OmniaScience

Journal of Industrial Engineering and Management
JIEM, 2017 – 10(4): 646-662 – Online ISSN: 2013-0953 – Print ISSN: 2013-8423
https://doi.org/10.3926/jiem.2159

A Granular Tabu Search Algorithm for a Real Case Study of a Vehicle Routing Problem with a Heterogeneous Fleet and Time Windows

Jose Bernal[1], John Willmer Escobar[2], Rodrigo Linfati[3]

[1]*Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle Cali (Colombia)*

[2] *Departamento de Contabilidad y Finanzas, Universidad del Valle Cali (Colombia)*

[3]*Departamento de Ingeniería Industrial, Universidad del Bío - Bío, (Chile)*

jose.bernal@correounivalle.edu.co, *john.wilmer.escobar@correounivalle.edu.co*, *rlinfati@ubiobio.cl*

Received: October 2016
Accepted: September 2017

Abstract:

Purpose: We consider a real case study of a vehicle routing problem with a heterogeneous fleet and time windows (HFVRPTW) for a franchise company bottling Coca-Cola products in Colombia. This study aims to determine the routes to be performed to fulfill the demand of the customers by using a heterogeneous fleet and considering soft time windows. The objective is to minimize the distance traveled by the performed routes.

Design/methodology/approach: We propose a two-phase heuristic algorithm. In the proposed approach, after an initial phase (first phase), a granular tabu search is applied during the improvement phase (second phase). Two additional procedures are considered to help that the algorithm could escape from local optimum, given that during a given number of iterations there has been no improvement.

Findings: Computational experiments on real instances show that the proposed algorithm is able to obtain high-quality solutions within a short computing time compared to the results found by the software that the company currently uses to plan the daily routes.

Originality/value: We propose a novel metaheuristic algorithm for solving a real routing problem by considering heterogeneous fleet and time windows. The efficiency of the proposed

approach has been tested on real instances, and the computational experiments shown its applicability and performance for solving NP-Hard Problems related with routing problems with similar characteristics. The proposed algorithm was able to improve some of the current solutions applied by the company by reducing the route length and the number of vehicles.

Keywords: vehicle routing problem, heterogeneous fleet, time windows, real case study

1. Introduction

Logistics Management belongs to the most important research areas of supply chain (Escobar, Linfati & Adarme-Jaimes, 2015). Companies need to establish operational and strategic decisions to optimize and efficiently manage the processes involved in their supply chains by ensuring that products are delivered with lower costs while maintaining high-quality service. These aspects guarantee not only excellent customer service but also storage and distribution savings.

Transportation and distribution of goods in an effective and efficient way are increasingly significant in both theoretical research and practical applications (Calvete, Carmen, Oliveros & Sánchez-Valverde, 2007). The problem of physical distribution of products is essential to consider because it represents a considerable proportion of the total logistic costs. These factors have rapidly increased in importance due to the continuously increasing complexity of businesses (Alvarenga & Mateus, 2004).

Some researchers suggest that transportation costs correspond to 10% to 15% of final commercialized goods (King & Mast, 1987). Thus, the logistics management of consumer product companies should focus on customer satisfaction, market needs, balance point, product rotation, logistics challenges, and coverage. The logistics operations involved in these processes can be described as combinatorial optimization problems (COP).

One of the most commonly addressed problems in COP is the Travelling Salesman Problem (TSP), in which an agent visits several cities and returns to their initial position with the goal of minimizing the distance traveled and visiting each city only once. The problem is usually represented by an undirected graph in which each node corresponds to a customer and each arc corresponds to the distance between them. Each node is weighted by the customer demand. Then, the minimum Hamiltonian cycle is determined by considering the traveled distance (Applegate, Bixby, Chvatal & Cook, 2007). This problem is considered NP-hard as it is computationally intractable when the number of nodes increases (Garey & Johnson, 1979; Fortnow, 2009).

The well-known vehicle routing problem (VRP) is an extension of the TSP (Toth & Vigo, 2001) in which the number of agents increases. In this case, the situation is rephrased as the problem of designing routes from one depot to a set of customers while satisfying the constraints of the demand and vehicle capacity. Several variants of this VRP have been studied within the literature (Golden, Assad, Levy & Gheysens, 1984; Baldacci & Mingozzi, 2008; Gendreau, Laporte, Musaraganyi & Taillard, 1999; Liu, Huang & Ma, 2009; Brando, 2011; Subramanian, Vaz-Penna, Uchoa & Ochi, 2012; Pérez & Guerrero, 2015). This paper considers a variant of a VRP that combines two classical problems of the COP area: VRP with a heterogeneous fleet (HVRP), which considers a fleet of vehicles with different capacities, and VRP with time windows (VRPTW), in which products can only be delivered to certain clients within defined time windows. In VRPTW, two types of time windows are considered: soft windows, where a penalty is considered if the constraint is not satisfied and hard windows, where the constraints must be satisfied. Note that hard windows are more difficult to address and are impractical in real life. Therefore, we consider soft time windows.

