symmetry

*Article*

# Multithreading Parallel Robust Approach for the VRPTW with Uncertain Service and Travel Times

**Mehdi Nasri *,†, Imad Hafidi † and Abdelmoutalib Metrane †**

National School of Applied Sciences Khouribga, Sultan Moulay Slimane University, 23052 Beni Mellal, Morocco; i.hafidi@usms.ma (I.H.); a.metrane@usms.ma (A.M.)
* Correspondence: mehdi.nasri@usms.ac.ma
† All authors contributed equally to this work.

**Abstract:** The objective of this paper is to consider the vehicle routing problem with time windows under two uncertainties: service and travel times. We introduce new resolution approaches for the robust problem and an efficient parallel procedure for the generation of all possible scenarios. The best robust solution of each scenario can be achieved by using a parallel adaptive large neighborhood search metaheuristic. Through our analysis, we expect to find the best compromise between the reduced running time and a best good solution, which leads to four distinct combinations of parallel/sequential approaches. The computational experiments are performed and tested on Solomon's benchmark and large randomly generated instances. Furthermore, our results can be protected against delay in service time in a reasonable running time especially for large instances.

**Keywords:** parallel multithreading; robust approach; ALNS; Monte Carlo; VRPTW

**MSC:** 90C17; 90B06; 90C59; 68W10; 65Y05

check for updates

**Citation:** Nasri, M.; Hafidi, I.; Metrane, A. Multithreading Parallel Robust Approach for the VRPTW with Uncertain Service and Travel Times. *Symmetry* **2021**, *13*, 36. https://dx.doi.org/10.3390/sym13010036

Received: 20 November 2020
Accepted: 15 December 2020
Published: 28 December 2020

## 1. Introduction

Over the past few decades, the Vehicle Routing Problem (VRP) and its variants have been the subject of massive investigations in operations research. This fact is due to the importance of its applications in different domains, such as logistics, supply chain management, scheduling, inventory, finance, etc. The main purpose of the vehicle routing problem is to find a set of least cost routes, beginning and ending at a depot, that together cover a set of customers (see, e.g., [1–3]). In real-world applications, several operational constraints must be taken into account, as for example considering the travel and service times with time-window limitations [4]. Then, the considered problem becomes the Vehicle Routing Problem with Time Windows (VRPTW) [5].

A challenging topic in solving the VRPTW problem consists of considering uncertain parameters. Different approaches have been proposed in order to handle uncertain events in a VRPTW, in demand, displacement time, and service time. From the literature, we distinguish between stochastic and robust approaches. The stochastic variant can be regarded as a methodology that aims at finding a near-best solution for the objective function responding to all uncertain events that are characterized by their probability distributions [6–8]. On the other hand, the purpose of the robust approach is to find a solution that protects against the impact of data uncertainties, taking into consideration several technical criteria challenges such as the worst case, best case, min-max deviation, etc. The choice of a mathematical model of uncertain data is a crucial step to provide robust solutions. This kind of approach was the subject of a series of papers (see, e.g., [9]). In this context, Rouky et al. [10] introduced the uncertainties to the travel times of locomotives and the transfer times of shuttles as a model of the Rail Shuttle Routing Problem (RSRP) at Le Havre port. The authors proposed the Robust Ant Colony Optimization (RACO) as an efficient technique to deal with the problem. In the same spirit, Wu et al. [11] proposed a robust model tested 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 in a majority of scenarios. For instance, we split the resolution methods of this last approach into two major categories: exact and heuristic methods.

Due to the NP-hard nature of such problems, we cannot expect to use exact methods for the resolution (see, e.g., [12,13]). Indeed, the heuristics are solution methods that yield very good solutions in a limited time at the expense of ensuring the optimal solution. A generic class called the metaheuristic is used to exploit the best capabilities to achieve better solutions to solve a wide range of problems, since the mechanism to avoid getting trapped in local minima is present. In this regard, the literature covers a considerable number of metaheuristics conceived of to solve the VRPTW such as Simulated Annealing (SA) [14,15], Variable Neighborhood Search (VNS) [16,17], Ant Colony Optimization Algorithm (ACO) [18,19], Genetic Algorithm (GA) [20], and Tabu Search (TS) [21–23].

The need for parallel computing becomes inevitable despite the good results obtained with metaheuristics, due to the huge scale of the input data and the unexpected way of its change, which makes the objective function time-consuming. However, it is important to notice that the quality of the solution can be influenced. For instance, the major challenge is to find a parallelization strategy that solves larger problem instances in reasonable computing times and offers a consistently high level of performance over a wide variety of problem characteristics. In this context, several authors have proposed parallel techniques to tackle the combinatorial problems. Following these ideas, Bouthillier et al. [24] proposed a multi-thread parallel cooperative multi-search method founded on a solution warehouse strategy to deal with the deterministic VRPTW. Their method is based mainly on two cooperating classes of heuristics, namely tabu search and the evolutionary algorithms. In this regard, Røpke (2009) [25] applied a parallel ALNS to the traveling salesman problem with pickup and delivery and the capacitated vehicle routing problem such that each worker thread obtains a copy of the current solution and performs destroy and repair operations on its local copy in order to produce the best global solution. In the same spirit, Hemmelmayr (2014) [26]) proposed a parallel variant of the Large Neighborhood Search (LNS) to solve the periodic location routing problem. On the same topic, Pillac [27] presented a parallel version of the ALNS by adding a set covering post-optimization model that combines the tours generated throughout the search to assemble a better solution. It is worth mentioning here that the longer a heuristic is run, the better the quality of the solution is. Our contribution in this work is to study the balance between the quality of the solution and the corresponding execution time. Therefore, we suggest through our investigation a compromise between the required running time and the objective function. This can be viewed as a multi-criteria optimization problem.

Thus, the goal of this work is to study the effect of multithreading parallelization of the resolution approach blocks on the running time and the objective function. The optimization problem considered here is a variant of the Robust Vehicle Routing Problem with Time Windows (RVRPTW) including both uncertainties in travel and service times. Our contribution to all previous works lies first on the choice of the efficient Parallel Adaptive Large Neighborhood Search (PALNS) metaheuristic of Ropke [25], which leads to a reduced running time. Moreover, we used our parallel version of the Metropolis Monte Carlo algorithm to generate all possible realizations and to transform the problem under uncertainties to a set of deterministic sub-problems. For more detail about this splitting process, see for instance the previous work [9] and the references therein. Based on the efficient implementation of [25], different combinations (sequential/parallel) of the Monte Carlo algorithm and ALNS are performed. In this way, our strategy offers to decision makers the choice of the combination depending on their preferences and the situation at hand.

To the best of our knowledge, this contribution is the first work to be devoted to the study of VRPTW considering the uncertainties on travel times and service times for different sizes of instances in terms of both the execution time and the objective value.

