A modified ALNS algorithm for vehicle routing problems with time windows

NASRI Mehdi, METRANE Abdelmoutalib, HAFIDI Imad
National School of Applied Sciences Khouribga, Sultan Moulay Slimane University, Beni-Mellal, Morocco
E-mail: nasri.mathh@gmail.com

Abstract. In this paper, we present an efficient heuristic for the vehicle routing problem with time windows (VRPTW), inspired by the Adaptive Large Neighborhood Search (ALNS) previously suggested by Røpke and Pisinger [21]. The proposed heuristic uses the Modified Choice function (MCF) of Drake [12] as an elegant selection mechanism to favor the most successful operators instead of the roulette wheel selection. This general method is denoted Modified Adaptive Large Neighborhood Search (MALNS). The computational experiments are performed and the comparison with the classical ALNS is given according to Solomon’s benchmark, and its extension the instances of Gehring and Homberger’s benchmark.

1. Introduction

During the last decade, Vehicle Routing Problem (VRP) and its variants have been studied very much, because of its large applications in numerous fields in the real world. The VRP regards to find a set of minimum-cost vehicle routes, starting and finishing to a depot, in order to serve some customers. Since its first formulation in 1959 by Dantzig and Ramser [9], the problem has been studied widely (see for e.g. [13], [3] and [29]). Furthermore, the researchers are interested in the most complex variants of VRP to bring it closer to the real world. The problem becomes more complex when including some additional restrictions, as for example considering the time windows constraints, which makes the problem a Vehicle Routing Problem with Time Windows (VRPTW).

Over the last period, the VRPTW has been the subject of several pieces of research ([5], [17] and [2]), the goal is to serve all the customers in a specific interval of time. Typically, there are two kinds of the problem such as soft time windows and hard time windows. In the hard version of the problem, the time windows constraints must be respected, if the vehicle arrives at a customer earlier than expected, it must wait until the customer opens and it is also not allowed to arrive delayed. Nevertheless, the time windows restrictions may be violated taking into consideration additional penalty costs to the solution cost in the soft time windows.

Different solution approaches have been proposed to deal with the VRPTW, being able to be classified into three different classes. First, the exact methods are only applied to solve small-scalar instances even with large amount of computing time (see for e.g. [16] and [28]). Second, the metaheuristics are solution methods which are able to achieve high-quality solutions for large instances within acceptable run-time. Recent trends show that a large number of metaheuristics have been designed to solve the VRPTW such as Tabu Search (TS) ([20], [25], [4]), the Simulated Annealing (SA) ([7], [1]), Ant Colony Optimisation Algorithm (ACO) ([14], [26]) and Variable Neighborhood Search (VNS) ([18], [10]). However, they require knowledge and experience to be applied effectively. The third class of solutions is called hyper-heuristics, conceived for automating the task of heuristic methods by working on low-level heuristics search space in order to solve hard computational problems. The main challenge of hyper-heuristics is to select the adequate
sequence of heuristics in a given situation rather than solving the problem directly. As survey of hyper-heuristics can be found in [6].

Different algorithms fitted with selection mechanisms have been conceived recently. Among them, the Adaptive Large Neighborhood Search (ALNS) developed by Ropke and Pisinger [21]. It is considered as one of the most efficient adaptive approaches which uses a roulette wheel mechanism to select the most successful operators. In the same spirit, the Choice Function (CF) is a clever selection technique introduced by [8], which scores heuristics according to a ponderation of three different measurements. A specialization of the concept of choice function called Modified Choice Function (MCF) has been proposed by Drake [12]. The MCF has outperformed the original Choice Function in several combinatorial problems.