The problem addressed here is called the vehicle routing problem with a heterogeneous fleet and time windows (HFVRPTW). The aim of HFVRPTW is to determine the routes to be performed by minimizing the distance traveled. In addition, the following constraints must be satisfied: i) each route starts and finishes at same node (i.e. the depot); ii) the demand of the customers in a route must not surpass the capacity of the assigned vehicle; iii) each customer must be attended; iv) the time windows must be satisfied; and v) the length of the route must not exceed parameter D.

In this work, a two-stage approach for solving the HFVRPTW for a franchise company bottling Coca-Cola products in Colombia is proposed. In the first phase, a hybrid heuristic approach is performed to generate an initial feasible solution, while an approach based on a Granular Tabu Search (MGTS) is performed during the second phase.

This paper is organized as follows: The algorithm proposed to address the routing problem of the company is explained in Section 2. The efficiency of the proposed algorithm has been tested on real life scenarios that are described in Section 3, and the results are reported and analyzed in Section 4. Finally, remarks and future work of our research are detailed in Section 5.

2. Algorithm Description

The proposed approach considers the following phases:

a) Generation of the initial solution by using a heuristic algorithm and
b) Improvement of the initial solution by using an MGTS algorithm.

Both phases of the proposed algorithm are described in the following sections.

2.1. First Phase: Initial Solution

The initial solution is generated using a *customer-insertion algorithm*. The heuristic algorithm is presented in Algorithm 1 and is described as follows. Initially, the set of instances is prepared by sorting the vehicles and the customers with respect to their capacities and positions (distance to the depot), respectively. Then, the main part of the first phase takes place:

1. A two-node route is created: the depot is added twice to represent both the start point and the end point. The vehicles are initially assigned to the routes according to their order in the sorted list, i.e. the vehicle with maximum capacity is associated with the first route.

2. The algorithm inserts as many customers as possible while ensuring that the following parameters are met: the length of the route is less than or equal to the maximum specified value D; the demand of all customers belonging to a given route must not exceed the capacity of the assigned vehicle, and the time window constraints must be satisfied. Whenever a customer is inserted in the route, the route length and demand is updated.

The previous two steps are carried out iteratively until all the vehicles have been assigned to a route. Therefore, the algorithm is structured; it is possible that some customers are not selected at the end of the process. In that case, the problem constraints are relaxed by increasing the tolerance thresholds for each considered variable (i.e., maximum demand, maximum length, and time windows) and the algorithm is repeated once again.

Algorithm 1 First Phase: Customer-Insertion Algorithm

```
1:  procedure INSERTION(depot; customers; vehicles; constraints)
2:      vehicles ← sort(vehicles)
3:      customers ← sort(customers)
4:      routes ← {}
5:      while not all customer considered do
6:          for vehicle_i ∈ vehicles do
7:              Route route
8:              route.insert(depot)
9:              route.insert(depot)
10:             route.deliveringVehicle(vehicle_i)
11:             for j ← 1 to size(customers) do
12:                 if !customers[j].added then
13:                     if route.canInsert(customers[j]) then
14:                         route.insert(customers[j])
15:                         route.updateValues()
16:                         customers[j].added = true
17:             routes.insert(route)
18:         if not all customer considered then
19:             smooth(constraints)
20:             routes ← {}
21:             for j ← 1 to size(customers) do
22:                 customers[j].added = false
23:     return routes
```

2.1.1. Insertion Step

The insertion step consists mainly in verifying whether a customer can be added to the route under construction and, if that is the case, inserting it in the best position. Otherwise, if some of the constraints of the problem are not fulfilled by inserting the customer, that customer is not considered in the current route.

Let $r = \{v_0, v_1, \ldots, v_n\}$ be a route containing n customers, and let $\bar{v}$ be the next customer to insert in r. The best position to insert the customer is given by the following expression:

$$\underset{0<i<n}{argmin} \|v_i - \bar{v}\|_2 + \|\bar{v} - v_{i+1}\|_2 \quad (1)$$

Then, the first customer to insert in the route is always placed in the middle of the two initial nodes.

