100_1000</td>
<td>52326.08</td>
<td>50678.87</td>
<td>50961.69</td>
<td>47992.05</td>
</tr>
<tr>
<td>R221_25_25_200</td>
<td>5399.58</td>
<td>4785.14</td>
<td>4879.5</td>
<td>4483.86</td>
</tr>
<tr>
<td>R241_25_25_400</td>
<td>11836.97</td>
<td>10442.34</td>
<td>10975.86</td>
<td>9210.15</td>
</tr>
<tr>
<td>R261_50_50_600</td>
<td>24045.81</td>
<td>22070.83</td>
<td>22702.28</td>
<td>18206.8</td>
</tr>
<tr>
<td>R281_50_50_800</td>
<td>34167.08</td>
<td>32289.08</td>
<td>33162.16</td>
<td>28114.25</td>
</tr>
<tr>
<td>R2101_100_100_1000</td>
<td>51402.42</td>
<td>48077.63</td>
<td>49298.56</td>
<td>42188.86</td>
</tr>
<tr>
<td>R225_25_25_200</td>
<td>4226.4</td>
<td>3625.7</td>
<td>3871.07</td>
<td>3366.79</td>
</tr>
<tr>
<td>R245_25_25_400</td>
<td>10150.65</td>
<td>8340.94</td>
<td>8986.2</td>
<td>7128.93</td>
</tr>
<tr>
<td>R265_50_50_600</td>
<td>18968.26</td>
<td>17022.95</td>
<td>17790.63</td>
<td>15962.6</td>
</tr>
<tr>
<td>R285_50_50_800</td>
<td>29807.24</td>
<td>28110.59</td>
<td>28519.41</td>
<td>24285.89</td>
</tr>
<tr>
<td>R2105_100_100_1000</td>
<td>48290.25</td>
<td>46144.16</td>
<td>46894.91</td>
<td>36232.18</td>
</tr>
<tr>
<td>R2210_25_25_200</td>
<td>3312.17</td>
<td>2913.82</td>
<td>3066.73</td>
<td>2654.97</td>
</tr>
<tr>
<td>R2410_25_25_400</td>
<td>8482.81</td>
<td>6679.72</td>
<td>7785.34</td>
<td>5786.4</td>
</tr>
<tr>
<td>R2610_50_50_600</td>
<td>15874.92</td>
<td>14609.57</td>
<td>15165.41</td>
<td>12253.47</td>
</tr>
<tr>
<td>R2810_50_50_800</td>
<td>25210.16</td>
<td>23839.6</td>
<td>24127.11</td>
<td>20401.47</td>
</tr>
</tbody>
</table>
In order to visualize the impact of increasing degrees of uncertainty on objective function, we set the value of $\Lambda$ to 25 and we adjusted the value of $\Gamma$ for multiple instances of size 100. Here is the curve obtained:
The graph shows clearly that the objective function increases according to degree of uncertainty. To the best of our knowledge, this contribution is the first work to be devoted to the study of VRPTW considering both the uncertainties on travel times and service times. Due to the lack of work in this direction, we compared our results with the deterministic VRPTW literature even if the two problems are different in the sense that a good solution found for the deterministic case could become worse in the presence of uncertainties or even unreachable.

From Table 1, the computation of the mean absolute percentage error (MAPE) of the data set indicates how much error in predicting the robust mean solution compared with the deterministic optimal solution. By including the uncertainties in travel times and service times, the minimization problem becomes robust and MAPE is about 22.68%. We need then extra cost to the objective function but the related counterpart risk gained is the protection against the delays and the solution is of good quality. In the meanwhile, for large instances, we concluded from table 2 that MAPE is about 22.57% where in this case the observed solution is the robust mean one compared to the deterministic case. The fact that the MAPE value for large and small instance around 23% is beyond our expectation and we believed that the robustness of the solution against the uncertain data can be achieved in the most cases by making the solution feasible for any travel times and service times in the uncertainty sets.

It is then clear that our approach is very powerful in terms of the robustness since it included several algorithms (Robustness verification, worst-case evaluation ...) which leads to near best solutions for all the possible realizations of the uncertainties without any further considerations but only nominal values and deviations possible uncertain data are sufficient.