The outline of this paper is organized as follows. The problem statement and mathematical model is provided in the second section. The sequential robust approach used to solve the problem is the subject of the third section. Within Section 4, we present the parallel Monte Carlo algorithm to generate all possible scenarios and the parallel ALNS algorithm to solve each sub-problem corresponding to each scenario. Finally, a detailed computational and comparative study is given before the concluding remarks and perspectives.

## 2. Problem Statement

The mathematical formulation of the problem can be represented on a graph $G = (N, A)$, where $N = \{o, 1, ..., n\}$ is the set of nodes and $A = \{(i, j) : i, j \in A, i \neq j\}$ is the set of arcs. The node $o$ represents the depot, and each other node is affected by a customer $i$. Each arc $(i, j)$ is assigned to the travel cost $c_{ij}$, which, in general, is proportional to the travel time $t_{ij}$ or the distance $d_{ij}$ between $i$ and $j$. For the rest of this paper, we consider only the travel time cost $t_{ij}$. It is worth mentioning that this travel time $t_{ij}$ is subject to uncertainty $\Delta_{ij}$. The nominal service time is denoted by $P_i^k$ for each vehicle $k$ and node $i$ within the time window $[a_i, b_i]$ and depends on uncertainty $\delta_i$. According to the work of [9], we identify the uncertainty sets related to these times by:

$$U_t = \{\widetilde{t} \in \mathbb{R}^{|A|} \;/\; \widetilde{t_{ij}} = t_{ij} + \Delta_{ij}\epsilon_{ij}, \sum_{(i,j)\in A} \epsilon_{ij} \leq \Gamma, 0 \leq \epsilon_{ij} \leq 1,\ \forall (i,j) \in A\}$$

and

$$U_P = \{\widetilde{P} \in \mathbb{R}^{|N|} \;/\; \widetilde{P_i} = P_i + \delta_i\omega_i, \sum_{i\in N} \omega_i \leq \Lambda, 0 \leq \omega_i \leq 1,\ \forall i \in N\}$$

We denote the subset of arcs that are dependent on uncertainty by $\Psi$ with a cardinal $\Gamma$ and the subset of nodes depending on uncertainty by $\theta$ with a cardinal $\Lambda$.
The binary decision variables $x_{ij}^k$ take the value one if vehicle $k$ travels between the pairs of nodes $(i, j)$ and zero otherwise.

We introduce the model of our problem, which tries to find a solution optimizing the total travel time taking into account the minimization of the worst evaluation over all scenarios:

$$(\text{Min} \sum_{k\in V} \sum_{(i,j)\in A} x_{ij}^k t_{ij} + \underset{\{\Psi/\Psi\subset A, |\Psi|=\Gamma\}}{max} \sum_{k\in V} \sum_{(i,j)\in \Psi} x_{ij}^k \Delta_{ij})$$

subject to:

$$\sum_{k\in V}\sum_{j\in N} x_{ij}^k = 1 \qquad \forall (i \in N) \tag{1}$$

$$\sum_{j\in N} x_{0j}^k = 1 \qquad \forall (k \in V) \tag{2}$$

$$\sum_{i\in N} x_{ih}^k = \sum_{j\in N} x_{hj}^k \qquad \forall (h \in N)\ \forall (k \in V) \tag{3}$$

$$\sum_{i\in N} x_{i0}^k = 1 \qquad \forall (k \in V) \tag{4}$$

$$a_i \leq P_i^k \leq b_i \quad \forall (i \in N) \qquad \forall (k \in V) \tag{5}$$

$$P_i^k + t_{ij} + \delta_i \nu_i^\theta + \Delta_{ij}\mu_{ij}^\Psi - P_j^k \leq (1 - x_{ij}^k)M, \tag{6}$$

$$\forall (i \in N),\ \forall (j \in N \setminus \{0\}),\ \forall (k \in V)\ \forall (\theta \subset N) \mid \theta \mid = \Lambda,\ \forall (\Psi \subset A) \mid \Psi \mid = \Gamma$$

where M is a great value and $\nu_i^\theta$ and $\mu_{ij}^\Psi$ are two indicator functions. When $i \in \theta$, $\nu_i^\theta$ takes the value of one. When $(i, j) \in \Psi$, $\mu_{ij}^\Psi$ takes one.

The constraint (1) stipulates that each customer must be visited once. The constraint (2) guarantees that each tour starts from the depot. The constraint (3) ensures that the same vehicle arrives and leaves from each node it serves. The constraint (4) ensures that each tour ends at the depot. The constraint (5) guarantees that the service time $P_i^k$ at any customer $i$ by vehicle $k$ starts inside a specified time interval $[a_i, b_i]$. The last constraint (6) prohibits the violation of the time windows. Then, if the vehicle arrives ahead of time at a customer $i$, it must wait until the time window $[a_i, b_i]$ opens, and besides, it is not allowed to arrive late.

## 3. Robust Optimization

In real-world applications of operations research, we cannot ignore the fact that in the presence of uncertainties, an optimal solution could become worse or even unreachable from a practical point of view. Therefore, the need to develop models that immunize against those uncertainties has become indispensable.

In general, the uncertain parameters are represented by closed, convex, and bounded uncertainty sets, which can be also estimated from the historic data. Thereby, constructing the adequate uncertainty set has a crucial role in identifying the conservativeness of the model.

In this section, we present briefly the most important sets of uncertainties and the corresponding robust optimization models.

In this regard, we consider the following uncertain linear programming problem:

$$\begin{aligned} &min\ c^\top x \\ &s.t\ Ax \leq B \end{aligned}$$

For the remainder of this section, only the coefficients $\widetilde{a}_{ij}$ of the matrix $A$ are the object of uncertainties, and their values belong to a bounded set of uncertainties called $U$. Accordingly, $\widetilde{a}_{ij}$ takes a value in the interval $[a_{ij} - \overline{a}_{ij}, a_{ij} + \overline{a}_{ij}]$ where $a_{ij}$ is the nominal value and $\overline{a}_{ij}$ represents the maximum positive deviation. Therefore, we can define $\widetilde{a}_{ij}$ as:

$$\widetilde{a}_{ij} = a_{ij} + \zeta_{ij}\overline{a}_{ij}$$

Generally, $\zeta_{ij}$ is a random variable that is subject to uncertainty and varies between $-1$ and one.

For instance, three types of uncertainty sets can be distinguished [28].