2.1.2. Exploiting Vehicle Capacity

As discussed previously, the vehicles are initially assigned with respect to their capacities to give more chances to the first routes to add more customers. However, this procedure not ensures that the capacity of the vehicle is exploited. Thus, every time a route is built, the algorithm explores whether there exists a

better correspondence between routes and vehicles. In case there is such option, the vehicles are reassigned.

2.2. Second Phase: Modified Granular Tabu Search

2.2.1. Granular Space

After the initial solution S_0 is obtained, an MGTS algorithm is applied. This algorithm uses the basic idea of granular search space introduced by Toth and Vigo (2003), which considers a sparse graph (uncompleted graph) instead of the complete graph for the VRP. The uncompleted graph contains all the edges connecting depots with customers, the edges forming the top feasible solutions during the search, and the edges from which the travelling cost is less than a granular threshold value $v = \beta \cdot z'$, where $z' = z\frac{S_0}{n+r}$ is the value of the average distance of the edges belonging to the feasible initial found. In particular, z is the objective function value of the initial solution S_0, n is the number of customers, r is the number of routes of S_0, and β is a sparsification factor that is dynamically updated according to the feasibility of the explored solutions. The modification of the value of β allows the algorithm to alternate between diversification stages (high values of β) and intensification stages (small values of β). The main goal of the granular tabu search is to find high-quality solutions while retaining the main characteristics of the original tabu search as well as short computing times. Successful algorithms based on the idea of granularity for solving different variations of the VRP have been proposed by Escobar (2014), Escobar, Linfati, Baldoquin and Toth (2014a, 2014b), Escobar, Linfati and Toth (2013), Linfati, Escobar and Gatica (2014) and Linfati, Escobar and Cuevas (2014).

We have proposed a classification of the edges of the complete graph considering two types: "big" or "small" edges. "Big" edges have values greater than the granular threshold v, while the values of "small" edges are less than v.

2.2.2. Neighborhood Structures

The MGTS algorithm considers infeasible solutions of the vehicle capacity, the total route length, and time windows, during the search. A feasible solution S is composed of a set of route $R = \{R_1, R_2, \ldots, R_r\}$. Each route $R_k \in R$ is denoted by $(v_0, v_1, \ldots, v_n)$ where $v_i \in V$ is a customer belonging to a route and $(v_i, v_j) \in R_k$ is an edge between two consecutive customers $v_i, v_j \in R_k$.

An objective function value is calculated for a solution Saccording to the following expression:

$$F_1(S) = \sum_{k=1}^{|R|} \sum_{(v_i,v_j)\in R_k} C_{v_i,v_j} + P_q(S) + P_{tw}(S) \quad (2)$$

where $P_q(S)$ is a penalty parameter calculated by multiplying the total fleet overload in the solution S by a penalty factor α_q, which is updated dynamically, and $P_{tw}(S)$ is a penalty term obtained by multiplying the total delay (in minutes) presented in S by a dynamically updated penalty factor α_{tw}. In particular, $\alpha_q = \rho_q \cdot F_1(S_0)$ and $\alpha_{tw} = \rho_{tw} \cdot F_1(S_0)$, where ρ_q and ρ_{tw} are parameters that are updated during the search. Note that if S is feasible, then all the penalty terms are equal to 0.

During the search, if the algorithm finds infeasible solutions of the capacity of the used vehicles for N_{fact} iterations, the value of ρ_q is set to $min(\rho_{max}, \delta_{inc} \cdot \rho_q)$,, where $\delta_{inc} > 1$. Otherwise, if any feasible solution is found for N_{fact} iterations, the value of ρ_q is set to $max(\rho_{min}, \delta_{dec} \cdot \rho_q)$, where $\delta_{dec} < 1$. A similar approach is used to calculate the value of ρ_{tw} during the search. The parameters N_{fact}, ρ_{min}, ρ_{max}, δ_{inc}, and δ_{dec} are given in advance.

We consider dynamic values of ρ_q and ρ_{qw} because these values can be specifically determined for each set of benchmark instances. In addition, dynamic values of these parameters allow a better penalization scheme. Indeed, one of the main differences between the proposed algorithm and that proposed in Toth and Vigo (2003), is the consideration of penalty parameters, which are updated dynamically during the search and are calculated respect to the value of the initial objective function S_0 instead of using penalty parameters set within fixed intervals. We have experimentally proved that this penalty scheme generally produces excellent solutions for determined benchmark sets within short computing times.

