A GRASP/VND Heuristic for the Heterogeneous Fleet Vehicle Routing Problem with Time Windows

Lucía Barrero, Franco Robledo,
Pablo Romero, and Rodrigo Viera

Instituto de Computación, INCO
Facultad de Ingeniería - Universidad de la República
Montevideo Uruguay
(lucia.barrero,frobledo,promero,rodrigo.viera)@fing.edu.uy

Abstract. The Heterogeneous Fleet Vehicle Routing Problem with Time Windows (HFVRPTW) is here introduced. This combinatorial optimization problem is an extension of the well-known Vehicle Routing Problem (VRP), which belongs to the $\mathcal{NP}$-Hard class. As a corollary, our problem belongs to this class, a fact that promotes the development of approximative methods.
A mathematical programming formulation for the HFVRPTW is presented, and an exact solution using CPLEX is performed. A GRASP/VND methodology is also developed, combining five different local searches. The effectiveness of our proposal is studied in relation with the exact solver. Our proposal outperforms the exact CPLEX in terms of CPU times, and finds even better solutions under large-sized instances, where the exact solver halts after ten hours of continuous execution.

Keywords: Combinatorial Optimization Problem, Vehicle Routing Problem, HFVRPTW, Computational Complexity, GRASP, VND.

1 Motivation

The transport industry employs more than 10 million people and it represents roughly the 5% of the Gross Domestic Product (GDP) of the European Union. Furthermore, logistics such as transport and storage account for 10%-15% of the cost of a finished product. In practice, this means that even a small relative reduction in the cost of logistics and transportation means huge savings.

Usually, large-scale corporations in the transport sector are mostly dedicated to savings, and an efficient delivery of goods and services. However, transport also represents an important source of CO_2 emissions, and traffic congestion. In synthesis, a smart vehicle routing engineering is not only meaningful in terms of savings, but also implies a responsible care of the environment.

Operational researchers are engaged with society, and try their best to develop mathematical models that are suitable for realistic transportation problems. A celebrated combinatorial problem is known as the Traveling Salesman Problem, or TSP. We are given non-negative costs in the edges of a complete network, and the goal is to find the cheapest Hamiltonian tour (i.e., visiting all the nodes of the network). The decision version for the TSP belongs to the class of $\mathcal{NP}$-Complete problems, and it is included in Karp list [8]. A natural generalization is the Vehicle Routing Problem, or VRP. In the VRP, we are given a fleet of vehicles, and we should determine the optimal set of routes in order to serve a given number of customers, starting and ending at the depot. The reader can appreciate that the TSP is a special VRP with a single vehicle; thus, the VRP belongs to the $\mathcal{NP}$-Hard class. Given its paramount importance, several variations in the basic VRP model appear in the literature, adding time-windows for customer delivery, heterogeneous fleets, one-way or two-way routes, dynamic demands, among many others. The reader can consult the recent survey for the different variants of the VRP and its applicability to different contexts [9].

To the best of our knowledge, there is no model that simultaneously combines heterogeneous fleets and time-windows, with a penalty factor due to overtime. The contributions of this paper can be summarized in the following items:

1. The Heterogeneous Fleet Vehicle Routing Problem with Time Windows (HFVRPTW) is introduced.
2. We formally prove that the HFVRPTW belongs to the $\mathcal{NP}$-Hard class.
3. As a consequence, a GRASP/VND methodology is proposed.
4. A novel mathematical programming formulation for the HFVRPTW is presented. It represents an adaptation of the previous formulation proposed in [7], adding a penalty due to overtime.
5. The effectiveness of our proposal with respect to an exact solution implemented in CPLEX is studied. The activity of the different local searches of our GRASP/VND methodology is also studied.

The document is organized in the following manner. The related work is presented in Section 2. A formal description for the HFVRPTW is presented in Section 3; its $\mathcal{NP}$-Hardness is also established. A GRASP/VND solution is introduced in Section 4. Numerical results are presented in Section 5. Section 6 contains concluding remarks and trends for future work.