### *3.1. Box Uncertainty Set*

The box uncertainty set is an uncertainty structure that takes its name from the box formed by the interaction of perturbations. It aims at finding a conservative solution for a robust problem where the value of all uncertain coefficient perturbations is less than a perturbation bound $\Psi_i$ (see, e.g., [29]). Its uncertainty set can be described as follows:

$$U^A = \{\widetilde{a}_{ij} = a_{ij} + \zeta_{ij}\overline{a}_{ij} \mid \ \mid \zeta_{ij} \mid \leq \Psi_i, \forall i\}$$

The robust counterpart of the problem is given by the following:

$$\begin{aligned} &min\ c^\top x \\ s.t\quad &\sum_j a_{ij}x_j + \Psi_i \sum_j \overline{a}_{ij} y_j \leq b_i \quad \forall\, i \\ &-y_j \leq x_j \leq y_j \quad \forall\, j \\ &y \geq 0 \end{aligned}$$

It is worth pointing out that the problem becomes more conservative as the value of $\Psi_i$ increases.

### *3.2. Ellipsoidal Uncertainty Set*

The ellipsoidal uncertainty set comes to avoid over conservativeness and to limit the uncertainty space by eliminating a subset of uncertainty. The level of robustness can be controlled by modifying the value of the parameter $\Omega$, which defines the borders of the set [30]. This uncertainty set can be given as:

$$U^A = \{\tilde{a}_{ij} = a_{ij} + \zeta_{ij}\bar{a}_{ij} \mid \sum_j \zeta_{ij}^2 \leq \Omega_i^2, \forall i\}$$

The robust counterpart model is expressed in the following way:

$$\begin{aligned} & min\ c^\top x \\ s.t \quad & \sum_j a_{ij}x_j + \sum_j \bar{a}_{ij}y_j + \Omega_i \sqrt{\sum_j \bar{a}_{ij}^2 z_{ij}^2} \leq b_i \quad \forall\, i \\ & -y_{ij} \leq x_j - z_{ij} \leq y_{ij} \quad \forall\, j \\ & y \geq 0 \end{aligned}$$

The inconvenience of this robust counterpart model lies in the generation of a convex nonlinear programming problem, with a greater computational requirement in contrast to linear models.

### *3.3. Polyhedral Uncertainty Set*

The polyhedral uncertainty set corresponds to the most frequent case of uncertainty sets defined as the set of solutions, which are protected against all situations in which at most $\Gamma_i$ coefficients of the $i$th constraints are perturbed. In this case, the robust counterpart is equivalent to a linear optimization problem.

$$U^A = \{\tilde{a}_{ij} = a_{ij} + \zeta_{ij}\bar{a}_{ij} \mid \sum_j \mid \zeta_{ij} \mid \leq \Gamma_i, \forall i\}$$

The robust counterpart of the problem can be defined as below:

$$\begin{aligned} & min\ c^\top x \\ s.t \quad & \sum_j a_{ij}x_j + \max_{\{S_i \cup \{t_i\} \mid S_i \subseteq J_i, |S_i| = |\Gamma_i|, t_i \in J_i \setminus S_i\}} \{\sum_{j \in S_i} \bar{a}_{ij}y_j + (\Gamma_i - \mid \Gamma_i \mid)\bar{a}_{it}y_t\} \leq b_i \quad \forall\, i \\ & -y_j \leq x_j \leq y_j \quad \forall\, j \\ & y \geq 0 \end{aligned}$$

where $J_i$ represents the set of coefficients $a_{ij}$ of the $i$th constraint, which are uncertain. We define for each $i$ a parameter $\Gamma_i$ that varies in the interval $[0, \mid J_i \mid]$. The solution of this model is immunized against all cases where coefficients up to $\mid \Gamma_i \mid$ will change, and one coefficient $a_{it}$ changes by $(\Gamma_i - \mid \Gamma_i \mid)a_{it}$ as reported by [31].

## **4. The Robust Approach for the VRPTW with Uncertain Travel and Service Times**

In this section, we present the robust resolution approach proposed by [9] to deal with the VRPTW under uncertain travel and service times. We assume that the uncertainty in travel times and service times is directed by two parameters $\Gamma$ and $\Lambda$, which belong respectively to the intervals $[0, \mid N \mid + \mid V \mid]$ and $[0, \mid N \mid]$. Those parameters are called budgets of uncertainty and are defined to control the number of travel times and service times, which are allowed to vary from their nominal values. Thus, the major challenge of this approach is to derive, for each scenario ($\Lambda$, $\Gamma$) considered, a robust solution that protects against time window violation or reduces the waiting times.

In order to explain the robust approach, we define some notations to be used in this approach:

$R_N^{\Lambda,\Gamma}$: A possible realization

$S_{best}$: The best robust solution

$S_N$: The solution found at the $N$th realization

$TotalCost(.)$: The total traveled time of a solution

$WorstEval^{\Gamma}(.)$: The worst evaluation of a solution

$Tr_k = (c_1 = 0, c_2, ..., c_n = 0)$: The tour of the vehicle $k$

$\sigma_l = (c_1, c_2, ..., c_l)$: A path of the tour $Tr_k$

$ArcSet^{\Gamma,l}$: A set of arcs within the $\Gamma$ larger deviations of travel time

$NodeSet^{\Lambda,l}$: A set of nodes within the $\Lambda$ larger deviations of service time

$\xi(\sigma_l) = \{c_1, c_2, ..., c_h\}$: All of the nodes that constitute $\sigma_l$

$Arc(\sigma_l) = \{\gamma_1 = (c_1, c_2), \gamma_2 = (c_2, c_3), ..., \gamma_{l-1} = (c_{l-1}, c_l)\}$: The set of the arcs that constitute the path $\sigma_l$

$(\overset{\_k}{s_l})$: The maximum date of arrival of the vehicle $k$ at customer $c_l$

To understand the idea behind this approach, we propose a simplified presentation in four steps summarized in Algorithm 1:

**Algorithm 1.** The robust approach algorithm.

```
Parameters: Set $Solutions$, set $realizations$
Outputs: Solution $solution$
$realizations \longleftarrow MonteCarlo()$
for each $realization \in realizations$ do
  $solution \longleftarrow ALNS(realization)$
  solutions.add(solution)
end for
for each $solution \in solutions$ do
  if $checkRobustness(solution) \neq True$ then
    solutions.remove(solution)
    return NULL
  end if
  if $WorstEval^{\Gamma}(solution) \neq True$ then
    solutions.remove(solution)
    return NULL
  end if
end for
$solution \longleftarrow MinObjective(solutions)$
return $solution$
```

In the first step, the robust algorithm generates a set of realizations using the Metropolis Monte Carlo sampling. Each realization $R_N^{\Lambda,\Gamma}$ corresponds to a possible scenario in which $\Gamma$ travel times of a subset of arcs ($\Psi \subset A$) achieves their maximum values $t_{ij} + \Delta_{ij}$, and $\Lambda$ service times related to a subset of vehicles $(\theta \subset V)$ take their maximum values $P_i + \delta_i$; whereas the other arcs and nodes take respectively their $t_{ij}$ and $P_i$ nominal values.