In this paper, we develop a solution approach which fits into the class of hyper-heuristic methods, namely a modified adaptive large neighborhood search (MALNS) with a view to deal with the VRPTW. The presented method consists on incorporating the modified choice function into the selection phase of the ALNS in order to efficiently manage the destroy and repair operators. To assess the performance of our MALNS, we perform computational experiments on a group of small instances in reference to Solomon’s benchmark [24], and its extension the instances of Gehring and Homberger’s benchmark [15], and we show that integrating the modified choice function can have more important impact on the performance of the ALNS.

The rest of the article is organized according to the following. Section 2 presents the mathematical formulation of the VRPTW. The proposed approach is explained in section 3 with a complete description of the used methods. Section 4 shows a computational and comparative study between our MALNS algorithm and the classical ALNS, found by solving a large number of well-known Solomon’s homogeneous VRPTW benchmark problems. The last section addresses some concluding remarks.

2. Problem statement

In this section, we introduce the mathematical formulation of the vehicle routing problem with time windows. We start with the VRP problem which is a minimization of the total traveled distance including four constraints. Then, we exhibit time windows and hence two other constraints. From one hand, the service at any client begins inside a given time interval. From the second hand, if the vehicle arrives earlier than desired at a client, it must wait until the point that the time window opens and also it is not allowed to arrive delayed. Taking into account these two constraints on time windows, the VRP problem becomes a VRPTW problem.

Presently, in the interest of describing our problem, the set of nodes will be denoted by $N$, using $i$ and $j$ to denote general nodes, we denote the depot by $o$. We affect to each customer a time window $[a_i, b_i]$, and to each edge $(i, j)$ which belongs to the set of arcs $A$ a cost $d_{ij}$. $x_{ij}^k$ are conditional decision variables, valued at 1 when the vehicle $k$ which belongs to the set $V$ utilizes the edge $(i, j)$. Here is the mathematical model of the VRPTW:[17]

$$\text{Min } (c(x) = \sum_{k \in V} \sum_{(i,j) \in A} d_{ij} x^k_{ij})$$

Under the constraints:

$$\sum_{k \in V} \sum_{j \in N} x^k_{ij} = 1 \quad \forall (i \in N)$$

(1)

$$\sum_{j \in N} x^k_{0j} = 1 \quad \forall (k \in V)$$

(2)

$$\sum_{i \in N} x^k_{ih} - \sum_{j \in N} x^k_{hj} = 0 \quad \forall (h \in N) \quad \forall (k \in V)$$

(3)

$$\sum_{i \in N} x^k_{0i} = 1 \quad \forall (k \in V)$$

(4)

The first constraint ensures that each client has to be visited once. The constraint (2) guarantess that each tour must begin from the depot. The thirth constraint consists in the flow conservation,
and consequently ensures that a vehicle arrives must leave from each node. Finally, the last constraint ensures that each tour returns to the depot.

We require an additional constraint ensuring that the service time $P^k_i$ at any client $i$ by vehicle $k$ must be between an earliest and latest arriving time $[a_i, b_i]$.

$$a_i \leq P^k_i \leq b_i \quad \forall \ i \in N \quad \forall \ k \in V,$$

(5)

The time windows studied in this paper is hard, i.e. they must be respected, if the vehicle arrives earlier than its lower bound at a customer $i$, it must wait until the client opens and likewise it is not allowed to arrive later than the upper bound.

$$P^k_i + d_{ij} - P^k_j \leq M(1 - x^k_{ij}) \quad \forall \ i \in N \quad \forall \ j \in N \setminus \{0\} \quad \forall \ k \in V.$$

(6)

with $M$ a great value.

3. Solution method

In this section, we give a detailed exposition of our MALNS algorithm for solving the VRPTW. MALNS is a heuristic solution approach, which integrates the mechanism of the modified choice function (MCF) in the adaptive large neighborhood search (ALNS), in order to guide the research to areas where high-quality solutions are intended by seeking a trade-off between diversification and intensification.

