Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 66 (2018) 207–214 www.elsevier.com/locate/endm

Variable Neighborhood Search for Vehicle Routing Problem with Multiple Time Windows

Huggo Silva Ferreira [a,1] Eduardo Theodoro Bogue [a,b,2]
Thiago F. Noronha [a,3] Slim Belhaiza [c,4] Christian Prins [d,5]

[a] *Department of Computer Science, Universidade Federal de Minas Gerais, Av. Antonio Carlos 6627, Belo Horizonte, MG 31270-010, Brazil*

[b] *Universidade Federal de Mato Grosso do Sul, Rua Itibire Vieira, s/n Residencial Julia Oliveira Cardinal BR 463 Km 4,5, Ponta Pora, MS 79907-414, Brazil*

[c] *Department of Mathematics and Statistics, KFUPM, Dhahran 31261, Saudi Arabia*

[d] *ICD-LOSI, UMR CNRS 6281, Universit de Technologie de Troyes, 12 rue Marie Curie, CS 42060, 10004 Troyes Cedex, France*

Abstract

The Vehicle Routing Problem (VRP) with Multiple Time Windows is a generalization of VRP, where the customers have one or more time windows in which they can be visited. The best heuristic in the literature is a Hybrid Variable Neighborhood Tabu Search (HVNTS) that mostly deals with infeasible solutions, because it is assumed that one may not reach some regions of the search space without passing through infeasible solutions. In this short paper, we propose a simpler Variable Neighborhood Search heuristic where all the computational effort is spent on searching for feasible solutions. Computational experiments showed that the proposed heuristic is competitive with the best heuristic in the literature.

Keywords: Variable Neighborhood Search, Vehicle Routing Problem, Multiple Time Windows

https://doi.org/10.1016/j.endm.2018.03.027
1571-0653/© 2018 Published by Elsevier B.V.

1 Introduction

Given a set of customers, where each one has a demand, a service time, and a set of time windows; and a set of vehicles with the same capacity and the same maximum route duration. The Vehicle Routing Problem with Multiple Time Windows (VRPMTW) [3] consists in defining a set of routes that begin and end at a single depot, where customers are visited only once and their service time begins at any of their respective time windows. The objective is to minimize a function that combines the size of the fleet used and the time it takes to visit all clients and return to the depot [1].

VRPMTW is formally defined as follows. Let $G = (V, A)$ be a complete graph, where $V = \{0\} \cup N$ is the set of nodes, with 0 representing the depot and $N = \{1, 2, ..., |N|\}$ representing the customers; and A is the set of arcs, with each arc $(i, j) \in A$ associated with a travel time t_{ij}. Each node $i \in N$ is associated with a demand q_i, a service time s_i, and a set of time windows J_i, where each window $[l_i^p, u_i^p] \in J_i$ defines a time period in which the service of customer i can begin. Besides, let R be a set of vehicles, where each vehicle $k \in R$ is associated with a capacity Q and a maximum route duration D. It is assumed that $|R| \geq |N|$. Each customer $i \in N$ must be visited once by one of the vehicles $k \in R$. If a vehicle k arrives at a customer i outside any of its time windows, it must wait a time w_i^k until the beginning of the next time window of i. Besides, the sum of the demands of all customers visited in the route of a vehicle $k \in R$ cannot be larger than the vehicle capacity Q. In addition, routes must begin and end at node 0, and the sum of the travel time, the waiting time, and service time of all customers visited in the route of a vehicle $k \in R$ cannot be larger than the maximum route duration D. The objective function, proposed in [3] and used in [1], consists in minimizing the total travel time and the number of vehicles used, as stated in Equation (1), where $x_{ij}^k = 1$ if arc $(i, j) \in A$ is in the route of vehicle $k \in R$. Besides, $r^k = 1$ if the route of vehicle $k \in R$ is not empty, otherwise $r^k = 0$, and F is a fixed cost, expressed in time units, of using a vehicle k.

$$f(x) = \sum_{k \in R} \sum_{i \in V, j \in V \setminus \{i\}} t_{ij} x_{ij}^k + \sum_{k \in R} \sum_{i \in V} w_i^k + F \sum_{k \in R} r^k \tag{1}$$

VRPMTW is a generalization of the Capacitated Vehicle Routing Problem

[1] Email: `huggosf@ufmg.br`
[2] Email: `eduardotheodoro@ufmg.br`
[3] Email: `tfn@dcc.ufmg.br`
[4] Email: `slim.belhaiza@gerad.ca`
[5] Email: `christian.prins@utt.fr`