The objective of the second step is to obtain for each realization $R_N^{\Lambda,\Gamma}$ a feasible solution $Sol_N$ that satisfies the related sub-problem. For this purpose, we apply the Adaptive Large Neighborhood Search (ALNS) metaheuristic [32], which presents an adaptive specialization of the notion of local search, so-called large neighborhoods. It aims to enhance an incumbent solution by diversifying the search process on large neighborhoods. This can be done by applying a pair of destroy and repair operators to the solution and then accepting or rejecting the new solution. In this context, we use three different destroy operators, which contribute to ruin a part of the current solution, namely: the proximity operator, the route portion operator, and the longest detour operator. Then, we recreate a complete solution using the greedy insertion heuristic. The destroy and repair neighborhoods are selected by a roulette wheel mechanism that uses the search history of each operator to favor the best performing one.

The third step depicts a mechanism conceived of to verify the feasibility of the solution achieved by using the ALNS approach in the preceding step, by examining the time windows associated with each visited customer. The last step is devoted to the evaluation of the robustness of the solution, according to the worst case criterion. In practice, we evaluate the solution $S_N$ on the worst of possible cases, which relates to the realization where the $\Gamma$ travel times of this solution reach at the same time their maximum values. The pseudocodes of those methods are shown in Algorithms 2 and 3, respectively.

**Algorithm 2.** Check for robustness.

```
$feasible \leftarrow true$
for $k \leftarrow 1$ to $| V |$ do
  for $l \leftarrow 2$ to $| \xi(Tr_k) |$ do
    calculate $ArcSet^{\Gamma,l}$ and $NodeSet^{\Lambda,l-1}$
    for $\lambda \leftarrow 1$ to $l-1$ do
      if $l > \Gamma + 1$ and $\gamma_\lambda \notin ArcSet^{\Gamma,l}$ then
        $t_{\gamma_\lambda} \leftarrow t_{\gamma_\lambda}$
      else
        $t_{\gamma_\lambda} \leftarrow t_{\gamma_\lambda} + \Delta_{\gamma_\lambda}$
      end if
    end for
    for $i \leftarrow 1$ to $l-1$ do
      if $l > \Lambda$ and $c_i \notin NodeSet^{\Lambda,l-1}$ then
        $P_{c_i} \leftarrow P_{c_i} + \delta_{c_i}$
      end if
    end for
    $(\overset{\_k}{s_l}) \leftarrow 0$
    for $i \leftarrow 2$ to $l$ do
      $(\overset{\_k}{s_l}) \leftarrow max((\overset{\_k}{s_l}) + t_{\gamma_{i-1}} + P_{c_{i-1}}, a_{c_i})$
    end for
    if $(\overset{\_k}{s_l}) > b_{c_l}$ then
      feasible takes false, and the algorithm ends
    end if
  end for
end for
```

**Algorithm 3.** Evaluation on the worst case.

```
$WorstEval^{\Gamma}(S_N) \leftarrow 0$
put in descending order all the arcs of $\gamma(S_N)$ according to their maximum deviations.
for $i \leftarrow 1$ to $\Gamma$ do
    $WorstEval^{\Gamma}(S_N) \leftarrow WorstEval^{\Gamma}(S_N) + t_{\gamma_i} + \Delta_{\gamma_i}$
end for
for $i \leftarrow \Gamma + 1$ to $\gamma(S_N)$ do
    $WorstEval^{\Gamma}(S_N) \leftarrow WorstEval^{\Gamma}(S_N) + t_{\gamma_i}$
end for
return $WorstEval^{\Gamma}(S_N)$
```

For a detailed description of this robust approach, we refer the reader to the study of [9] and the references therein.

## 5. The Parallel Robust Approach for the VRPTW with Uncertain Travel and Service Times

In this section, we provide a detailed exposition of the multi-threading parallel approach used in this paper. We start by giving some insights into the motivation before handling the complete description of the proposed approach.

The first challenge of such an approach is to derive the best robust solution that responds to to all uncertainties with a reduced running time. However, the sequential robust approach suffers from lengthy computational times; partly because the generation of scenarios is time-consuming, as well as the research of the solution block. Unlike those blocks, the check of robustness and the evaluation of the worst case blocks are not time-consuming, generally because they are restricted to evaluating the obtained solution.

Table 1 confirms our assertion that the Monte Carlo and the ALNS blocks take considerable time compared to the other blocks. This can be explained by the fact that the first phase is responsible for generating all the possible scenarios in which $\Gamma$ displacement times take their maximum values and $\Lambda$ service times take their maximum values. The ALNS block is also time-consuming, since it chooses at each iteration a neighborhood to explore, based on a score that reflects its past performance. This is possible by the application of several destroy and repair operators. Oppositely, the other blocks are not time-consuming, considering that the first mechanism verifies the feasibility of our solution by investigating the related time windows of each visited customer, and the second mechanism is dedicated to the evaluation of the robustness of the solution based on the worst case robust criterion. As an alternative to overcome this impediment, we propose a multithreading parallelization of the costly blocks as detailed in the next subsections.

**Table 1.** The execution time of the sequential robust approach blocks in ms.

| Instance Size | Monte Carlo Block/Iteration | ALNS Block/Iteration | Check of Robustness Block/Iteration | Worst Case Evaluation Block/Iteration |
|---|---|---|---|---|
| 1000_100_100 | 210 | 258 | 46 | 30 |
| 1500_100_100 | 325 | 409 | 85 | 52 |
| 2000_100_100 | 478 | 632 | 217 | 96 |
| 2500_100_100 | 621 | 867 | 377 | 163 |
| 3000_100_100 | 1042 | 1436 | 510 | 294 |
| 3500_100_100 | 1893 | 2229 | 682 | 325 |
| 4000_100_100 | 2365 | 3538 | 793 | 549 |
| 4500_100_100 | 2901 | 4428 | 907 | 703 |

### *5.1. The Parallel Monte Carlo Sampling*

The parallel Metropolis Monte Carlo algorithm described below is a scenario generation technique that uses a defined number of worker threads to generate in a parallel way a predefined number $n$ of independent identically distributed scenarios. In practice, each worker thread produces a realization $R_l^{\Lambda,\Gamma}$ that corresponds to a deterministic VRPTW problem, in which $\Gamma$ displacement times and $\Lambda$ service times reach simultaneously their maximum values. The pseudocodes of this parallel method is shown in Algorithm 4:

**Algorithm 4.** Parallel Monte Carlo.

```
Input: $\Lambda, \Gamma, n$
Output: scenarios
for $l \leftarrow 1$ to $n$ in parallel do
  for $k \leftarrow 1$ to $\Gamma$ do
    Select randomly two clients $i$ and $j$
    $\widetilde{t_{ij}} \leftarrow t_{ij} + \Delta_{ij}$
  end for
  for $k \leftarrow 1$ to $\Lambda$ do
    Select randomly a client $i$
    $\widetilde{P_i} \leftarrow P_i + \delta_i$
  end for
  Generate a scenario $R_l^{\Lambda,\Gamma}$
  $t(R_l^{\Lambda,\Gamma}) \leftarrow \widetilde{t}(R_l^{\Lambda,\Gamma})$
  $P(R_l^{\Lambda,\Gamma}) \leftarrow \widetilde{P}(R_l^{\Lambda,\Gamma})$
  Add $R_l^{\Lambda,\Gamma}$ to scenarios
end for
return scenarios
```

### *5.2. The Parallel ALNS*

In this subsection, we present the Parallel Adaptive Large Neighborhood Search (PALNS) method developed by [25]. This method can be presented in three phases (see Figure 1).