The proposed algorithm uses an intra-route and inter-route move called shift. This neighborhood allows removing a customer from its position on the current route, and inserting it either in a different position on the same route or in another route. This neighborhood was selected because it is easy to implement and can easily control the time window constraints with respect to different operators. In addition, shifts can reduce the number of routes for a given solution.

The shift exchange operator is evaluated if and only if the new edges to be inserted in the current solution belong to the granular space (sparse graph). Figure 1 shows the considered neighborhood.

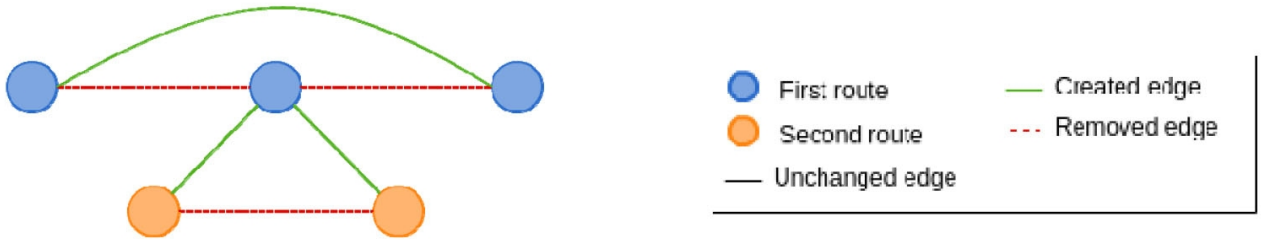

Figure 1. Shift exchange operator

2.2.3. Description of the Proposed Algorithm

This section discusses the proposed algorithm for the HFVRPTW. In particular, after constructing an initial solution S_0 (using the procedure described in Section 2.1), the MGTS algorithm is executed during several iterations through the selected neighborhood structure to improve the best feasible solution S^* found so far, until a given number of iterations is reached.

The algorithm initially sets $S^* = 'S = \hat{S} = S_0$, where $'S$ is the current solution (feasible or infeasible) and $\hat{S}$ is the current feasible solution. Then, the following steps are executed for IT_{max} iterations:

1. Apply the MGTS algorithm in the considered neighborhood N_k until a local minimum S' is found.
2. If S' is unfeasible and $F_1(S') \leq F_1('S)$, set $'S = S'$.
3. If S' is feasible and $F_1(S') \leq F_1(\hat{S})$, set $\hat{S} = S'$ and $'S = S'$.
4. If S' is feasible and $F_1('S) \leq F_1(S')$, set $'S = S'$.

Finally, the best feasible solution found so far S^* is kept. The algorithm moves, in each iteration, from a solution $\overline{S}$ to the best solution found in the neighborhood of $\overline{S}$, $N(\overline{S})$, even if it is unfeasible. Then, the selected move is set as *tabu* for *tabu tenure* iterations. The *tabu tenure* value is defined as a random integer within $[t_{min}, t_{max}]$, where t_{min} and t_{max} are given parameters. The algorithm for improving the initial solution is presented in Algorithm 2.

The algorithm considers a diversification strategy that is based on the idea of granularity (sparse graph) introduced by Toth and Vigo (2003). Initially, the sparcification factor β is set to a minimum value β_0. If $\overline{S}$ is unfeasible after N_{beta} iterations, the sparcification factor is changed to β_d. Then, a new sparse graph is generated considering only edges that are lower than the updated granular threshold value v_{beta_d}, and N_{change} iterations are performed from the best feasible solution found (S^*). Finally, if $\overline{S}$ is feasible after N_{change}

iterations, the sparcification factor is set back to its initial value β_0, and the search continues. The parameters β_0, β_d, N_{beta}, N_{change} are given in advance.