3.1. Adaptive Large Neighborhood Search (ALNS)

The ALNS is a metahuristic proposed by Ropke and Pisinger in 2006 [21] as an extension of the Large Neighborhood Search (LNS) heuristic presented by Shaw (1998) [23]. It is an adaptive approach employed to ameliorate an incumbent solution. The basic idea of this algorithm is to improve a given initial solution by exploring many large neighborhoods. This can be done by the application of various destroy and repair operators. More precisely, the destroy operator removes nodes from the solution. We obtain then an incomplete or infeasible solution $d(x)$. The repair operator reinserts the removed nodes at more favored position which leads to a new feasible (complete) solution $r(d(x))$. The resulting solution will be accepted or rejected based on a Hill Climbing acceptance criteria which only accepts solutions that are better than the current one. This process will be terminated when a stop criterion is met.

It is worth mentioning that ALNS uses an adaptive layer in the step where the operators are selected, by using a score $\phi_j$ which measures the best performance of an operator during the search to choose the most successful one. The adaptiveness lies in a roulette wheel mechanism which selects an operator $j$ with a probability $\frac{\phi_j}{\sum_i \phi_i}$. During $M$ iterations, the score $\phi_j$ is updated and the probabilities of selecting an operator are recalculated.

Here is the detailed ALNS algorithm:
Algorithm 1 Adaptive Large Neighborhood Search

Construct a feasible solution \( x \); set \( x^b = x \)

repeat
  Choose a destroy neighborhood \( d \) and a repair neighborhood \( r \) using roulette wheel selection based on previously obtained scores \( \pi_j \)

  Generate a new solution \( x^t \) from \( x \) using the heuristics corresponding to the chosen destroy and repair neighborhoods

  if \( x^t \) can be accepted then
    \( x = x^t \)
  end if

  if \( c(x^t) < c(x) \) then
    \( x^b = x^t \)
  end if

  Update scores \( \pi_j \) of \( d \) and \( r \)

until Stop criteria is met

return \( x^b \)

3.2. The modified choice Function

The Modified Choice Function (MCF) is an efficient technique presented by Drake [12] as an extension of the original choice function of Cowling [8]. The idea behind this method is to dynamically control the selection of heuristics on the basis of a combination of three different measures. Thereby, the heuristic to be selected must have the higher score \( F_t \).

The first measure \( f_1 \) reflects the past performance of each single heuristic. This measure is represented by the equation:

\[
f_1(h_j) = \sum_n \phi^{n-1}_t \frac{I_n(h_j)}{T_n(h_j)}
\]

where \( I_n(h_j) \) presents the change in fitness function, \( T_n(h_j) \) is the time it takes the heuristic \( h_j \) to produce a solution for an invocation \( n \), and \( \phi \) is a parameter from the interval \([0, 1]\) highlighting the recent performance.

The second measure \( f_2 \) tracks the dependency between a pair of heuristics \((h_k, h_j)\), by considering their past performance when selected consecutively. The formula of this measure is given as follows:

\[
f_2(h_j) = \sum_n \phi^{n-1}_t \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)}
\]

where \( I_n(h_k, h_j) \) presents the change in fitness function, \( T_n(h_k, h_j) \) is the time it takes to call the heuristic \( h_j \) immediately after \( h_k \) for an invocation \( n \).

The third measure \( f_3 \) notes the elapsed time \( (\tau(h_j)) \) since an heuristic \( h_j \) was last called. This gives the heuristics which are inactive for certain time, an opportunity to be selected.

\[
f_3(h_j) = \tau(h_j)
\]

The formulation of the modified choice function is given as follows:

\[
F_t(h_j) = \phi_t f_1(h_j) + \phi_k f_2(h_k, h_j) + \delta_t f_3(h_j)
\]

where \( t \) denotes the number of invocations of heuristic \( h_j \) indicating an improvement by the used heuristic.