2 Related Work

The classical VRP is presented by Dantzig as a generalization of the TSP [4]. The problem is there motivated by fuel distribution, trying to find the optimal routing of a fleet between a depot and several stations. In general, the VRP consists of how to share customers geographically distributed by a given fleet of vehicles, based on one or multiple depots. The goal is to fulfill the customer demands, finding adequate routes starting and ending at the depot. Rapidly, the VRP found an impressive diversity of applications, ranging from transport network design to efficient garbage collectors. Current VRP models include more realistic assumptions (such as traffic congestion and time-windows for the customers), given the greater possibilities in processing resources. In [1], Baldacci presents a framework for exact algorithms useful for several variations of the VRP, such as capacitated VRP, VRP with Time Windows (VRPTW), pick-up and delivery, multi-depot VRP, among others. In the Heterogeneous Fleet VRP, we are given vehicles with different capacities, and the goal is to design a minimum cost solution meeting the customer demands, starting and ending at the central depot. A fixed cost is associated to the vehicle-type, while a variable cost is proportional to the distance of the tours.

An exact Branch and Cut solution for the HFVRP is proposed in [11], adapting the most competitive exact algorithms for the problem such as route enumeration and extended capacity cuts for large-sized instances.

Other works address the VRP with Time-Windows (VRPTW), where the TW have either soft or hard constraints. In the hard constraint, an early vehicle can wait until the customer is available. In a soft constraint, a penalty is carried to the objective when the constraint is not satisfied. Historical works for the soft VRPTW show that an incorrect usage of a Tabu Search the TW can have a negative impact in the cost [10, 14, 17]. A hybrid solution for the VRPTW is proposed in [16], that jointly considers Large Neighborhood Search (LNS) and a Bat Algorithm (BA), inspired by the eco-location of bats. The results were satisfactory, under benchmarks with 100 customers.

In [2], a two-phase solution combines a Construction phase with Tabu Search, to avoid locally optimum solutions. The solution reduce the distances, in a practical industrial application. A hierarchic *cluster-first route-second* solution for a large super-market chain is proposed in [3], with remarkable benefits with respect to a naive solution.

In this work, we combine Heterogeneous Fleet with a new concept of soft constraint with overtime. Our formulation is adapted from the mathematical programming presented in [7]. The reader is invited to consult the recent review on the VRP for other variations of this problem [9].

3 Problem and Complexity

In this section, a formal combinatorial optimization problem is introduced. The hardness of the problem is also established.

3.1 Formulation

The exact formulation is based on the integer linear programming model defined in [7]. However, we consider flexible time-windows instead, where delays are penalized with a cost (i.e,, an additive term in the objective function). Consider a complete graph $G = (V, E)$ where:

- $V = \{0, 1, \ldots, n\}$, being 0 the depot and $N = \{1, ..., n\}$ the customers.
- $E = \{(i, j) : 0 \leq i, j \leq n, i \neq j\}$ represent the links between the nodes.

- t_{ij} is the required time to cross the link (i, j).

All the customers must be visited, and the following information is known for each customer $i \in N$:

- d_i is a fixed demand for customer i.
- s_i represents the required time for a vehicle to service the customer i.
- $[e_i, l_i]$ is the time-window (available and deadline) for customer i. This window is not a hard constraint (a penalty occurs if it is not respected).
- ot_i is the *overtime*, or the tolerance after the deadline. It is found with the following expression: $ot_i = \omega(l_i - e_i)$ for some known factor $\omega : 0 \leq \omega \leq 1$. The extended Time Window (TW) is then $[e_i, l_i + ot_i]$. A penalty occurs if the vehicle meets customer i during the interval $[l_i, l_i + ot_i]$, as shown in red color in Figure 1.

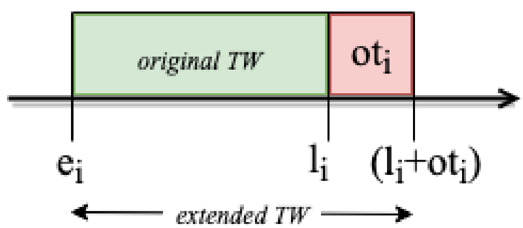

Fig. 1. Original and Extended Time-Windows.

With respect to the depot, we know that:

- $[e_0, l_0] = [E, L]$ is the time-window for the depot.
- $d_0 = s_0 = 0$, since in the depot there is no demand nor service.

The fleet is modeled as $K = \{1, ..., k\}$, C represents the vehicle-types, and S_c the set of c-type vehicles. For each vehicle, we are given:

- q_c is the capacity.
- f_c is its fixed-cost.
- α_c is its variable-cost.
- n_c is the number of available type-c vehicles.

We consider the following set of decision variables:

- $x_{ij}^k = 1$ iff the vehicle k visits the link (i, j); 0 otherwise.
- a_{ik}: time which the vehicle k reaches the customer i.
- o_{ik}: overtime of vehicle k for the customer i.