Algorithm 2 MGTS-Based Algorithm

```
1:  procedure MGTS(s_0, IT_max)
2:      s* ← s_0                                   ▷ the best current solution
3:      TL ← {}                                    ▷ the initial tabu list
4:      for i ← 1 to IT_max do
5:          s' ←' s
6:          z' ← ∞
7:          for all s ∈ N(s_{i-1}) do              ▷ Search in the neighbourhood
8:              if F_1(s) < z' ∧ s ∉ TL then
9:                  s' ← s
10:                 z' ← F_1(s)
11:         updated ← false
12:         if unfeasible(s') ∧ z' < F_1('s) then
13:             updated ← true
14:         if feasible(s') ∧ z' < F_1(ŝ) then
15:             ŝ ← s'
16:             updated ← true
17:         if feasible(s') ∧ z' = F_1('s) then
18:             updated ← true
19:         if updated then
20:             's ← s'
21:             TL ← TL ∪ {s_i}                    ▷ Update the tabu list
22:             update(TL, i)                      ▷ Delete the oldest moves
23:     s* ← ŝ
24:     return s*
```

2.2.4. Cost and Tabu List Verification

To select the solution with the minimum cost within the neighborhood, it is necessary to verify the following conditions:

- The cost of the move is expressed by the difference between the costs of created edges and the costs of deleted edges. Additionally, there could be a penalty associated with the move. The penalty becomes effective when one of the constraints is not fulfilled.
- If some of the implicated edges belong to the tabu list, the movement is not taken into account.
- The implicated edges are either valid or not according to the type of explored solutions (feasible or unfeasible). "Small" edges are explored if the solution is feasible. Otherwise, "big" edges are considered.

2.2.5. Additional Procedures

We have proposed two additional procedures (smoothing and restart) in order to help the algorithm to escape from a local minimum, when the algorithm has been executed during a certain number of iterations.

Because this situation is not desired, a procedure called smoothing, which extends the time window constraints performed by τ minutes after N_{reset} iterations have passed without improving the objective function. In the restart procedure, the current solution is set to the best solution found so far by the restart process after N_{reset} iterations. Both parameters were experimentally set to $\tau = 60$ minutes and $N_{reset} = 10\%$ of the maximum number of iterations.

3. Evaluation Elements

3.1. Test Datasets

A test set containing 12 real life scenarios of two cities in Colombia (Cali and Pereira) was considered to prove the efficiency of the proposed approach. The characteristics of the instances are summarized in Table 1 and are described as follows:

Case #	Information	Vehicles	Customers	Demands (CU)
1	Cali Monday	A, B, C	887	10781
2	Cali Tuesday	A, B, C	915	12205
3	Cali Wednesday	A, B, C	1022	14344
4	Cali Thursday	A, B, C	1097	13686
5	Cali Friday	A, B, C	966	11839
6	Cali Saturday	A, B, C	851	13348
7	Pereira Monday	B	783	7719
8	Pereira Tuesday	B	912	6545
9	Pereira Wednesday	B	993	7248
10	Pereira Thursday	B	872	7584
11	Pereira Friday	B	872	7059
12	Pereira Saturday	B	881	6408

Table 1. Characteristics of the case of study

- The demands of the customers (in cubic units) in Cali and Pereira are based on historical data from 2012, 2013 and 2014.
- The instances correspond to each day of the week from Monday to Saturday for Cali and Pereira.
- The number of vehicles k that are available vary depending on the day and the city. There are 16 vehicles on average with a minimum of 11 vehicles for Pereira on Tuesday, Friday and Saturday and a maximum of 23 vehicles for Cali on Wednesday and Thursday.
- Three types of vehicles with different capacities are considered: A, B and C have capacities of 832, 666 and 499 cubic units (CU), respectively.
- The number of customers varies depending on the day and the city, with a minimum of 683 for Pereira on Tuesday and a maximum of 842 for Cali on Thursday.
- Finally, 22% of the customers have time windows, and they represent 47% of the sales.

4. Results

The proposed algorithm was implemented in C++, and the computational tests were executed on an Intel Core 2 Quad (2.83 GHz) computer. Since the algorithm performance depends on the value for each of the mentioned parameters, a calibration process is carefully performed. This step is iteratively performed by considering each single variable and finding its value giving the highest objective function. The initial values of some of the parameters are obtained from previous works (Escobar, 2014; Escobar et al., 2014a; Bernal-Moyano, Escobar, Marín-Moreno, Linfati & Gatica, 2017). Table 2 displays the calibration of the parameter β_d in the refinement process after all the other variables were tuned up. The table shows the importance of setting the parameter β correctly. If the value is high, more arcs are considered (i.e. more options to explore in the refinement process) but, at the same time, the processing time increases. On the other hand, if the value for this parameter is low, the problem becomes more restrictive (i.e. less option to take into account while improving the routes). Hence, although there could be changes from the initial solution to the refined one, the improvement may not be significant. As seen in the table, higher improvement is observed when β is set to 2 or 2.5 than when it is set to 0.5 or 1.0. Although, all the cases were processed in less than two seconds, in problems with a larger number of nodes, the proper tuning of β establishing the trade-off between exploration and computational cost is essential. According to our experiments, the following sets of parameters were set for all the instances: N_{fact} =1.0, ρ_{min} = 1.0, ρ_{min} = 10.0, $\delta_{dec} = \frac{1}{1.1}$, t_{min} = 7, t_{max} = 49, β_0 = 1.0, β_d = 2.5, N_{beta} = 1, $N_{change} = 0.1 \cdot n$ and $IT_{max} = 3 \cdot n$.