The measures \( f_1 \) and \( f_2 \) bring intensification to the search process while the measure \( f_3 \) supports diversification by giving a chance to inactive heuristics to be selected. This is possible by the incorporation of the parameters \( \phi_t \) and \( \delta_t \). Where \( \phi_t \) is an intensification parameter which weights \( f_1 \) and \( f_2 \) respectively, and \( \delta_t \) is the relative weight to \( f_3 \) and hence it is defined to control the diversification degree. At each iteration, if the objective value improves, the value of \( \phi_t \) is increased.
while $\delta_t$ is concurrently decreased. Conversely, $\phi_t$ is decreased and $\delta_t$ is increased when the objective value does not improve. The parameters $\phi_t$ and $\delta_t$ are expressed in the following way:

$$\phi_t(h_j) = \begin{cases} 
0.99, & \text{if the objective value improves} \\
\max\{\phi_{t-1} - 0.01, 0.01\}, & \text{otherwise}
\end{cases}$$

and

$$\delta_t(h_j) = 1 - \phi_t(h_j)$$

3.3. The Modified Adaptive Large Neighborhood Search (MALNS)

The idea behind the MALNS is to efficiently explore the search space using many large neighborhoods. The task consists on incorporating the modified choice function into the selection phase of the ALNS while sparing the whole process. In other words, the destroy and repair operators will not be selected by the roulette wheel mechanism as in the original ALNS, but they will be rather selected by the MCF. Here is the detailed algorithm of the MALNS method:

**Algorithm 2 Modified Adaptive Large Neighborhood Search**

Construct a feasible solution $x$; set $x^b = x$

repeat

Choose a destroy neighborhood $d$ and a repair neighborhood $r$ using the modified choice function based on the obtained scores $F_t$

Generate a new solution $x^t$ from $x$ using the heuristics corresponding to the chosen destroy and repair neighborhoods

if $x^t$ can be accepted then

$x = x^t$

end if

if $c(x^t) < c(x)$ then

$x^b = x^t$

end if

Update scores $F_t$ of $d$ and $r$

until Stop criteria is met

return $x^b$

For the sake of completeness, we will describe the initialization step as well as the removal and insertion operators involved in the destroy/repair block to ensure the diversity during the searching process.

To deal with the initial solution, we used the greedy insertion heuristic introduced by Solomon [24]. This method consists of finding the best location of a given node by testing the different possible configurations. More explicitly, the algorithm selects the best possible insertion place in the present route for each non inserted node under two considerations: the increase in total cost of the present route after the insertion, and the delay of service start time of the client following the new inserted client. This process ends when all deleted nodes will be inserted.

During the phase of destruction, we adopt three different removal heuristics. The first operator is referred to as proximity operator. This operator aims to delete a set of customers that are similar in terms of a spatio-temporal measure ([22] and [23]). In the same manner, the route portion operator gives more flexibility of change on the routes by selecting a pivot client attributed to a route and remove it with its adjacents. The third operator is known as longest detour operator and tries to remove the customers that lead to the largest increase of the cost of the current solution. For more details about those operators, we refer the reader to [19] and references therein.

After the destruction phase, the obtained solution must be repaired in order to get a feasible solution. Therefore, we use two repair operators. First, the greedy insertion [24] tries to select the location that reduces the cost of insertion over all nodes and routes. Second, the regret insertion [11] defines a regret value which is the cost difference of inserting the customer $i$ in its best route and its second best route. Thereby, customers with the highest value should be inserted first.
4. Computational experiments

In order to examine the performance of the proposed MALNS, we accomplished several computational tests. The algorithm was examined on a group of small instances in reference to Solomon’s benchmark [24], and its extension the instances of Gehring and Homberger’s benchmark [15]:

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

As the examined algorithms are based on the related performance, the Modified Adaptive Large Neighborhood Search (MALNS) and the Adaptive Large Neighborhood Search (ALNS) are compared to the competition entries independently. Therefore, both of algorithms use three distinct destroy operators namely the proximity operator, the route portion operator and the longest detour operator. And two repair operators which are the greedy insertion and the regret insertion.