We also assume that the following parameters are known:

- $M = \max\limits_{(i,j \in V)} (l_i + ot_i + t_{ij} + s_i - e_j)$: represents the longest time consumed between any two customers.
- ρ: represents the penalty associated to overtime.

The HFVRPTW can be formulated as follows:

$$\min \sum_{c \in C} f_c \sum_{k \in S_c} \sum_{j \in N} x_{0j}^k + \sum_{c \in C} \alpha_c \sum_{k \in S_c} \sum_{\substack{i,j \in V, \\ i \neq j}} t_{ij} x_{ij}^k + \sum_{k \in K, i \in N} o_{ik} * \rho \tag{1}$$

s.t.:

$$\sum_{k \in K} \sum_{\substack{j \in V, \\ i \neq j}} x_{ij}^k = 1 \ \forall \ i \in N \tag{2}$$

$$\sum_{j \in N} x_{0j}^k \leq 1 \ \forall \ k \in K \tag{3}$$

$$\sum_{i \in N} x_{i0}^k \leq 1 \ \forall \ k \in K \tag{4}$$

$$\sum_{i \in V} x_{ij}^k = \sum_{i \in V} x_{ji}^k \ \forall \ j \in V, \ k \in K \tag{5}$$

$$\sum_{i \in N} d_i \sum_{\substack{j \in V, \\ i \neq j}} x_{ij}^k \leq q_c \ \forall \ k \in S_c, \ c \in C \tag{6}$$

$$a_{ik} + s_i + t_{ij} - M(1 - x_{ij}^k) \leq a_{jk} \ \forall \ k \in K, \ i \in N, \ j \in V, \ i \neq j \tag{7}$$

$$t_{0i} * x_{0i}^k \leq a_{ik} \ \forall \ k \in K, \ i \in N \tag{8}$$

$$a_{ik} \leq (l_i + ot_i) \sum_{\substack{j \in V, \\ i \neq j}} x_{ij}^k \ \forall \ k \in K, \ i \in N \tag{9}$$

$$e_i \sum_{\substack{j \in V, \\ i \neq j}} x_{ij}^k \leq a_{ik} \leq (l_i + ot_i) \sum_{\substack{j \in V, \\ i \neq j}} x_{ij}^k \ \forall \ k \in K, \ i \in N \tag{10}$$

$$E \leq a_{0k} \leq L + ot_0 \ \forall \ k \in K \tag{11}$$

$$\sum_{k \in S_c} \sum_{j \in N} x_{0j}^k \leq n_c \ \forall \ c \in C \tag{12}$$

$$o_{ik} \geq \max(0, a_{ik} - l_i) \geq 0 \ \forall \ k \in K, \ i \in V \tag{13}$$

$$a_{ik} \geq 0 \ \forall \ k \in K, \ i \in N \tag{14}$$

$$x_{ij}^k \in \{0, 1\} \ \forall \ k \in K, \ (i,j) \in E \tag{15}$$

The objective function 1 is an additive cost, considering fixed and variable costs in the vehicles, as well as penalties related to overtime. Constraints 2 state that all the customers must be visited by only one vehicle. The set of Constraints 3, 4 and 5 represent flow conservation, and state that all the vehicles start and end at the depot. Constraints 6 state that the customer demands cannot exceed the capacities of the vehicles. Constraints 7 state the precedence relation between the arrival times of the vehicles to the customers.

Constraints 8 state the first arrival time to the first node in the route. The set of Constraints 9, 10 and 11 model the time-windows for both the customers and the depot, while Constraints 12 bounds the number of available vehicles for each type. Finally, the set of Constraints 13, 14 and 15 define the domain of the respective decision variables.

3.2 Hardness

The hardness of the corresponding decision version for the HFVRPTW is straight from the $\mathcal{NP}$-Completeness of Hamiltonian Tour. Recall that a graph G is *Hamiltonian* if there exists an elementary cycle $\mathcal{C} \subseteq G$ that contains all the nodes.

Definition 1 (Hamiltonian Tour). *Given a simple graph $G = (V, E)$. Is G Hamiltonian?*

It is known that Hamiltonian Tour belongs to the class of $\mathcal{NP}$-Complete decision problems [5, 8].

Proposition 1. *The HFVRPTW belongs to the $\mathcal{NP}$-Hard class.*

Proof. By reduction from Hamiltonian Tour. Consider an arbitrary graph $G = (V, E)$. We will see that there exists a feasible solution for the HFVRPTW whose cost is not greater than $n = |V|$ if and only if there exists a Hamiltonian tour for G.