At the first level, we generate an initial feasible solution by using the greedy insertion metaheuristic [33]. The main idea behind this method is to select the best feasible insertion place in the incumbent route for each non-inserted node taking into account two major factors: the increase in total cost of the current route after the insertion and the delay of the service start time of the customer succeeding the newly inserted customer.

The second phase is related to a set of destroy and repair operators designed to enhance the incumbent solution. In this context, each worker thread deals with a copy of the current solution and executes destroy and repair methods on this local copy in order to improve it.

The third phase collects the routes of different local solutions that each thread has obtained, for the purpose of combining it into a new better temporary global solution and sending it to be improved. At this stage, we will accept or reject the generated solution, based on a hill climbing acceptance criterion. It is worth mentioning that only the current and global best solutions are shared between worker threads, in order to update them as necessary by repeating the process until a stop criterion is met.

We should point out here that the ALNS uses a flexible layer with a set of destruction heuristics (proximity operator, route portion operator, and longest detour operator) and an insertion heuristic (the greedy insertion) and applies them by a roulette wheel selection that highlights the corresponding performance obtained during the search. On the other hand, the LNS heuristic does not use this scoring mechanism.

For a completed description of the used PALNS, we refer the reader to the study of [25] and the references therein.

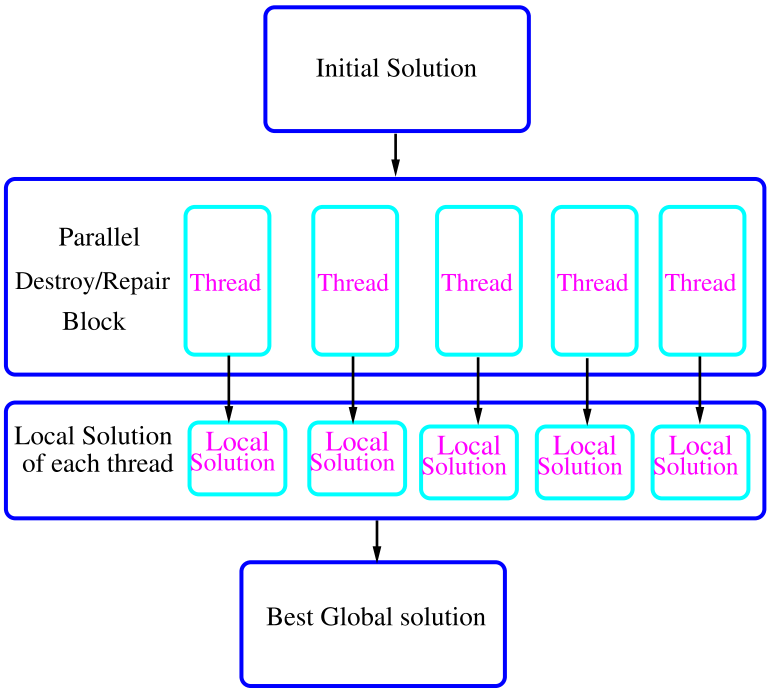

**Figure 1.** The parallel adaptive large neighborhood search.

## 6. Computational Experiments

In this section, we summarize a few of the results obtained by evaluating different robust approaches conceived of to solve the vehicle routing problem with time windows with uncertain service and travel times.

We explore the effect of applying the thread parallelism to the Monte Carlo and ALNS blocks, on the execution time and the objective value. This leads to four different robust combinations: the sequential approach that uses sequential Monte Carlo and sequential ALNS (MC sequential and ALNS sequential), the approach employing sequential Monte Carlo and parallel ALNS (MC sequential and ALNS parallel), the approach that uses parallel Monte Carlo and sequential ALNS (MC parallel and ALNS sequential), and the parallel approach combining parallel Monte Carlo and parallel ALNS (MC parallel and ALNS parallel). We should note here that the sequential approach (MC sequential and ALNS sequential) coincides with the only method from the literature [9] that deals with the considered problem.

It is important to mention that the notion of thread parallelism used in our context can be defined as the capability of a processing unit to execute multiple processes contemporaneously or with time slicing. By means of a thread, the smallest unit of processing can be performed in an operating system in order to accelerate the execution time and manage the code over time. In our study, we use four threads in order to establish the comparison between the different approaches. The choice of four threads is not restrictive, and we can use as many threads as possible. Our assumption is that using more threads leads to the improvement of the average execution time, but it slightly decreases the quality of the obtained solution. For more details, see, e.g., [25].

The robust approaches studied in this paper were tested on a classical set of instances in reference to Solomon's benchmark (1987) [33] and Gehring and Homberger's benchmark [34]:

- Set $R$ contains problems with randomized customers.
- Set $C$ contains problems with clustered customers.
- Set $RC$ contains problems with both clustered and randomized customers.

In order to simulate the uncertainty of RVRPTW by discrete scenarios, the uncertain travel time and uncertain service time are generated at random, so that they go from 0 to 10. We denote the used instances as follows: $Gr\_\Gamma\_\Lambda$, where $\Gamma$ and $\Lambda$ present respectively the number of travel times and service times considered uncertain and $Gr$ refers to the instance of Solomon and Gehring and Homberger's benchmark or the size of larger instances.

For small instances (Solomon and Homberger's instances), we chose 15,600 iterations as a stop criterion in order to diversify the research, which may ameliorate the quality of our solution, because the solution has a greater chance to escape from a local minimum. As far as we are aware, the maximal number of iterations for instances is falling in the literature. With a view toward examining the capability of our approaches for tackling that problem, we judge it based on the average performance over 10 multiple independent runs.

For the set of instances larger than 1000, we generate random representative instances in such a manner that the travel time between each pair of nodes is between 0 and 100, and the same for the service time. The time interval has a capacity of 200 between the start and the end of the service at each customer. We forced a stop condition of about 20 min, which allows a good comparison between the proposed methods in terms of the number of reached iterations for the same time interval. Then, we present the measurements achieved for a single run.