The algorithms were implemented in Java 7, compiled with Intel compiler Celeron 1.80 GHz core i5 with 8GB RAM. The MALNS approach was run for 15600 iterations and was applied 10 times to each instance. In this section, we will give the description of the impact of our method following different parameters: objective function and execution time.

4.1. Objective function

Table 1 illustrates the MALNS improvement results in terms of objective value for all instance groups compared to the ALNS algorithm. The first column defines the instance group, the second column denotes the objective value obtained of the initial solution, the third column contains the results of the ALNS, and the fourth column our MALNS algorithm results.
Table 1: Comparison of objective functions between the ALNS and MALNS

| Instance | Initial solution | ALNS solution | MALNS solution |
|----------|------------------|---------------|----------------|
| R101     | 1747.12          | 1650.80       | 1645.79        |
| C101     | 929.21           | 828.94        | 828.94         |
| RC101    | 1793.01          | 1708.80       | 1701.21        |
| R201     | 1310.64          | 1253.23       | 1253.23        |
| C201     | 682.52           | 591.56        | 591.56         |
| RC201    | 1516.79          | 1406.94       | 1406.94        |
| R121     | 4896.01          | 4819.12       | 4796.26        |
| C121     | 2807.20          | 2704.57       | 2704.57        |
| RC121    | 3725.37          | 3606.06       | 3606.06        |
| R221     | 4603.41          | 4513.10       | 4483.16        |
| C221     | 2063.76          | 1931.44       | 1931.44        |
| RC221    | 3681.32          | 3605.40       | 3427.37        |
| R141     | 10741.11         | 10639.75      | 10512.51       |
| C141     | 7306.13          | 7152.06       | 7152.06        |
| RC141    | 9652.33          | 9127.15       | 8724.36        |
| R241     | 9961.12          | 9758.46       | 9594.32        |
| C241     | 4837.71          | 4116.33       | 4116.33        |
| RC241    | 7949.65          | 7471.01       | 7003.61        |
| R161     | 23027.04         | 22838.65      | 22145.03       |
| C161     | 14296.24         | 14095.64      | 14095.64       |
| RC161    | 18215.52         | 17924.88      | 17432.87       |
| R261     | 22210.38         | 21945.30      | 19806.15       |
| C261     | 8344.17          | 7977.98       | 7977.98        |
| RC261    | 14967.21         | 14817.72      | 14111.79       |
| R181     | 39891.04         | 39315.92      | 38614.81       |
| C181     | 25774.19         | 25184.38      | 25184.38       |
| RC181    | 32626.95         | 32268.95      | 31316.12       |
| R281     | 34260.80         | 33816.90      | 30641.10       |
| C281     | 12212.17         | 11687.06      | 11687.06       |
| RC281    | 23878.06         | 23289.40      | 22116.02       |
| R1101    | 58014.95         | 56903.84      | 55243.64       |
| C1101    | 43154.49         | 42488.66      | 42488.68       |
| RC1101   | 49389.09         | 48702.83      | 47530.42       |
| R2101    | 46357.93         | 45427.58      | 43994.26       |
| C2101    | 17535.18         | 16879.24      | 16879.24       |
| RC2101   | 35908.40         | 35073.70      | 33104.52       |

From the table, we can observe that the MALNS method outperforms the ALNS method, and yields better results in terms of solution quality, the gain average percent can reach 10.36%.