Case #	β_d	Initial solution		MGTS solution	
		Cost	Time (ms)	Cost	Time (ms)
1	0.5	349.80	0.53	286.97	150.96
	1.0		0.49	270.43	202.85
	1.5		0.50	259.85	323.80
	2.0		0.49	255.59	393.78
	2.5		0.50	**245.37**	373.12
2	0.5	409.56	0.55	386.34	42.29
	1.0		0.53	376.13	68.54
	1.5		0.53	370.41	95.90
	2.0		0.53	368.50	122.53
	2.5		0.53	**366.40**	154.83
3	0.5	539.18	0.60	410.10	283.28
	1.0		0.57	375.43	610.74
	1.5		0.57	356.38	636.39
	2.0		0.56	346.50	726.72
	2.5		0.48	**343.44**	1196.83
4	0.5	602.10	0.96	567.77	103.45
	1.0		0.92	557.25	180.52
	1.5		0.92	539.13	263.48
	2.0		0.92	**537.92**	314.39
	2.5		0.92	551.08	395.48
5	0.5	349.80	0.50	286.97	149.86
	1.0		0.49	270.43	201.70
	1.5		0.49	259.85	320.82
	2.0		0.49	255.59	370.11
	2.5		0.40	**245.37**	324.28
6	0.5	452.97	0.76	430.72	572.00
	1.0		0.77	411.88	644.06
	1.5		0.55	413.95	776.44
	2.0		0.53	**405.78**	647.03
	2.5		0.53	409.40	517.90

Table 2. Results obtained with different values of β in the refinement process

After defining the parameters of the algorithm, the evaluation was conducted considering the elements presented in the previous section. The current and real solutions as well as the solution obtained using the proposed algorithm are presented in Table 3. Note that when the MGTS solution GAP is equal to 0%, the proposed algorithm outperforms the results found by the company.

Case #	BKS	Real solution			Initial solution			MGTS solution		
		k	Cost	GAP	*k*	Cost	GAP	*k*	Cost	GAP
1	219.51	11	241.62	10.10%	12	295.05	34.41%	12	219.51	00.00%
2	196.80	11	316.18	60.66%	11	294.27	49.53%	11	196.80	00.00%
3	255.00	12	255.00	00.00%	16	420.47	64.89%	16	292.63	14.76%
4	242.91	13	353.37	45.47%	15	347.66	43.12%	15	242.91	00.00%
5	206.45	12	252.47	22.29%	14	332.66	61.19%	14	206.45	00.00%
6	259.13	12	259.13	00.00%	13	397.69	53.47%	13	274.74	06.02%
7	246.51	9	246.51	00.00%	8	392.00	59.02%	8	336.48	36.50%
8	236.22	9	236.22	00.00%	7	307.81	30.31%	7	258.18	09.30%
9	351.90	10	351.90	00.00%	10	449.19	27.65%	10	408.41	16.06%
10	291.04	11	291.04	00.00%	10	470.71	61.73%	10	425.68	46.20%
11	262.54	10	262.54	00.00%	9	370.77	41.22%	9	323.13	23.08%
12	352.53	10	357.36	01.37%	9	396.99	12.61%	9	352.13	00.00%

Table 3. Comparison between the current solution applied by the company and the proposed approach

The table notation used in the table is described as follows: Case # indicates the selected instance, Cost is the objective function value of the real, initial solution (first phase) or the MGTS algorithm (second phase), *k* is the number of vehicles considered in the solution, BKS is the best known solution value of the objective function with respect to the solution found by the company or by the proposed algorithm, and GAP refers to the percentage gap between the solution cost and the best one.