[2]. Therefore, it is also NP-Hard. A Multiple Ant Colony System heuristic (MACS) was proposed in [3]. However, the running times of this heuristic increases significantly with the number of time windows. Another ILP formulation was proposed in [1]. As this formulation can only be efficiently solved for small instances, a Hybrid Variable Neighborhood Tabu Search (HVNTS) heuristic is also proposed. The constraints regarding the vehicle capacity, the time windows, and the maximum route duration are relaxed and added as fixed penalties in the objective function. These penalties are supposed to be large enough so that infeasible solutions are allowed but discouraged if similar feasible solutions exist. The shaking phase is based on five neighborhoods and the local search phase consists of a tabu search based on eight neighborhoods. Although feasible solutions are not guaranteed to be found, computational experiments showed that feasible solutions were found for all instances tested. These experiments also showed that HVNTS outperforms MACS. Therefore, it is the best heuristic in the literature for VRPMTW.

In this paper, we propose a Variable Neighborhood Search (VNS) heuristic that only deals with feasible solutions. Therefore, no computational effort is spent on infeasible solutions and feasible solutions are guaranteed to be found. With this heuristic, we aim at showing that different regions of the search space, where good solutions are found, can be reached without passing through infeasible solutions.

The remainder of this work is organized as follows. The proposed VNS heuristic is discussed in Section 2. Computational Experiments that compare the results of VNS with those of HVNTS are reported in Section 3. Concluding remarks are drawn and future works are discussed in the last section.

2 Variable Neighborhood Search for VRPMTW

The neighborhood operators used in the VNS heuristic were first applied to VRPMTW in [1]. They are divided into *single-route operators* and *multi-route operators*. The former changes a single route, while the latter modifies at least two routes.

There are two single-route operators. The *single-route relocate* operator consists in removing a customer from its current position and inserting it into another position in the same route. The *single-route 2-exchange* operator consists in exchanging the position of any two customer in a single route.

There are five multi-route operators. The *multi-route relocate* operator consists in removing a customer from its current position and inserting it into a new position in another route. The *multi-route swap* operator consists in exchanging the position of two customers from two distinct routes. The *multi-route 3-node swap* operator consists in exchanging the position of three customers $\langle i_1, i_2, i_3 \rangle$ from three distinct routes, such that i_2 assumes the position of i_1, i_3 assumes the position of i_2, and i_1 assumes the position of i_3. The *multi-route cross* operator consists in exchanging the positions of two pairs of consecutive customers $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$ from two distinct routes, such that i_1 swaps position with i_2 and j_1 swaps position with j_2. The *multi-route 3-Exchange* operator consists in exchanging the positions of three pairs of customers $\langle i_1, j_1 \rangle$, $\langle i_2, j_2 \rangle$ and $\langle i_3, j_3 \rangle$, from three distinct routes, such that $\langle i_2, j_2 \rangle$ assumes the position of $\langle i_1, j_1 \rangle$, $\langle i_3, j_3 \rangle$ assumes the position of $\langle i_2, j_2 \rangle$, and $\langle i_1, j_1 \rangle$ assumes the position of $\langle i_3, j_3 \rangle$.

The heuristic proposed in this paper is composed of three stages. In every stage, only single-route or multi-route operations that result in feasible solutions are allowed. First, a greedy heuristic builds a feasible solution. Next, the Route Minimization heuristic is run to decrease the number of non-empty vehicle routes in this solution. Then, the latter is used as the initial solution of the VNS heuristic.

The constructive heuristic is based on the Insertion Heuristic for VRPTW [4]. It starts with a partial solution where the route of every vehicle in R is empty. At each iteration, the customers not already present in the partial solution are identified, and the incremental cost of inserting each of them in every feasible position in this solution is evaluated. Then, the customer with the smallest incremental cost is inserted in its best possible position. This procedure stops when all customers are inserted in the solution. As $|R| \geq |N|$, the solutions provided by this heuristic are feasible for VRPMTW.

The Route Minimization heuristic aims only to minimize the number of non-empty vehicle routes in the initial solution S. This algorithm is composed of a *Route Elimination* procedure and a *Perturbation* procedure. The $RouteElimination(k, S)$ procedure tries to remove all customers from the route of a vehicle k in a solution S by applying systematically the multi-route relocate and swap operators to the customers in the route of k.