Consider an instance of HFVRPTW with the complete graph K_n as a ground graph, where $n = |V|$, a single vehicle with cost $\alpha = 1$ and sufficient capacity $q_c = n$ rooted at some arbitrary depot $v \in V$, no penalties and customers with infinite patience. The time to traverse the links $(i, j) \in E$ is always $t_{i,j} = 1$, but $t_{i,j} = n$ if $(i, j) \notin E$. A feasible solution must be a Hamiltonian tour, and its cost is not greater than n if and only if it is strictly included in $G = (V, E)$. Therefore, the HFVRPTW is at least as hard as Hamiltonian Tour. ■

4 Solution

GRASP and VND are well known metaheuristics that have been successfully used to solve many hard combinatorial optimization problems. GRASP is a powerful multi-start process which operates into two phases. A feasible solution is built in a first phase, whose neighborhood is then explored in the Local Search Phase [13]. The second phase is usually enriched by means of different variable neighborhood structures. For instance, VND explores several neighborhood structures in a deterministic order. Its success is based on the simple fact that different neighborhood structures do not usually have the same local minimum. Thus, the resulting solution is simultaneously a locally optimum solution under all the neighborhood structures. The reader is invited to consult the comprehensive Handbook of Heuristics for further information [6]. Here, we develop a GRASP/VND methodology. The main building-blocks of our $Main$ algorithm are presented in Figure 2. An arbitrary input instance $I = (G, t_{ij}, d_i, s_i, e_i, l_i, ot_i, \omega, K, q_c, f_c, \alpha_c, n_c)$ for the HFVRPTW is considered, where the symbols represent the aforementioned variables in the problem formulation. Observe that the whole GRASP/VND solution is executed *iter* times, and the best solution is returned. The parameter $\alpha \in [0, 1]$ trades greediness for randomization during the *Construction* phase, by means of a Restricted Candidate List (RCL). The VND is composed by five local searches, to know, *FleetOpt*, *Exchange*, *Relocate*, $2 - opt$ and $3 - opt$, in the respective order. In the following paragraphs, we describe the Construction phase, as well as the local searches.

Algorithm 1 $sol = Main(I, iter, \alpha)$

$i \leftarrow 0$; $sol \leftarrow \emptyset$
while $i < iter$ **do**
    $sol \leftarrow Construction(I, \alpha)$
    $\overline{sol} \leftarrow VND(sol, I, FleetOpt, Exchange, Relocate, 2 - opt, 3 - opt)$
    if $cost(\overline{sol}) < cost(sol)$ **then**
        $sol \leftarrow \overline{sol}$
    end if
end while
return sol

Fig. 2. Pseudocode for the $Main$ algorithm.

4.1 Construction Phase

Figure 3 presents a full pseudocode for the *Construction* phase. The following functions are considered:

- $GetClients(data)$: returns the clients in a list for a given dataset.
- $SelectVehicles(vehicles)$: returns a vehicle that is available, and updates the number of available vehicles.
- $GetCapacity(vehicle)$: returns the capacity of a given vehicle.
- $CreateRoute(vehicle, path)$: creates a route using the given *path*. This route is performed with the given *vehicle*.
- $IsFeasible(route, client)$: determines whether it is feasible or not to append the given *client* at the end of the given *route*, or not.

Algorithm 2 $sol = Construction(instance, vehicles)$

```
sol ← φ
clients ← GetClients(instance)
newRoute ← true
while clients ≠ φ do
   candidates ← φ
   if newRoute then
      path ← {depot}
      vehicle ← SelectVehicle(vehicles)
      q ← GetCapacity(vehicle)
      route ← CreateRoute(vehicles, path)
      newRoute ← false
   end if
   for client ∈ clients do
      if IsFeasible(route, client) then
         candidates ← candidates ∪ {client}
      end if
   end for
   if candidates ≠ φ then
      incr(e) ∀ e ∈ candidates
      i_min ← min{incr(e) : e ∈ candidates}
      i_max ← max{incr(e) : e ∈ candidates}
      RCL ← {e ∈ candidates : incr(e) ≤ i_min + α(i_max − i_min)}
      client ← Random(RCL)
      path ← path ∪ {client}
      q = q − GetDemand(client)
      clients ← clients\{client}
   end if
   if candidates = φ ∨ q = 0 then
      path ← path ∪ {depot}
      sol ← sol ∪ {route}
      newRoute ← true
   end if
end while
return sol
```

