Comparing Two Heuristic Local Search Algorithms for a Complex Routing Problem

Pablo CABRERA-GUERRERO[1*], Andrés MOLTEDO-PERFETTI[2], Enrique CABRERA[3], Fernando PAREDES[4]

[1] Pontificia Universidad Católica de Valparaíso,
Av. Brasil 2241, Valparaíso, 2362807, Chile,
pablo.cabrera@pucv.cl (**Corresponding author)*

[2] Escuela de Psicología, Universidad Santo Tomás,
Los Limonares 190, Viña del Mar, 2561694, Chile,
andresmoltedo@santotomas.cl

[3] CINFAV, Universidad de Valparaíso,
Blanco 951, Valparaíso, 2362905, Chile,
enrique.cabrera@uv.cl

[4] Escuela de Ingeniería Industrial, Universidad Diego Portales,
Manuel Rodríguez Sur 415, Santiago, 8370109, Chile,
fernando.paredes@udp.cl

Abstract: Vehicle routing problems (VRP) have been widely studied in literature. Heuristics as well as exact algorithms have been applied to solve this kind of problems. In this study we approximately solve the VRP with simultaneous pickup and delivery and time windows by means of two well-known heuristics namely Tabu Search and Simulated Annealing. We compare the obtained results and then propose a restoration technique that allows both Tabu Search and Simulated Annealing to better explore the solution space. Results show that the proposed restoration technique allows both heuristic algorithms to obtain better results.

Keywords: Tabu Search, Simulated Annealing, Reverse Logistic, Restoration Techniques.

1. Introduction

Heuristic algorithms have been applied on a variety of problems in operations research and logistics for more than 50 years now. They have been shown to be very effective in dealing with complex problems that cannot be solved to optimality by traditional techniques such as mathematical programming. This is because mathematical programming methods often fail as the optimisation problem gets larger (i.e. more decision variables are involved).

Two well-known heuristic local search algorithms that have been applied on this kind of complex optimisation problems are Tabu Search [15] and Simulated Annealing [20, 21, 27]. In this paper we study the performance of these two heuristic algorithms when solving a complex problem arising in logistics called vehicle routing problem (VRP) with simultaneous pickup and delivery and time windows (VRPSPDTW).

Tabu Search algorithm has been used to approximately solve a range of combinatorial optimisation problems [5, 10, 23]. It has been shown to be a very simple, yet effective, method to approximately solve large and complex combinatorial optimisation problems such as the one we address in this paper. Just as Tabu Search, Simulated Annealing has also been considered to approximately solve a variety of optimisation problems (see for instance [3, 17, 24, 36]).

In this paper we implement both Tabu Search and Simulated Annealing algorithms and apply them on the VRPSPDTW. We then compare the results obtained by each technique.

One difficult we face when solving routing problems by means of local search algorithms is that neighbours of the current solution are often not feasible. Thus, we need to restore such neighbours so they become feasible. In this paper we propose a generic restoration strategy in the aim of making both local search algorithms to perform more efficiently. It is important to note that the restoration strategy we propose in this paper is directly applied on the local search algorithms. Thus, it can be seen as a generic strategy that can be included in any local search algorithm other than Tabu Search and Simulated Annealing.

This paper is organised as follows. In next section both Tabu Search and Simulated Annealing algorithms are described and their main features are highlighted. In Section 3 the VRPSPDTW problem we address in this paper is introduced. The mathematical formulation for this problem is presented at the end of this section. In Section 4 we introduce the restoration technique that is used within both heuristic algorithms. In Section 5 computational experiments performed in this paper are presented. In this section we discuss how the restoration technique helps both local search algorithms to better explore the solution space. Finally, in Section 6 some conclusions are drawn and future work is outlined.