In the study cases in Cali (cases 1-6), the MGTS algorithm significantly improves the results obtained by the first phase (first procedure), achieving similar or better objective function values than the current solution applied by the company. However, it can be seen that the number of vehicles is similar or larger than the ones used in the real solution. Since our proposal is indeed oriented to satisfy time window constraints, this situation could mean that more vehicles are needed to fulfill this requirement and, hence, obtain a higher satisfaction index from the customer's side. It is also important to note that, these vehicles will travel less distance than in the real case as pointed out by the cost of the MGTS solution. This could mean that some of the associated costs (e.g. as fuel) are reduced.

In the study cases in Pereira (cases 7-12), the situation is the opposite of the Cali. The MGTS algorithm is not able to outperform the current solution applied by the company. Unlike in the case of Cali, the second procedure was not able to enhance considerably the initial solution. This could mean that insertion exchange operator was not enough to escape from local minima around the starting point. We believe this issue could be tackled by implementing other exchange operators (e.g. swap, double swap, double insertion and 2-opt) within the refinement step. Another form to deal with this situation is to

implement a procedure, which randomly modifies the current solution (usually referred as shaking procedure). Nevertheless, the number of vehicles considered by the MGTS algorithm is larger than in the real case. However, the solutions found by our proposal are beneficial for the company since i) all the routes are feasible according to our problem formulation and ii) the solution finds a solution contemplating less number of vehicles than in the current case while satisfying customers demand. Note that the second element also indicates that total logistics cost and other associated costs (for instance, vehicle crew, fuel, insurance, and maintenance) could be reduced.

In addition, the occupation of the solution provided by the meta-heuristic approach is approximately 83% (lower than the current occupation), and also the level service is satisfied considering the time windows of the customers. Considering that a high percentage of the sales are from clients with time windows, this situation could increase the satisfaction index, increase the sales volume and brand confidence and improve the operations related to product distribution. Analysis of the impact of the satisfaction factor on the company should be carried out along with cost analysis on the different solutions to decide which option is more suitable for the company.

5. Final Remarks

A two-stage heuristic was proposed to solve a real HFVRPTW. The proposed approach was compared to the current results found by the software used by the company to plan the performed routes.

The proposed approach initially found a solution using a customer-insertion algorithm (first phase). Then, the initial solution was improved by a modified GTS, for which the main idea is to improve the performance with HFVRPTW instances. Additional procedures are proposed to help the proposed approach to escape from local optimum, when a given number of iterations with no improvements have been reached. According to our experience, the algorithm described in this paper can be equipped afterwards with i) more exchange operators to help in exploring the search space and, perhaps, reducing the objective function; and ii) shaking procedures to escape local minima. Both approaches should be carefully implemented to comply with the time window constraints and are considered for future work.

The proposed algorithm was able to improve some of the current solutions applied by the company by reducing the route length and the number of vehicles. Although in some cases the results were not as good as the current solution, it is notable that the time windows were contemplated in all the cases. This fact could imply an increased satisfaction index and sales volume, which are cornerstones for the company. Also, the results showed that, under the considered configurations and problem representation, there is room for improvement in the company solutions for the case of study.

The authors believe that the future work on real HFVRPTW should be directed in four ways. Firstly, the impact on the satisfaction index should be considered to determine the tradeoff between duration and customer service. Secondly, more factors that are part of the real scenario (such as fleet costs, uncertainty in arrival and service times, and product variability) should be contemplated since they could improve the problem representation and, hence, a realistic comparison can be performed. Thirdly, a having a public dataset for the HFVRPTW should be released to evaluate the performance of different proposed algorithms under the same conditions. Fourthly, cost analysis should be carried out to establish which option is favorable to the company. We believe that the analysis should take into account not only logistic costs and associated costs of the problem, but also the negative impact that losing clients may have for the brand.

Acknowledgement

This work has been partially supported by Universidad del Valle from Colombia and by Universidad del Bío-Bío from Chile. In addition, R. Linfati acknowledges the funds received by project CONICYT FONDECYT 11150370 and the "Grupo de Logística y Transporte" at the Universidad del Bío-Bío. This support is gratefully acknowledged.

References

Alvarenga, G.B., & Mateus G.R. (2004). A two-phase genetic and set partitioning approach for the vehicle routing problem with time windows. In *Hybrid Intelligent Systems, 2004.HIS'04. Fourth International Conference on.* 428-433. https://doi.org/10.1109/ICHIS.2004.13

Applegate, D.L., Bixby, R.E., Chvatal, V., & Cook, W.J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press.