5. Conclusion

Our main goal in this paper was to consider the robust vehicle routing problem with time windows under both travel times and service times uncertainties. For this purpose, a new resolution robust approach has been suggested to minimize the total distance of the travel time in the presence of the maximum deviations of possible uncertain data. In this contrast, we generated all possible scenarios by using Metropolis-Monte Carlo simulation and we opt for the adaptive large neighborhood search ALNS algorithm to solve each sub-problem related to each scenario. In this context, several destroy/repair method has been combined to explore multiple neighborhoods within the same search and defined implicitly the large neighborhood. In order to study the feasibility of the resulting solution, following the previous works, efficient mechanisms have been developed; the first concerns the verification of the robustness, while the second takes into consideration the evaluation of the solution on the worst case.
The introduction of an effective way of modeling and handling several uncertainty data levels defined by pairs of uncertainty \((A, I)\), which represent respectively the number of service times and the number of travel times assumed uncertain, has been tested on several sets of problems and showed improved robustness results for benchmark instances. Furthermore, the resolved method using the ALNS approach confirmed as in the previous work a great protection against uncertainties compared to what would have been found if a deterministic solution had been applied.

In this work, the computational experiments were performed to examine our proposed new approach compared to the deterministic VRPTW literature on a set of small instances based on the Solomon VRPTW benchmark and large instances of Gehring & Homberger benchmark. The results have shown the robustness of our solutions against delays and offer decision-making tool that allows choosing the level of protection, as well as the deterministic solution, is applied.

Future work will focus on the extension of the robust routing vehicle problem with time windows, in which both unexpected delays in travel time and service time may occur, to the application of parallel adaptive large neighborhood search in order to develop fast optimization procedures able to react in real time to changes in the problem information.

Acknowledgments

The first author is indebted to the Moroccan CNRST National Centre for Scientific and Technical Research for the partial funding of this research project.

References


## Appendices

We introduce some algorithms used for the three different destruction operators of our Adaptive Large Neighborhood Search (ALNS) to ensure the diversity of the searching process. In contrast of the LNS heuristic which uses only one destroy procedure and one repair procedure, the ALNS uses an adaptive layer with a set of removal and insertion heuristics and applies them by preference using a selection mechanism that considers the statistics obtained during the search based on their performance and past success. The detail of each algorithm is given as follows:

### Appendix A

<table>
<thead>
<tr>
<th>Algorithm 5</th>
<th>Proximity operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select randomly a node <code>firstToRemove</code> from the list of clients that have been removed, and add it to the list <code>removedNodes</code></td>
<td></td>
</tr>
<tr>
<td>Add the other nodes to <code>remainingNodes</code></td>
<td></td>
</tr>
<tr>
<td><strong>For</strong> <code>i = 1</code> <strong>to</strong> <code>numToRelax</code> <strong>do</strong></td>
<td></td>
</tr>
<tr>
<td>Choose randomly a node <code>removedNodeId</code> from <code>removedNodes</code></td>
<td></td>
</tr>
<tr>
<td>Choose by Rank and relatedness a node <code>nodeId</code></td>
<td></td>
</tr>
<tr>
<td>Add <code>nodeId</code> to <code>removedNodes</code></td>
<td></td>
</tr>
<tr>
<td>Remove <code>nodeId</code> from <code>remainingNodes</code></td>
<td></td>
</tr>
<tr>
<td><strong>End For</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Return</strong> <code>removedNodes</code></td>
<td></td>
</tr>
</tbody>
</table>
### Appendix B

**Algorithm 6** Route portion operator

| Choose the First node to delete and its adjacent and add them to `adjacents` list |
| Add the nodes of `adjacents` to `removedNodes` list and remove them from `remainingNodes` |
| Add the other nodes to `remainingNodes` |
| For | \( i = 1 \) to `numToRelax` do |
| | Choose randomly a node `removedNodeId` from `removedNodes` |
| | Choose by Rank and relatedness a node `nodeId` |
| | Add `nodeId` to `removedNodes` |
| | Remove `nodeId` from `remainingNodes` |
| End For |
| Return `removedNodes` |

### Appendix C

**Algorithm 7** Longest detour operator

| Inputs: feasible solution \( x \), Array `maxTab` |
| Outputs: \( z \) max detour |
| For | `route \in routes(x)` do |
| | \( m = \text{MaxDetour} \) |
| | Add \( m \) to `maxTab` |
| End for |
| \( z = \max(maxTab) \) |
| Return \( z \) |

© 2019 by the authors; licensee Growing Science, Canada. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).