2. Local Search Algorithms

2.1 Simulated annealing

Simulated Annealing is a local search heuristic algorithm that is inspired by thermodynamic systems. The algorithm takes concepts such as *energy*, *state* and *temperature* from thermodynamic and adapts them to fit within optimisation framework. Simulated Annealing needs the following parameters to work: the initial temperature t_0, the maximum number of iterations the algorithm can perform, *maxIter* and the parameter α, which is part of the annealing schedule. The SA algorithm begins with a solution $s_{current}$, also called *state*, that can be either randomly generated or user-provided. After that, a neighbour solution $s_k \in \Re(s_{current})$ is generated, where $k < maxIter$ denotes the current iteration and $\Re(\cdot)$ returns the neighbourhood of a solution. Once a neighbour is chosen from the neighbourhood, we compute its cost and observe the change in objective function values when moving from $s_{current}$ to s_k, $\Delta E = cost(s_k) - cost(s_{current})$. If $\Delta E < 0$ then the neighbour s_k is set as the new current solution, $s_{current}$. If $\Delta E \geq 0$ then the neighbour solution s_k is accepted with a probability

$$P(\Delta E) = e^{\frac{-\Delta E}{t_k}} \quad (1)$$

It is clear that acceptance probability $P(\Delta E)$ depends on the temperature parameter t_k which usually varies over the algorithm execution. As the *temp* variable *cools down*, worst solutions are no longer accepted, which provokes that the algorithm converges to a locally optimal solution. The variable *temp* cools down according to an *annealing schedule*. In this paper we use the following annealing schedule

$$temp_k = \alpha temp_{k-1} \quad (2)$$

with α in the range $[0,1]$. Other annealing schedules can be found in [20, 26, 38]. The algorithm ends when either no further improvements can be made or the maximum number of iterations *maxIter* is reached. Algorithm 1 shows the Simulated Annealing algorithm that is implemented in this paper.

```
Algorithm 1: Simulated Annealing
Input: maxIter , t0 , tmax ;
Output: sbest
 1  begin
 2    scurrent = initSol();
 3    sbest = scurrent ;
 4    k = 0 ;
 5   while k < maxIter do
 6     sk = selectSol( ℜ(scurrent) );
 7     ΔE = cost(sk) − cost(scurrent) ;
 8    if ΔE < 0 then
 9      scurrent = sk ;
10     if cost(sk) < cost(sbest) then
11       sbest = sk ;
12     else
13       tk = calcTemp( k , t , t0 , α );
14       P(ΔE) = e^(−ΔE/tk) ;
16      if P(ΔE) > rand() then
17        scurrent = sk ;
18       End
19      end
20      k = k + 1 ;
21    End
22   end
23  end
```

2.2 Tabu search

Tabu Search is a heuristic local search algorithm that implements adaptive memory structures [15]. Since Tabu Search algorithm is a local search, neighbourhood movements must be performed to explore the search space. Also, Tabu Search needs a set of parameters to work. The *divThreshold* parameter is the diversification threshold. This parameter allows Tabu Search to move out from regions where solutions' quality is low and "jump" to new regions where high quality solutions are expected to be found. In this paper a restart method is used to diversify the search. This

restart method keeps the best solution found by the algorithm before the restart. The *tabuSize* parameter is the length of the list of movements that are banned by the algorithm that is called *tabu list*. The tabu list, is a short-term memory where the last *tabuSize* neighbourhood moves done by the algorithm are stored. Once the tabu list is full, every time a new movement is added to the list, another movement is removed from the list, so the number of movements labelled as tabu is always equal to *tabuSize* . Although the tabu list is very useful to avoid getting trapped into a loop of movements, some strategies must be considered to avoid missing good quality solutions. One strategy Tabu Search implements is the aspiration criterion. The aspiration criterion allows that a movement from the tabu list can be considered only if the objective function value of the resulting solution is better than the best solution value found so far by the Tabu Search algorithm. The aspiration criterion ensures that no good quality solutions are missed because of applying the tabu list. Further, Tabu Search implements a frequency list which tracks how often a neighbourhood move has been used. Knowing which movements have been applied in previous iterations helps the algorithm to move into not-well-explored regions of the search space.