4.2. Execution time

The table 2 presents the execution time of our method compared to ALNS. We computed then the gain average percent (GAP) which is the absolute difference between execution time of the MALNS and the ALNS divided by the magnitude of the execution time of the ALNS.
Table 2: Comparison of runtime in seconds between the ALNS and MALNS

| Instance | ALNS (s) | MALNS (s) | Gap (%) |
|----------|----------|-----------|---------|
| R101     | 368      | 452       | 22.82   |
| C101     | 325      | 396       | 21.84   |
| RC101    | 351      | 430       | 22.50   |
| R201     | 422      | 510       | 20.85   |
| C201     | 365      | 437       | 19.72   |
| RC201    | 415      | 500       | 20.48   |
| R121     | 490      | 568       | 15.91   |
| C121     | 435      | 504       | 15.86   |
| RC121    | 454      | 525       | 15.63   |
| R221     | 588      | 679       | 15.47   |
| C221     | 505      | 581       | 15.04   |
| RC221    | 530      | 613       | 15.66   |
| R141     | 581      | 655       | 12.73   |
| C141     | 560      | 627       | 11.96   |
| RC141    | 501      | 564       | 12.57   |
| R241     | 736      | 824       | 11.95   |
| C241     | 703      | 787       | 11.94   |
| RC241    | 718      | 801       | 11.55   |
| R161     | 556      | 606       | 8.99    |
| C161     | 680      | 741       | 8.97    |
| RC161    | 619      | 672       | 8.56    |
| R261     | 749      | 816       | 8.94    |
| C261     | 817      | 890       | 8.93    |
| RC261    | 788      | 858       | 8.88    |
| R181     | 692      | 726       | 4.91    |
| C181     | 708      | 743       | 4.94    |
| RC181    | 697      | 731       | 4.87    |
| R281     | 852      | 894       | 4.92    |
| C281     | 937      | 983       | 4.90    |
| RC281    | 917      | 962       | 4.90    |
| R1101    | 884      | 901       | 1.92    |
| C1101    | 910      | 928       | 1.97    |
| RC1101   | 903      | 921       | 1.99    |
| R2101    | 1318     | 1344      | 1.97    |
| C2101    | 1361     | 1388      | 1.98    |
| RC2101   | 1331     | 1357      | 1.95    |

It is clear from the table above, that our MALNS takes more time of execution compared to the ALNS, with a percentage gap of an average between 22.84% and 1.95%. This can be explained by the fact that the Modified choice function adopts three different measures to choose the adequate operator to use, in the contrast of the roulette wheel selection which uses only the recent performance of the operator to favor the most appropriate one. When the instance size increases, the execution time of our approach converges to the ALNS one.

5. Conclusion
This paper presents a modified adaptive large neighborhood search (MALNS) to tackle the vehicle routing problem with time windows (VRPTW). In this context, we integrate the modified choice function as selection mechanism in the ALNS method which appears well-suited for the VRPTW.
The competitiveness of our proposed method is demonstrated on the Solomon’s benchmark instance of VRPTW. We conclude that our MALNS algorithm outperforms the classical ALNS in terms of solution quality.

Future work will focus on the application of the multithreading parallelization technique to the MALNS algorithm, in order to develop fast optimization procedure able of reacting to changes in problem information in real time. We believe that this study will lead to an improved execution time of the MALNS method.