The proposed algorithms were implemented in Java 7, compiled with Intel compiler Celeron 1.80 GHz core i5 with 8 GB RAM.

### *6.1. Execution Time*

Table 2 presents a comparison of the execution time for each instance group between different robust approaches with the maximal number of iterations of 15,600. As expected, the results show that the approaches containing the parallel ALNS succeeded by those containing parallel Monte Carlo lead to the improvement of the average execution time compared to other sequential approaches. This can be explained by the fact that the ALNS block succeeded by the Monte Carlo block consumes most of the execution time compared to other blocks.

In the same spirit, we report in Table 3 a comparison of different approaches according to the number of reached iterations when the stopping limit time is about 20 min for the group of instances 2500–4500. The approach containing more parallel blocks attained more iterations for the same time interval since it reduced the execution time of the consuming blocks.

Table 4 depicts the improvement results in execution time for Solomon's instances. The conclusion from this table is clear: the running time is much faster for the approaches using parallel ALNS succeeded by those employing the parallel Monte Carlo algorithm.

**Table 2.** The execution time of different robust approaches in seconds for 15,600 iterations.

| Instance | MC Sequential and ALNS Sequential | MC Sequential and ALNS Parallel | MC Parallel and ALNS Sequential | MC Parallel and ALNS Parallel |
|---|---|---|---|---|
| 100_10_10 | 832.24 | 701.03 | 725.45 | 574.24 |
| 200_25_25 | 1007.41 | 853.11 | 883.25 | 669.33 |
| 400_25_25 | 1331.47 | 1037.98 | 1101.12 | 770.91 |
| 600_50_50 | 1402.50 | 1160.24 | 1230.64 | 858.38 |
| 800_50_50 | 1545.39 | 1302.85 | 1331.68 | 979.14 |
| 1000_100_100 | 2687.50 | 2062.25 | 2165.74 | 1340.49 |
| 1500_100_100 | 4519.79 | 3267.98 | 3665.37 | 2413.56 |
| 2000_100_100 | 5606.81 | 3953.18 | 4389.74 | 2736.11 |

**Table 3.** Number of iterations reached in 20 min.

| Instance | MC Sequential and ALNS Sequential | MC Sequential and ALNS Parallel | MC Parallel and ALNS Sequential | MC Parallel and ALNS Parallel |
|---|---|---|---|---|
| 2500_100_100 | 3038 | 4275 | 3879 | 5681 |
| 3000_100_100 | 2163 | 3572 | 2794 | 4141 |
| 3500_100_100 | 1375 | 2992 | 1985 | 3709 |
| 4000_100_100 | 912 | 2401 | 1386 | 3124 |
| 4500_100_100 | 594 | 1994 | 1078 | 2300 |

**Table 4.** Solomon's instance: Comparison of the runtimes in seconds of different robust approaches.

| Instance | MC Sequential and ALNS Sequential | MC Sequential and ALNS Parallel | MC Parallel and ALNS Sequential | MC Parallel and ALNS Parallel |
|---|---|---|---|---|
| R101_10_10 | 433 | 267 | 378 | 212 |
| C101_10_10 | 491 | 320 | 434 | 263 |
| RC101_10_10 | 468 | 286 | 407 | 225 |
| R201_10_10 | 487 | 301 | 425 | 239 |
| C201_10_10 | 563 | 367 | 497 | 301 |
| RC201_10_10 | 553 | 338 | 482 | 267 |
| R121_25_25 | 580 | 277 | 512 | 309 |
| C121_25_25 | 653 | 442 | 583 | 372 |
| RC121_25_25 | 605 | 406 | 539 | 340 |
| R221_25_25 | 673 | 438 | 595 | 360 |
| C221_25_25 | 784 | 531 | 700 | 447 |
| RC221_25_25 | 707 | 475 | 629 | 397 |
| R141_25_25 | 668 | 423 | 586 | 341 |
| C141_25_25 | 775 | 538 | 696 | 459 |
| RC141_ 25_25 | 747 | 504 | 666 | 423 |
| R241_25_25 | 937 | 629 | 823 | 479 |
| C241_25_25 | 981 | 683 | 877 | 589 |
| RC241_25_25 | 957 | 646 | 854 | 543 |
| R161_50_50 | 741 | 541 | 675 | 475 |
| C161_50_50 | 907 | 682 | 832 | 607 |
| RC161_50_50 | 825 | 607 | 753 | 535 |
| R261_50_50 | 923 | 730 | 909 | 640 |
| C261_50_50 | 1089 | 814 | 999 | 726 |
| RC261_50_50 | 1051 | 774 | 958 | 681 |
| R181_50_50 | 923 | 698 | 848 | 623 |
| C181_50_50 | 942 | 732 | 874 | 664 |
| RC181_ 50_50 | 929 | 706 | 855 | 632 |
| R281_50_50 | 1136 | 860 | 1044 | 768 |
| C281_50_50 | 1249 | 971 | 1157 | 879 |
| RC281_50_50 | 1223 | 925 | 1123 | 829 |
| R1101_ 100_100 | 1179 | 764 | 1040 | 625 |
| C1101_100_100 | 1213 | 817 | 1081 | 685 |
| RC1101_100_100 | 1204 | 800 | 1069 | 665 |
| R2101_100_100 | 1757 | 1140 | 1552 | 935 |
| C2101_100_100 | 1815 | 1224 | 1618 | 1027 |
| RC2101_100_100 | 1775 | 1181 | 1577 | 983 |

### *6.2. Objective Function*

Table 5 presents the objective function of different robust approaches for the group of instances 2500–4500. When the size of the instance increases, the cost function of the approaches containing parallel and sequential ALNS solution converges. Then, we compute the mean absolute percent deviation (MAPD), which is the absolute difference between the cost function of the approach containing the sequential ALNS and the parallel ALNS divided by the magnitude of the objective function in the approach with sequential ALNS. This indicator (MAPD) goes from 13.05% for the instance of size 1000 to 3.96% for the instance of size 4500. We can conclude that incorporating the parallel ALNS in the approaches is efficient for large instances.

Table 6 depicts the results of the objective value for some of Solomon's instances. We observe that the ALNS controls the solution quality. Then, the approaches that contain a sequential ALNS yield better results than those that contain parallel ALNS. The ALNS is responsible for finding the solution of each scenario, in contrast with the Monte Carlo algorithm, which is limited to generating the possible scenarios. When we increase the instance size, the quality of the solution of the parallel approach becomes more interesting.