If the route elimination procedure is not able to empty the route of k, the $Perturbation(k, S)$ procedure is applied to randomly change the routes of the vehicles in $R^S \setminus \{k\}$, hoping that the customers in the route of k can be reallocated in the resulting solution. This procedure consists in randomly selecting one of the five multi-route neighborhood operators described above and applying it α times to randomly selected customers. The pseudocode of $RouteMinimization(S)$ is presented in Algorithm 1.

Algorithm 1 $RouteMinimization(S)$

```
1: Let $R^S$ be the set of vehicles with non-empty routes in $S$.
2: $k \leftarrow argmin_{k' \in R^S} size(k', S)$
3: $RouteElimination(k, S)$
4: while Stopping condition not met do
5:     if $size(k, S) > 0$ then $Perturbation(k, S)$
6:     $RouteElimination(k, S)$
7:     if if $size(k) = 0$ or Stagnation condition is met then
8:         $R^S \leftarrow R^S \setminus \{k\}$
9:         $k \leftarrow argmin_{k' \in R^S} size(k', S)$
10:    end if
11: end while
12: return $S$
```

The set R^S of vehicles with non-empty routes in S is initialized in line 1, and the vehicle $k \in R^S$ whose route has the smallest number of customers in S is identified in line 2, where $size(k', S)$ denotes the number of customers in the route of k. Then, the route elimination procedure is first applied to k and S in line 3. The loop of lines 4 to 11 alternates between the perturbation procedure and the route elimination procedure. The former is applied to k and S in line 5, if the route of k is not empty, and the latter is applied to k and S in line 6. If the route of k is empty or a stagnation condition is met, the heuristic changes the route being eliminated. It removes k from R^S in line 8, and a new vehicle whose route has the smallest number of customers in R^S is selected in line 9. The stagnation condition is triggered after β iterations without emptying the route of k. Let $T = \frac{1}{Q} \sum_{i \in N} q_i$ be a lower bound to the minimum number of vehicles required to ship all the customer demands. The stopping condition is met when the number of non-empty routes in S is equal to T or after θ iterations of the loop in lines 4-11. By the end of this loop, the current solution S is return in line 12. In this paper, the value of β and θ was set to 200 and 700, respectively, and the value of α varies from 5 to 20.

The VNS heuristic aims to minimize the sum of the route durations over all vehicles in the solution returned by the Route Minimization heuristic, without increasing the number of non-empty routes. This algorithm is composed of a *shaking* procedure and a *Variable Neighborhood Descent (VND)* procedure. The $Shaking(S, n)$ procedure randomly chooses between the multi-route relocate and multi-route swap operators, and applies the chosen operator n times to randomly chosen customers in S. The $VND(S')$ procedure is applied to the solution S' returned by the shaking procedure. It consists of a classic variable neighborhood descent heuristic, composed of seven depth first local searches, based on the seven neighborhoods operator described above. The local searches are sorted in the VND in the same order they are defined above. Besides, after the last local search is performed, a restricted version of the Route Elimination procedure is applied, if the number of routes in the current solution is larger than the lower bound T to minimum number of vehicles required to ship all the customer demands. The pseudocode of VNS heuristic is presented in Algorithm 2.

Algorithm 2 VNS(S)

```
1: $S^* \leftarrow S$
2: $n \leftarrow \eta_{min}$
3: while Stopping condition is not met do
4:     $S' = Shaking(S, \text{n})$
5:     $S' \leftarrow VND(S')$
6:     if $f(S') < f(S)$ then
7:         $S \leftarrow S', S^* \leftarrow best(S, S^*), n \leftarrow \eta_{min}$
8:     else if $n < \eta_{max}$ then
9:         $n \leftarrow n + 1$
10:    end if
11:    if Stagnation condition is met then
12:        Let $\Gamma$ be the set of the best $\gamma$ solutions found so far.
13:        Set $S$ as a random solution from $\Gamma$.
14:        $n \leftarrow \eta_{min}$
15:    end if
16: end while
17: return $S*$
```