Fig. 3. Construction Phase

We need to select vehicles and routes for them, in order to build feasible solutions. We collect all the customers that were not yet visited in the variable *clients*. A metric is considered to decide the priority for the different vehicles. The route is then constructed, that starts and ends at the depot, for that vehicle. A Restricted Candidate List (RCL) is built in order to include different customers in the route, always picking customers from the collection of non-visited customers in order to meet feasibility. The marginal cost to include some customer is found using the following expression:

$$incr = VariableCost \times t + overtime \times penalty + arrival,$$

being $arrival$ the arrival time at the new candidate. Observe that $incr$ represents an estimation for the marginal increase in the objective, since we need to adjust all the time-windows for the other customers. Nevertheless, the marginal costs are useful to build the RCL, following a classical implementation. We find the least and the greatest marginal costs i_{min} and i_{max}, and the RCL consists of the candidates e such that $incr(e) \leq i_{min} + \alpha \times (i_{max} - i_{min})$, being $\alpha \in [0, 1]$ the GRASP parameter that trades greediness for randomization. Finally, a random member belonging to the RCL is inserted into the partial route, and the whole collection of non-visited customers are updated accordingly, with a new evaluation of marginal costs. The route is closed whenever the vehicle capacity is reached, or when there are no more candidates to be included. In that case, the depot node is included.

4.2 Local Search Phase - VND

respective local searches are called in order, after the *Construction* phase:

1. $Fleet - opt$
2. *Exchange*
3. *Relocate*
4. $2 - opt$
5. $3 - opt$

We followed a strict time-complexity order of the local searches, as suggested in [12]. For practical reasons, we assume that there are more customers than vehicle-types.

Definition 2 (Fleet-Opt). *The goal is to change the vehicles. There are two different flavors of this local-search:*

- *Fleet-opt A: given two node-disjoint routes p and q associated to the respective vehicles v_p and v_q. We exchange the vehicles, such that v_q is associated to p and v_p is associated to q.*
- *Fleet-opt B: we can replace a given vehicle v_p associated to the route p by some different available vehicle v_d.*

Definition 3 (Exchange). *Consider two node-disjoint routed p and q that serve two distinct customers $i \in p$ and $j \in q$. We literally exchange the customers as follows. The edges $(i-1, i), (i, i+1) \in p$ are replaced by $(i-1, j), (j, i+1)$, and the edges $(j-1, j), (j, j+1)$ inq are replaced similarly, by $(j-1, i), (i, j+1)$. Figure 4 illustrates this local search.*

Definition 4 (Relocate). *Given two node-disjoint routes p and q, and some customer i that belongs to p. We relocate the customer i to the route q, as follows. First, replace the edges $(i-1,i)$ and $(i,i+1)$ by $(i-1,i+1)$, and then replace the edge $(j,j+1) \in q$ by the edges (j,i) and $i,j+1$. An illustration is presented in Figure 5.*

Definition 5 (2-opt). *Pick two non-adjacent edges $(i,i+1)$ and $(j,j+1)$ from a fixed tour of a feasible solution, such that $i < j$. Replace both links by (i,j) and $(i+1,j+1)$. Figure 6 illustrates this local search in a fixed tour.*

Definition 6 (3-opt). *Pick three non-adjacent edges (i,j), (k,l) and (m,n). We can either delete two, or the three edges. In the former, we replace as in 2-opt. In the latter, consider the four non-isomorphic reconstructions of the tour illustrated in Figure 7.*

The reader can appreciate that 3-opt is dominant, with cubic time-complexity in terms of the number of links.