Acknowledgement

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
[1] Afifi S, Dang D C and Moukrim A 2013 A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints International Conference on Learning and Intelligent Optimization: Lecture Notes in Computer Science vol 7997
[2] Ben Ticha H, Abdi N, Felleti D and Quilliot A 2017 Empirical analysis for the VRPTW with a multigraph representation for the road network Computers & Operations Research 88 103–116
[3] Braekers K, Ramaekers K and Nieuwenhuyse I 2015 The Vehicle Routing Problem: State of the Art Classification and Review Computers & Industrial Engineering 99
[4] Bräysy O and Gendreau M 2002 Tabu Search heuristics for the Vehicle Routing Problem with Time Windows Top Journal 10 211-237
[5] Bräysy O and Gendreau M 2005 Vehicle Routing Problem with time windows, Part II: Metaheuristics Journal of the Operational Research Society 56 195-215
[6] Burke E K, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E and Qu R 2013 Hyper-heuristics: a survey of the state of the art Journal of the Operational Research Society 64 1095-1274
[7] Chiang W C and Russell R A 1996 Simulated annealing metaheuristics for the vehicle routing problem with time windows Annals of Operations Research 63 3-27
[8] Cowling P I, Kendall G and Soubeiga E 2001 A Hyperheuristic Approach to Scheduling a Sales Summit In: Burke, E., Erben, W. vol 2079, ed PATAT 2000 pp 176-190
[9] Dantzig G and Ramser J 1959 The Truck Dispatching Problem Management Science 6 80-91
[10] Dhahri A, Mjirda A, Zidi K and Ghedira K 2016 A VNS-based Heuristic for Solving the Vehicle Routing Problem with Time Windows and Vehicle Preventive Maintenance Constraints Procedia Computer Science 80 1212–1222
[11] Diana M and Dessouky M 2004 A New Regret Insertion Heuristic for Solving Large-Scale Dial-a-Ride Problems with Time Windows Transportation Research Part B: Methodological 38 539–557
[12] Drake J, Özcan E and Burke E 2012 An Improved Choice Function Heuristic Selection for Cross Domain Heuristic Search Lecture Notes in Computer Science 7492 307–316
[13] Eksioglu b, Vural A V and Reisman A 2009 The vehicle routing problem: A taxonomic review Computers & Industrial Engineering 57 1472–1483
[14] Gambardella L M, Taillard E, Agazzi G 1999 MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows New Ideas in Optimization 63–76
[15] Gehring H and Homberger J 1999 A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows In Proceedings of EUROGEN99 pp 57-64. http://sintef.no/projectweb/top/vrptw/Homberger-benchmark/
[16] Gutin G and Punnen A 2002 The Traveling Salesman Problem and Its Variations Eds A.P.
[17] Kallehauge B, Larsen J, Madsen O B and Solomon M M 2005 Vehicle Routing Problem with Time Windows ed Column Generation 1 67-98
[18] Mladenovic N and Hansen P 1997 Variable neighborhood search Computers & Operations Research 24 1097–1100
[19] PrescottGagnon E, Desaulniers G and Rousseau L M 2009 A branch and price based large neighborhood search algorithm for the vehicle routing problem with time windows Networks 54 190-204
[20] Potvin J Y, Kervahut T, Garcia B L, Rousseau J M 1996 The vehicle routing problem with time windows part I: tabu search INFORMS Journal on Computing 8 158–164
[21] Ropke S and Pisinger D 2006 An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows Transportation Science 40 455–472
[22] Shaw P 1997 A new local search algorithm providing high quality solutions to vehicle routing problems Technical Report
[23] Shaw P 1998 Using constraint programming and local search methods to solve vehicle routing problems In: Maher M., Puget JF. (eds) Principles and Practice of Constraint Programming CP98. Lecture Notes in Computer Science 1520 pp 417–431
[24] Solomon M 1987 Algorithms for the Vehicle Routing and Scheduling Problem with time Window Constraints Operations Research Journal 35 254-265
[25] Taillard É, Badeau P, Gendreau M, Guertin F and Potvin J Y 1997 A tabu search heuristic for the vehicle routing problem with soft time windows Transportation Science 31 170–186

[26] Tan X, Zhuo X and Zhang J 2006 Ant Colony System for Optimizing Vehicle Routing Problem with Time Windows (VRPTW) In: Computational Intelligence and Bioinformatics. ICIC 2006. Lecture Notes in Computer Science 4115 33–38

[27] Thangiah S 1995 Vehicle routing with time windows using genetic algorithms In Application Handbook of Genetic Algorithms: New Frontiers 253–277

[28] Toth P and Vigo D 2002 An overview of vehicle routing problems 9 of SIAM Monographs on Discrete Mathematics and Applications SIAM, Philadelphia vol 1 pp 1–26

[29] Vidal T, Laporte G and Matl P 2019 A concise guide to existing and emerging vehicle routing problem variants European Journal of Operational Research 286 401–416