The best known solution S^* is initialized in line 1, while the shaking size n is assigned to its minimum value η_{min} in line 2. At each iteration of the loop in lines 3-18, a new solution, that is a local optimum for the seven neighborhoods described above, is generated. Each of this solution is generated by applying the shaking procedure to the current solution S in line 4, and then applying VND procedure to the resulting solution S' in line 5. If the new solution S' is better than S (line 6), the current and the best known solutions, as well as the shaking size, are updated in line 7. Otherwise, the value of n is incremented in line 9 if n is smaller than the maximum shaking size η_{max}. If a stagnation condition is trigged in line 11, a restart occurs in lines 12 and 13, and the current solution is replaced by a solution randomly chosen among the γ best solutions found so far. In this case, the perturbation size n is also reset to its smallest value in line 14. The stagnation condition is triggered after λ iterations without improving the best known solution, the stopping condition is set to δ iterations of the loop in lines 3-16. By the end of this loop, the best known solution S^* is return in line 17. In this paper, the values of γ, λ and δ were set to 10, 60 and 1200, respectively, and the values of η_{min} and η_{max} were set to 5 and 20, respectively.

3 Numerical results and concluding remarks

This section compares the results of our VNS heuristic, that only explores feasible regions of the search space, with those of the HVNTS heuristic [1], that deals with infeasible solutions. The former was implemented in C++, and compiled with the GNU *gcc* version 6.3. HVNTS was also implemented in C++ and its results were obtained by the executable code provided by the authors of [1]. Computational experiments were performed in a single core of an Intel i5-3230M machine with $2.60GHz$ of clock speed and $8GB$ of RAM.

In this paper, we show results for the 72 instances proposed in [1]. They are divided into two sets where each set is composed of six group of instances. In the first set, there is no time window overlaps. There are 8 instances at each of the six groups, named rm1, rm2, cm1, cm2, rcm1, and rcm2. In the second set, there might be time window overlaps. There are 4 instances at each of the six groups, named prm1, prm2, pcm1, pcm2, prcm1, and prcm2. The comparison between HVNTS and VNS for these instances is presented in Table 3.

	HVNTS		VNS		
Group	$f(S)$	time (s)	$f(S)$	time (s)	%dev
rm1	2831.12	108.96	2863.38	145.96	-1.15
rm2	3525.06	107.95	2945.88	31.62	15.43
cm1	3453.47	159.36	3660.57	154.32	-6.16
cm2	5098.53	147.60	4482.88	135.78	12.06
rcm1	3468.88	94.89	3490.50	143.50	-0.62
rcm2	3793.33	140.04	3423.13	69.70	9.70
prm1	2695.59	111.46	2692.25	235.48	0.12
prm2	2743.74	133.24	2869.00	59.63	-4.57
pcm1	2867.45	143.49	2846.75	145.22	0.72
pcm2	2751.69	157.00	2918.00	105.30	-6.05
prcm1	2810.87	101.85	2902.00	49.58	-3.25
prcm2	2772.08	181.55	3219.25	60.99	-16.04
				Average:	0.02

Table 1
Comparison between HVNTS and VNS for the instances proposed in [1].

The name of each instance group is given in the first column. Average solution cost and the average running times over ten independent runs of HVNTS are reported in the second and third columns, respectively. The same data is shown for VNS in the forth and fifth columns, respectively. The percent relative deviation %dev = $(f^{TS} \cdot f^{VNS})/f^{TS}$ of the average solution cost of HVNTS (f^{TS}) and that of VNS (f^{VNS}) is given in the last column. We note that positive values of this metric means that VNS outperforms HVNTS. One can see that the average relative deviation over all instance groups tested was 0.02% which point out to the fact that the results of VNS are competitive with those of HVNTS, the best heuristic in the literature, which mostly deals with infeasible solutions. The results showed in this short paper indicate that different regions of the search space, where good solutions are found, can be reached without passing through infeasible solutions, which may guide the design of future research on VRPMTW.

Acknowledgement

This work was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq), the Foundation for Support of Research of the State of Minas Gerais, Brazil (FAPEMIG), and Coordination for the Improvement of Higher Education Personnel, Brazil (CAPES)

References

[1] Belhaiza, S., P. Hansen and G. Laporte, *A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows*, Computers & Operations Research **52** (2014), pp. 269–281.

[2] Dantzig, G. B. and J. H. Ramser, *The truck dispatching problem*, Management Science **6** (1959), pp. 80–91.

[3] Favaretto, D., E. Moretti and P. Pellegrini, *Ant colony system for a vrp with multiple time windows and multiple visits*, Journal of Interdisciplinary Mathematics **10** (2007), pp. 263–284.

[4] Solomon, M. M., *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Operations Research **35** (1987), pp. 254–265.