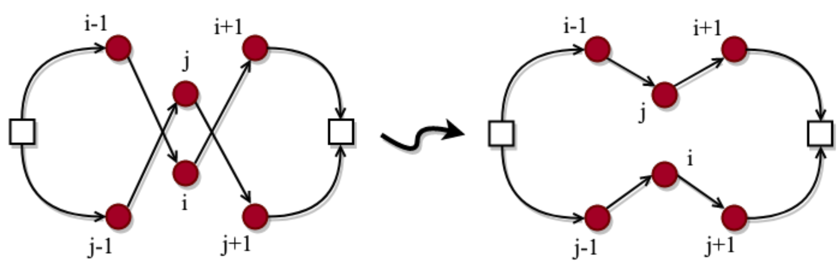

Fig. 4. Exchange

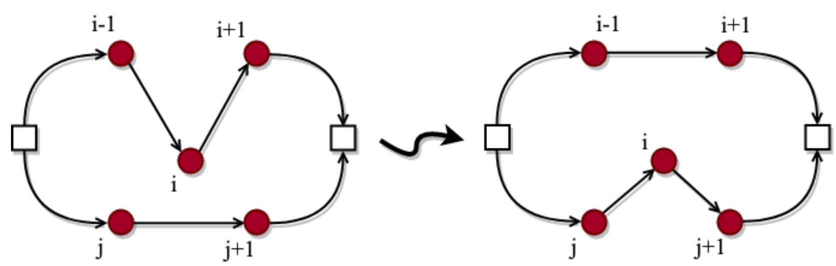

Fig. 5. Relocate

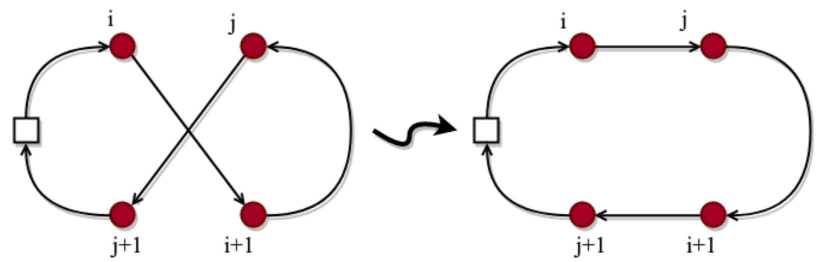

Fig. 6. 2-opt

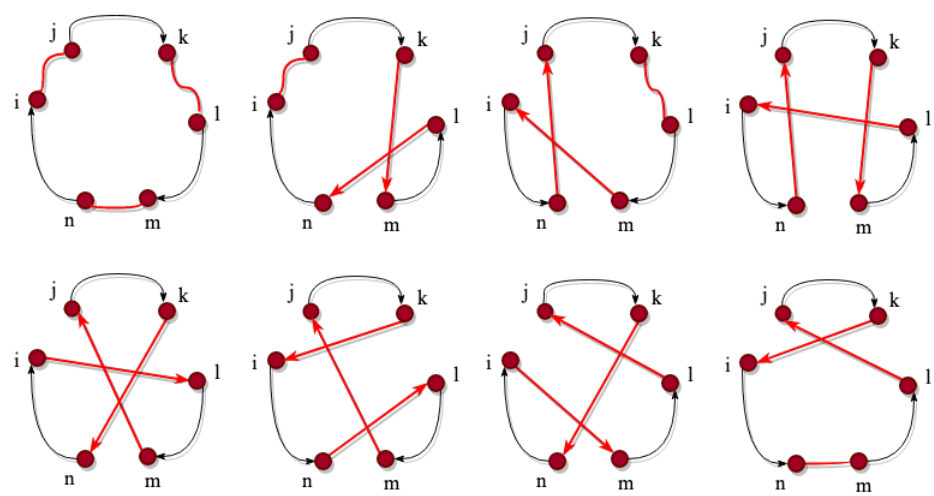

Fig. 7. 3-opt

5 Numerical Results

In order to understand the effectiveness of our proposal, an extensive computational study was carried out using our *Main* algorithm versus the exact CPLEX solver, with an halting time of 10 hours. Therefore, CPLEX returns either the globally optimum solution, or the best solution found so far after 10 hours. The experimental analysis was carried out in a Home-PC (Intel

Core i5 2.7 GHz). Since there are no benchmark for our specific problem we adapted Solomon instances [15], adding penalties and time-windows, using $\omega = 0.3$. This means that the time-window is enlarged a factor 1.3, but a penalty is assumed in the last portion of the window. Since the HFVRPTW with penalties in delays is novel, we cannot perform a fair comparison with previous proposals. Instead, we study the effectiveness of our methodology with respect to the exact CPLEX solver, and the activity of the different local searches. Table 1 shows the activity of the five local searches of our VND. Exchange and Relocate have the largest activity, followed by 2-opt and 3-opt. Fleet-opt has the least activity. However, Fleet-opt A has considerable activity as well in many instances under study. Further experiments show that Fleet-opt B has large activity when the number of customers is increased to 100 and 200.