**Table 5.** Comparison of different approaches according to the objective function.

| Instance | MC Sequential and ALNS Sequential | MC Sequential and ALNS Parallel | MC Parallel and ALNS Sequential | MC Parallel and ALNS Parallel |
|---|---|---|---|---|
| 1000_100_100 | 18,612 | 21,031 | 18,642 | 21,075 |
| 1500_100_100 | 25,961 | 28,557 | 25,904 | 28,516 |
| 2000_100_100 | 36,420 | 38,969 | 36,408 | 38,943 |
| 2500_100_100 | 44,981 | 48,129 | 44,998 | 48,161 |
| 3000_100_100 | 52,817 | 55,986 | 52,836 | 56,014 |
| 3500_100_100 | 60,356 | 63,977 | 60,321 | 63,937 |
| 4000_100_100 | 67,675 | 71,059 | 67,642 | 70,986 |
| 4500_100_100 | 75,306 | 78,318 | 75,329 | 78,371 |

**Table 6.** Solomon's instance: comparison of the objective value of different robust approaches.

| Instance | MC Sequential and ALNS Sequential | MC Sequential and ALNS Parallel | MC Parallel and ALNS Sequential | MC Parallel and ALNS Parallel |
|---|---|---|---|---|
| R101_10_10 | 1918.56 | 2225.52 | 1926.14 | 2249.69 |
| C101_10_10 | 870.46 | 1009.73 | 883.01 | 1025.84 |
| RC101_10_10 | 1882.76 | 2145.48 | 1897.04 | 2183.92 |
| R201_10_10 | 1434.99 | 1663.44 | 1415.09 | 1663.24 |
| C201_10_10 | 666.54 | 779.27 | 663.52 | 761.51 |
| RC201_10_10 | 1663.38 | 1895.82 | 1679.45 | 1900.11 |
| R121_25_25 | 5696.41 | 6379.52 | 5709.64 | 6659.85 |
| C121_25_25 | 3115.86 | 3519.95 | 3115.86 | 3504.38 |
| RC121_25_25 | 3915.01 | 4384.80 | 3919.63 | 3492.62 |
| R221_25_25 | 5385.14 | 6031.35 | 5385.14 | 6024.07 |
| C221_25_25 | 2490.94 | 2813.70 | 2496.91 | 2813.70 |
| RC221_25_25 | 3657.12 | 4095.84 | 3657.12 | 4106.49 |
| R141_25_25 | 10,939.12 | 11,485.95 | 10,951.24 | 11,485.95 |
| C141_25_25 | 8138.18 | 8463.52 | 8127.63 | 8480.31 |
| RC141_25_25 | 10,673.61 | 11,099.92 | 10,697.27 | 11,080.79 |
| R241_25_25 | 10,442.34 | 10,859.68 | 10,463.29 | 10,843.81 |
| C241_25_25 | 4932.06 | 5129.28 | 4863.65 | 5129.28 |
| RC241_25_25 | 20,917.12 | 21,690.92 | 7181.07 | 21,709.56 |
| R161_50_50 | 24,277.04 | 24,714.02 | 24,277.04 | 24,703.48 |
| C161_50_50 | 15,511.06 | 15,945.30 | 15,523.45 | 15,940.06 |
| RC161_50_50 | 21,917.15 | 22,245.75 | 21,937.13 | 22,291.10 |
| R261_50_50 | 22,070.80 | 22,445.19 | 22,121.72 | 22,457.14 |
| C261_50_50 | 10,617.60 | 10,871.80 | 10,632.96 | 10,886.71 |
| RC261_50_50 | 16,167.74 | 16,458.00 | 16,186.14 | 16,430.92 |

**Table 6.** *Cont.*

| Instance | MC Sequential and ALNS Sequential | MC Sequential and ALNS Parallel | MC Parallel and ALNS Sequential | MC Parallel and ALNS Parallel |
|---|---|---|---|---|
| R181_50_50 | 39,011.14 | 39,479.13 | 38,979.56 | 39,431.41 |
| C181_50_50 | 28,491.39 | 29,117.85 | 28,491.39 | 29,117.83 |
| RC181_50_50 | 35,821.46 | 36,358.31 | 35,853.12 | 36,347.02 |
| R281_50_50 | 32,289.08 | 32,870.20 | 32,271.22 | 32,870.20 |
| C281_50_50 | 14,213.43 | 14,483.04 | 14,219.34 | 14,497.15 |
| RC281_50_50 | 28,307.07 | 28,788.21 | 28,307.07 | 28,805.09 |
| R1101_100_100 | 60,506.88 | 62,805.22 | 60,493.16 | 62,805.22 |
| C1101_100_100 | 52,251.98 | 53,766.83 | 52,257.48 | 53,742.30 |
| RC1101_100_100 | 53,322.91 | 55,188.27 | 43,318.10 | 55,188.27 |
| R2101_100_100 | 48,103.23 | 49,449.88 | 48,103.23 | 48,463.23 |
| C2101_100_100 | 20,735.79 | 21,357.05 | 20,700.14 | 21,357.05 |
| RC2101_100_100 | 43,853.65 | 44,905.47 | 43,853.65 | 44,884.35 |

## 7. Conclusions

In this work, we study the robust vehicle routing problem with time windows where travel times and service times are both the subject of uncertainty. For this purpose, we opt for the robust technique proposed by [9] to deal with the problem. As far as the adopted approach derives the best robust solution that responds to all uncertainties, it still suffer from lengthy computational times; partly because the generation of scenarios, as well as the research of the solution block are time-consuming. As an alternative to remedy this problem, we introduce a procedure for the thread parallelism in the Monte Carlo block, and we use the parallel ALNS proposed by [25]. This leads to four different robust approaches combining the (sequential/parallel) Monte Carlo algorithm and the (sequential/parallel) ALNS.

The considered approaches are tested on Solomon's benchmark instance of VRPTW and lager instances generated randomly. Accordingly, we can offer a decision-making solution that provides great protection against delays in a reasonable running time. However, we should note that the related counterpart of using the parallel ALNS, which is the objective value, can be influenced, since the parallel ALNS slightly reduces the quality of the solution, especially for small instances.

In future works, we intend to include a pre-processing step based on different clustering techniques such as K-means, K-medoids, density-based spatial, etc., in order to ensure the commitment of the solutions. These techniques will not change the structure of the suggested approach drastically, and our assumption is that they will enhance the solution quality obtained with the parallel ALNS.