The TS algorithm begins with a solution $s_{current}$ that can be either randomly generated or user-provided. This initial solution $s_{current}$ is set as the best solution (s_{best}). A list of candidate solutions from the neighbourhood of $s_{current}$, $\Re(s_{current})$, is then generated. Size of the list of candidates is equal to *listOfCandidates* parameter. The best solution from $\Re(s_{current})$, is then selected and set as the new $s_{current}$. If the new $s_{current}$ is in the tabu list, then it will not be considered unless its cost is less than the cost of s_{best} (aspiration criterion). Each time $s_{current}$ is updated, the associated movement is added to the tabu list and the movement that has been in the tabu list for *tabuSize* iterations is removed from the list. If $cost(s_{current}) < cost(s_{best})$ then s_{best} is updated and the *noImprovement* counter is reset. Otherwise, *noImprovement* counter is updated. The Tabu Search algorithm ends after the number of iterations is equal to *maxIter* . Algorithm 2 presents the Tabu Search algorithm that is implemented in this paper.

```
Algorithm 2: Tabu Search
Input: maxIter , divLim , tabuSize ;
Output: s_best
Begin
 k=0;
 s_current =initSol();
 s_best = s_current ;
 while k < maxIter do
  s_k = best(ℜ(s_current)) ;
  while s_k ∈ tabuList do
   if cost(s_k) < cost(s_best) then
    remove s_k from tabuList ;
   Else
    remove s_k from ℜ(s_current) ;
    s_k = best(ℜ(s_current)) ;
   end
  end
  updateTabuList(s_current, s_k, tabuSize) ;
  s_current = s_k ;
  k = k + 1 ;
  if cost(s_current) < cost(s_best) then
   s_best = s_current ;
   noImprovement = 0 ;
  end
  noImprovement = noImprovement + 1 ;
  if (noImprovement > divLim) then
    s_current =initSol();
  end
 End
End
```

3. The VRPSPDTW

In this section we first present a brief overview on the VRPSPDTW problem we address in this paper. Then, the mathematical model of the VRPSPDTW is introduced.

Many research articles addressing different vehicle routing problems can be found in the literature (see [6] for a recent survey on VRP). Since the first academic paper addressing the VRP in late 50's [11], many optimisation problems based on the simplest version of the VRP have been proposed [32]. One of these problems is the VRPSPD. The VRPSPD was firstly introduced by Min in [28]. Since then, many authors have proposed different strategies to (approximately) solve the VRPSPD problem [1, 4, 8, 9, 13, 22, 31]. Tabu search as well as simulated annealing have also been considered to solve the VRPSPD [35, 39].

One optimisation problem that result of adding time windows constraints to the VRPSPD

problem is the VRPSPDTW we address in this paper. Although less studied than the VRPSPD problem, this problem is very important in reverse logistics. Swarm intelligence [14, 16, 18] as well as local search algorithms [19, 25, 29, 37] have been used to approximately solve this problem. Exact algorithms have also been presented to solve the VRPSPDTW. For instance, a branch-and-price algorithm is presented in [2]. The authors claim that this is the first exact algorithm to solve the VRPSPDTW. Only small instances can be solved using this technique though.

Authors in [12], [30] and [35] proposed three different models for the VRPSDP. Particularly, the author in [12] modelled the VRPSPD as part of the reverse logistics process. In this paper we extend the VRPSPD model presented in [12] and present a model for the VRPSPDTW we solve in this paper.

Table 1 shows the parameters we consider in this paper.

Table 1. Parameters of the VRPSPDTW

Parameter					**Value**		
Max number of vehicles (m)					10;25		
Vehicles capacity (L)					[100,350]		
Number of customers (n)					25;50;100		
v	x	y	d	p	t_{min}	t_{max}	t_s
0	x_0	y_0	--	--	--	--	--
1	x_1	y_1	d_1	p_1	t^1_{min}	t^1_{max}	t^1_s
2	x_2	y_2	d_2	p_2	t^2_{min}	t^2_{max}	t^2_s
...	...	...	...	...			
N	x_n	y_n	d_n	p_n	t^n_{min}	t^n_{max}	t^n_s

In the VRPSPDTW a set of customers C must be served for a fleet of vehicles we denote by V. Each customer $i \in C$ has its own delivery and pick-up demands, denoted by d_i and p_i, respectively, with $i = 1,\ldots,n$; indexing customers in C. We assume that both, delivery and pick-up demands are served by the same vehicle $v = 1,\ldots,m$ and at the same time. A vehicle $v \in V$ can serve one or more customers and it always starts and ends in a central depot we call O. Thus, the problem consists on finding a set of routes for the vehicles in the fleet that minimises the total distance covered by the vehicles while ensuring that all customers' demands are served considering time windows constraints. Routes are represented by a binary decision variable x_{ijv} such that $x_{ijv} = 1$ if the customer j is visited immediately after customer i by vehicle v; $x_{ijv} = 0$ otherwise.