Instance	***fleet-optA***	***fleet-optB***	***exch-ange***	***relo-cate***	***2-opt***	***3-opt***	***Instance***	***fleet-optA***	***fleet-optB***	***exch-ange***	***relo-cate***	***2-opt***	***3-opt***
HC101	0	0	123	96	0	0	HC201	0	0	28	26	12	0
HC102	0	0	128	93	42	20	HC202	0	0	45	46	28	24
HC103	0	0	90	66	41	25	HC203	0	0	92	102	50	50
HC104	0	0	162	118	75	34	HC204	0	0	64	57	40	28
HC105	0	0	98	74	26	0	HC205	0	0	36	40	29	18
HC106	0	0	53	42	18	0	HC206	0	0	45	51	36	24
HC107	0	0	107	77	43	0	HC207	0	0	40	38	25	19
HC108	0	0	67	54	28	4	HC208	0	0	39	40	30	20
HC109	0	0	138	99	64	19							
HR101	66	5	69	45	0	0	HR201	0	0	31	40	29	14
HR102	114	6	144	107	50	27	HR202	0	0	97	140	74	53
HR103	40	1	58	40	23	4	HR203	0	0	71	92	58	41
HR104	62	0	134	90	66	14	HR204	0	0	48	55	32	22
HR105	33	0	50	38	18	0	HR205	0	0	59	78	52	33
HR106	74	2	120	87	55	23	HR206	0	0	49	76	38	26
HR107	69	2	140	109	64	19	HR207	0	0	42	51	29	26
HR108	12	0	58	33	26	8	HR208	0	0	71	85	46	36
HR109	71	0	145	99	67	13	HR209	0	0	48	61	45	32
HR110	31	0	68	51	35	0	HR210	0	0	122	155	90	63
HR111	58	0	126	92	61	20	HR211	0	0	55	71	44	33
HR112	32	0	102	67	43	2							
HRC101	43	4	87	43	41	4	HRC201	0	0	53	37	27	6
HRC102	38	1	80	46	37	7	HRC202	0	0	86	57	40	9
HRC103	18	1	65	37	27	5	HRC203	2	1	158	122	74	27
HRC104	26	0	78	47	34	2	HRC204	0	0	185	132	76	19
HRC105	29	1	87	45	39	3	HRC205	0	0	7	58	36	12
HRC106	47	0	152	91	71	12	HRC206	0	0	170	122	86	17
HRC107	19	0	63	38	28	5	HRC207	0	0	62	44	29	15
HRC108	36	0	110	69	52	8	HRC208	0	0	98	69	49	9
Total	256	7	722	416	329	46	**Total**	2	1	819	641	417	114

Table 1. Activity of the different local searches (instances with 50 customers).

	CPLEX	GRASP/VND	Difference	
Instance	***Cost***	***Cost***	***Gap***	***Relative Error***
HC101	828.912	936.61	107.70	12.99%
HC102	871.274	887.47	16.20	1.86%
HC103	994.16	887.86	**-106.30**	**-10.69%**
HC104	901.258	886.28	**-14.98**	**-1.66%**
HC105	832.252	936.34	104.09	12.51%
HC106	805.756	937.79	132.03	16.39%
HC107	872.714	921.04	48.33	5.54%
HC108	903.65	936.91	33.26	3.68%
HC109	1036.794	883.53	**-153.26**	**-14.78%**
HC201	691.32	738.45	47.13	6.82%
HC202	645.58	750.25	104.67	16.21%
HC203	828.57	706.99	**-121.58**	**-14.67%**
HC204	724.75	674.55	**-50.20**	**-6.93%**
HC205	690.93	737.47	46.54	6.74%
HC206	825.22	696.31	**-128.91**	**-15.62%**
HC207	843.96	716.15	**-127.81**	**-15.14%**
HC208	772.99	719.22	**-53.77**	**-6.96%**
HR101	2475.672	2,577.31	101.63	4.11%
HR102	2678.248	2,488.57	**-189.67**	**-7.08%**
HR103	2674.69	2,450.45	**-224.24**	**-8.38%**
HR104	2463.58	2,285.83	**-177.75**	**-7.21%**
HR105	2629.696	2,517.14	**-112.56**	**-4.28%**
HR106	2781.796	2,431.01	**-350.78**	**-12.61%**
HR107	2578.342	2,338.32	**-240.02**	**-9.31%**
HR108	2503.142	2,321.05	**-182.09**	**-7.27%**
HR109	2484.816	2,401.86	**-82.95**	**-3.34%**
HR110	2658.084	2,359.41	**-298.68**	**-11.24%**
HR111	2496.894	2,341.67	**-155.22**	**-6.22%**
HR112	2406.202	2,334.26	**-71.94**	**-2.99%**
Average	-	-	**-155.37**	**-6.12%**

Table 2. CPLEX VS our GRASP/VND proposal (instances with 50 customers).