**Author Contributions:** All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

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

**Acknowledgments:** We thank the anonymous reviewers for their careful reading of our paper and their many insightful comments and suggestions.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Eksioglu, B.; Vural, A.V.; Reisman, A. The vehicle routing problem: A taxonomic review. *Comput. Ind. Eng.* **2009**, *57*, 1472–1483. [CrossRef]
2. Braekers, K.; Ramaekers, K.; Nieuwenhuyse, I. The Vehicle Routing Problem: State of the Art Classification and Review. *Comput. Ind. Eng.* **2015**, *99*, 300–313. [CrossRef]

3. Vidal, T.; Laporte, G.; Matl, P. A concise guide to existing and emerging vehicle routing problem variants. *Eur. J. Oper. Res.* **2020**, *286*, 401–416. [CrossRef]
4. Bräysy, O.; Gendreau, M. Vehicle Routing Problem with time windows, Part II: Metaheuristics. *J. Transp. Sci.* **2005**, *39*, 119–139. [CrossRef]
5. Kallehauge, B.; Larsen, J.; Madsen, O.B.; Solomon, M.M. Vehicle Routing Problem with Time Windows. *Column Gener.* **2005**, *1*, 67–98.
6. Dror, M.; Trudeau, P. Stochastic vehicle routing with modified savings algorithm. *Eur. J. Oper. Res.* **1986**, *23*, 228–235. [CrossRef]
7. Dror, M.; Laporte, G.; Louveaux, F.V. Vehicle Routing with Stochastic Demands and Restricted Failures. *ZOR-Model. Oper. Res.* **1993**, *37*, 183–273. [CrossRef]
8. Gendreau, M.; Laporte, G.; Séguin, R. Stochastic vehicle routing. *Eur. J. Oper. Res.* **1996**, *88*, 3–12. [CrossRef]
9. Nasri, M.; Metrane, A.; Hafidi, I.; Jamali, A. A robust approach for solving a vehicle routing problem with time windows with uncertain service and travel times. *Int. J. Ind. Eng. Comput.* **2020**, *11*, 1–16. [CrossRef]
10. Rouky, N.; Boukachour, J.; Boudebous, D.; Alaoui, A. A Robust Metaheuristic for the Rail Shuttle Routing Problem with Uncertainty: A Real Case Study in the Le Havre Port. *Asian J. Shipp. Logist.* **2018**, *34*, 171–187. [CrossRef]
11. Wu, L.; Hifi, M.; Bederina, H. A new robust criterion for the vehicle routing problem with uncertain travel time. *Comput. Ind. Eng.* **2017**, *112*, 607–615. [CrossRef]
12. Gutin, G.; Punnen, A. *The Traveling Salesman Problem and Its Variations*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2002.
13. Toth, P.; Vigo, D. An overview of vehicle routing problems. In *9 of SIAM Monographs on Discrete Mathematics and Applications*; SIAM: Philadelphia, PA, USA, 2002; pp. 1–26.
14. Chiang, W.C.; Russell, R.A. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Ann. Oper. Res.* **1996**, *63*, 3–27. [CrossRef]
15. Afifi, S.; Dang, D.C.; Moukrim, A. A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints. In *International Conference on Learning and Intelligent Optimization*; Lecture Notes in Computer; Science Springer: Berlin/Heidelberg, Germany, 2013; Volume 7997.
16. Mladenović, N.; Hansen, P. Variable neighborhood search. *Comput. Oper. Res.* **1997**, *24*, 1097–1100. [CrossRef]
17. Dhahri, A.; Mjirda, A.; Zidi, K.; Ghedira, K. A VNS-based Heuristic for Solving the Vehicle Routing Problem with Time Windows and Vehicle Preventive Maintenance Constraints. *Procedia Comput. Sci.* **2016**, *80*, 1212–1222. [CrossRef]
18. Gambardella, L.M.; Taillard, E.; Agazzi, G. MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In *New Ideas in Optimization*; Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale: London, UK, 1999; pp. 63–76.
19. Tan, X.; Zhuo, X.; Zhang, J. Ant Colony System for Optimizing Vehicle Routing Problem with Time Windows (VRPTW). In *Computational Intelligence and Bioinformatics. ICIC 2006*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4115, pp. 33–38.
20. Thangiah, S. Vehicle routing with time windows using genetic algorithms. In *Application Handbook of Genetic Algorithms: New Frontiers*; Artificial Intelligence Lab., Slippery Rock Univ.: Slippery Rock, PA, USA, 1995; pp. 253–277.
21. Potvin, J.Y.; Kervahut, T.; Garcia, B.L.; Rousseau, J.M. The vehicle routing problem with time windows part I: Tabu search. *INFORMS J. Comput.* **1996**, *8*, 158–164. [CrossRef]
22. Taillard, É.; Badeau, P.; Gendreau, M.; Guertin, F.; Potvin, J.Y. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transp. Sci.* **1997**, *31*, 170–186. [CrossRef]
23. Bräysy, O.; Gendreau, M. Tabu Search heuristics for the Vehicle Routing Problem with Time Windows. *Top J.* **2002**, *10*, 211–237. [CrossRef]
24. Bouthillier, A.; Crainic, T.G. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Comput. Oper. Res.* **2005**, *32*, 1685–1708. [CrossRef]
25. Røpke, S. PALNS—A software framework for parallel large neighborhood search. In Proceedings of the 8th Metaheuristic International Conference CDROM, Metaheuristic International Conference, Hamburg, Germany, 13–16 July 2009.
26. Hemmelmayer, V.C. Sequential and Parallel Large Neighborhood Search Algorithms for the Periodic Location Routing Problem. *Eur. J. Oper. Res.* **2002**, *243*, 52–60. [CrossRef]
27. Pillac, V.; Gendreau, M.; Guéret, C.; Medaglia, A.L. A parallel matheuristic for the technician routing and scheduling problem. *Optim. Lett.* **2013**, *7*, 1525–1535. [CrossRef]
28. Ordóñez, F. *Robust Vehicle Routing. Tutorials in Operations Research*; Institue for Operations Research and the Management Sciences (INFORMS): Catonsville, MD, USA, 2010; pp. 153–178.
29. Ben-Tal, A.; Nemirovski, A. Robust convex optimization. *Oper. Res. Lett.* **1998**, *23*, 769–805. [CrossRef]
30. Ben-Tal, A.; Nemirovski, A. Robust solutions of uncertain linear programs. *Oper. Res. Lett.* **1999**, *25*, 1–13. [CrossRef]
31. Bertsimas, D.; Sim, M. The price of robustness. *Oper. Res.* **2004**, *52*, 35–53. [CrossRef]
32. Røpke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472. [CrossRef]

33. Solomon, M. Algorithms for the Vehicle Routing and Scheduling Problem with time Window Constraints. *Oper. Res. J.* **1987**, *35*, 254–265. [CrossRef]
34. Gehring, H.; Homberger, J. A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. In *Proceedings of EUROGEN99*; Springer: Berlin, Germany, 1999; pp. 57–64. Available online: http://sintef.no/projectweb/top/vrptw/Homberger-benchmark/ (accessed on 5 October 2020).