Baldacci, R., & Mingozzi, A. (2008). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming,* 120(2), 347-380. https://doi.org/10.1007/s10107-008-0218-9

Bernal-Moyano, J.A., Escobar, J.W., Marín-Moreno, C., Linfati, R. & Gatica, G. (2017). A comparison of trajectory granular based algorithms for the location-routing problem with heterogeneous fleet (LRPH) *DYNA,* 84(200), 193-201. https://doi.org/10.15446/dyna.v84n200.55533

Brando, J. (2011). A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers & Operations Research* 38(1), 140-151. https://doi.org/10.1016/j.cor.2010.04.008

Calvete, H.I., Carmen, G., Oliveros, M., & Sánchez-Valverde, B. (2007). A goal programming approach to vehicle routing problems with soft time windows. *European Journal of Operational Research* 177(3), 1720-1733. http://www.sciencedirect.com/science/article/pii/S0377221705006508 https://doi.org/10.1016/j.ejor.2005.10.010

Escobar, J.W. (2014). Heuristic algorithms for the capacitated location-routing problem and the multi-depot vehicle routing problem. *4OR,* 12(1), 99. https://doi.org/10.1007/s10288-013-0241-4

Escobar, J.W., Linfati, R., & Adarme-Jaimes, W. (2015). A hybrid meta-heuristic algorithm for the capacitated location routing problem. *DYNA,* 82(189), 243-251. https://doi.org/10.15446/dyna.v82n189.48552

Escobar, J.W., Linfati, R., Baldoquin, M.G., & Toth, P. (2014a). A Granular Variable Tabu Neighborhood Search for the capacitated location-routing problem. *Transportation Research Part B: Methodological,* 67, 344-356. https://doi.org/10.1016/j.trb.2014.05.014

Escobar, J.W., Linfati R., & Toth, P. (2013). A two-phase hybrid heuristic algorithm for the capacitated location-routing problem. *Computers & Operations Research,* 40(1), 70-79. https://doi.org/10.1016/j.cor.2012.05.008

Escobar, J.W., Linfati R., Toth, P., & Baldoquin, M.G. (2014b). A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. *Journal of Heuristics,* 20(5), 483-509. https://doi.org/10.1007/s10732-014-9247-0

Fortnow, L. (2009). The Status of the P Versus NP Problem. *Commun ACM,* 52(9), 78-86. https://doi.org/10.1145/1562164.1562186

Garey, M.R., & Jhonson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W.H. Freeman & Co.

Gendreau, M., Laporte, G., Musaraganyi, C., & Taillard, E.D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research,* 26(12), 1153-1173. http://www.sciencedirect.com/science/article/pii/S0305054898001002 https://doi.org/10.1016/S0305-0548(98)00100-2

Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research,* 11(1), 49-66. http://www.sciencedirect.com/science/article/pii/0305054884900078 https://doi.org/10.1016/0305-0548(84)90007-8

King, G.F., & Mast, T.M. (1987). Excess travel: causes, extent, and consequences. *Transportation Research Board,* 1111, 126-134.

Linfati, R., Escobar, J.W., & Cuevas, B. (2014). An algorithm based on granular tabu search for the problem of balancing public bikes by using multiple vehicles. *DYNA,* 81(186), 284-294. https://doi.org/10.15446/dyna.v81n186.45220

Linfati, R., Escobar, J.W., & Gatica, G. (2014). A Metaheuristic Algorithm for the Location Routing Problem with Heterogeneous Fleet. *Ingeniería y Ciencia,* 10(19), 55-76. https://doi.org/10.17230/ingciencia.10.19.3

Liu, S., Huang, W., & Ma, H. (2009). An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review,* 45(3), 434-445. http://www.sciencedirect.com/science/article/pii/S1366554508001324 https://doi.org/10.1016/j.tre.2008.10.003

Pérez, E., & Guerrero, W. (2015). Métodos de optimización para el problema de ruteo de vehículos con inventarios y ventanas de tiempo duras. *Revista Ingeniería Industrial,* 14(3), 31-49.

Subramanian, A., Vaz-Penna, P.H., Uchoa, E., & Ochi, L.S. (2012). A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem. *European Journal of Operational Research,* 221(2), 285-295. http://www.sciencedirect.com/science/article/pii/S0377221712002093 https://doi.org/10.1016/j.ejor.2012.03.016

Toth, P., & Vigo, D. (2001). *The Vehicle Routing Problem.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Toth, P., & Vigo, D. (2003). The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *Informs Journal On Computing,* 15(4), 333-346. https://doi.org/10.1287/ijoc.15.4.333.24890