Tables 2 shows the performance of our proposal with respect to CPLEX for 50 customers. The bold instances present negative gaps; this means that our GRASP/VND proposal outperforms CPLEX and, naturally, CPLEX could not find the globally optimum solution during 10 hours. It is worth to remark that the average gap is negative, meaning that our proposal outperforms the solver. Furthermore, the CPU time of our GRASP/VND solution ranges between seconds and five minutes in the worst cases.

6 Conclusions and Trends for Future Work

Operational researchers are engaged with modeling variations of the celebrated Vehicle Routing Problem (VRP), given its paramount importance and diverse applications. Here we introduced a novel Heterogeneous Fleet VRP with Time Windows (HFVRPTW) version, with penalties due to overtime. The HFVRPTW belongs to the class of $\mathcal{NP}$-Hard problems, since it subsumes the Traveling Salesman Problem. This result promotes the development of approximative algorithms. A GRASP/VND methodology is here proposed, using five different local searches. Numerical results suggest that the most simple local searches have more activity; further experiments illustrate that fleet-opt local search works when the number of customers is increased. The exact solution show limited applicability, where the optimality is reached only under small-sized instances.

As future work, we want to introduce key concepts of the VRP and variations in real-life metropolitan transportation systems. Further, we would like to explore novel local searches and study different versions of VNS governed by probabilistic flow diagrams, governed by Markov chains.

7 Acknowledgements

This work is partially supported by Project ANII FCE_1_2019_1_156693 *Teoría y Construcción de Redes de Máxima Confiabilidad*, MATHAMSUD 19-MATH-03 *Rare events analysis in multi-component systems with dependent components* and STIC-AMSUD ACCON *Algorithms for the capacity crunch problem in optical networks.*

References

1. Roberto Baldacci, Enrico Bartolini, Aristide Mingozzi, and Roberto Roberti. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7(3):229–268, 2010.
2. Jose Bernal, John Willmer Escobar, and Rodrigo Linfati. A granular tabu search algorithm for a real case study of a vehicle routing problem with a heterogeneous fleet and time windows. *Journal of Industrial Engineering and Management*, 10(4):646, 2017.
3. Serap Ercan Cömert, Harun Reşit Yazgan, İrem Sertvuran, and Hanife Şengüi. A new approach for solution of vehicle routing problem with hard time window: an application in a supermarket chain. *Sādhanā*, 42(12):2067–2080, 2017.
4. author G. B. Dantzig and author J. H. Ramser. The truck dispatching problem. *Management Science*, page 80, 1959.
5. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.
6. Pierre Hansen. *Variable Neighborhood Search.* Springer International Publishing, Handbook of Heuristics, 2018.

7. Jun Jiang, Kien Ming Ng, Kim Leng Poh, and Kwong Meng Teo. Vehicle routing problem with a heterogeneous fleet and time windows. *Expert Systems with Applications*, 41(8):3748 – 3760, 2014.
8. Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.
9. A. Mor and M. G. Speranza. Duality in nonlinear programming: a simplified applications-oriented development. *4OR - A Quarterly Journal of Operations Research*, 18(2):129–149, 2020.
10. Yuichi Nagata, Olli Brysy, and Wout Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4):724 – 737, 2010.
11. Artur Pessoa, Ruslan Sadykov, and Eduardo Uchoa. Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. *European Journal of Operational Research*, 2018.
12. Petrică C. Pop, Levente Fuksz, and Andrei Horvat Marc. A variable neighborhood search approach for solving the generalized vehicle routing problem. In Marios Polycarpou, André C. P. L. F. de Carvalho, Jeng-Shyang Pan, Michał Woźniak, Héctor Quintian, and Emilio Corchado, editors, *Hybrid Artificial Intelligence Systems*, pages 13–24, Cham, 2014. Springer International Publishing.
13. Mauricio Resende and Celso Ribeiro. *Optimization by GRASP*. Springer, 2016.
14. Michael Schneider, Bastian Sand, and Andreas Stenger. A note on the time travel approach for handling time windows in vehicle routing problems. *Computers & Operations Research*, 40(10):2564 – 2568, 2013.
15. Marius M Solomon. On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks*, 16(2):161–174, 1986.
16. A. Taha, M. Hachimi, and A. Moudden. A discrete bat algorithm for the vehicle routing problem with time windows. In *2017 International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA)*, pages 65–70, April 2017.
17. Duygu Taş, Ola Jabali, and Tom Van Woensel. A vehicle routing problem with flexible time windows. *Computers & Operations Research*, 52:39–54, 2014.