As mentioned before, in this paper we also consider time windows. Time windows constraints ensure that the service for customer i starts within a pre-defined period of time $[t^i_{min}, t^i_{max}]$. Each service (delivery + pick up) takes t^i_s time units and it does not depend on the vehicle that provides the service. We assume that a service can finish after t^i_{max}.
Another distinctive feature of the VRPSPDTW is that we have to make sure that the vehicle capacity is never violated during the entire route. To do this, we need, first, to define the initial load a vehicle v has after leaving the depot. This initial load is defined as follows

$$l^v_0 = \sum_{i=0}^{n} \sum_{j=1}^{n} x_{ijv} d_j \tag{3}$$

We then define the vehicle load after visiting a customer in its route as

$$l^v_j \geq \begin{cases} l^v_0 - d_j + p_j \; \forall j \in C, v \in V; if \; x_{0jv} = 1 \\ l^v_i - d_j + p_j \; \forall i, j \in C, v \in V; if \; x_{ijv} = 1 \end{cases} \tag{4}$$

Equation (4) says that *in route* load of a vehicle $v \in V$ is equal to the load of the vehicle after serving the previous customer i (or after leaving the depot in case customer j is the first customer in the route of vehicle v) minus the items that are left to customer j (d_j) plus the items that are picked up from the same customer (p_j). It is clear that if

$$l^v_j > L \tag{5}$$

the route becomes infeasible. To ensure that each customer is served exactly once, we make

$$\sum_{i=0}^{n} \sum_{v=1}^{m} x_{ijv} = 1 \quad \forall j \in C \tag{6}$$

Also, we need to make sure that a customer is served by the same vehicle.

$$\sum_{i=0}^{n} x_{ihv} = \sum_{j=0}^{n} x_{hjv} \quad \forall h \in C, v \in V \tag{7}$$

Moreover, we need to make sure all vehicles start from the depot

$$\sum_{j=1}^{n} x_{0jv} = 1 \quad \forall v \in V \tag{8}$$

Regarding time windows, three constraints are considered in this paper

$$st^j \geq \begin{cases} st^i + t_s^i + T_{ij} \; \forall \, i, j \in C, v \in V \; ; if \; x_{ijv} = 1 \\ 0 \qquad Otherwise \end{cases} \tag{9}$$

$$st^j \leq t_{max}^j \; \forall \; j \in C \tag{10}$$

$$st^j \geq t_{min}^j \; \forall \; j \in C, \tag{11}$$

where st^j is the starting time for serving customer j, and T_{ij} is the travel time between customers i and j. Equation (9) ensures that there is enough time to serve a customer i and to travel to the next customer j before serving it. Equation (10) ensures that service of customer j will not start after t_{max}^j. Similarly, Equation (11) ensures that service of customer j will not begin before t_{min}^j.

The VRPSPDTW problem we solve in this paper is then

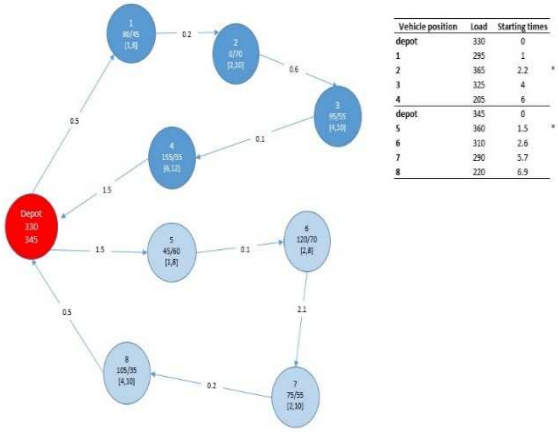
$$min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{v=1}^{m} D_{ij} x_{ijv} \tag{12}$$

subject to Equations (3) to (11).

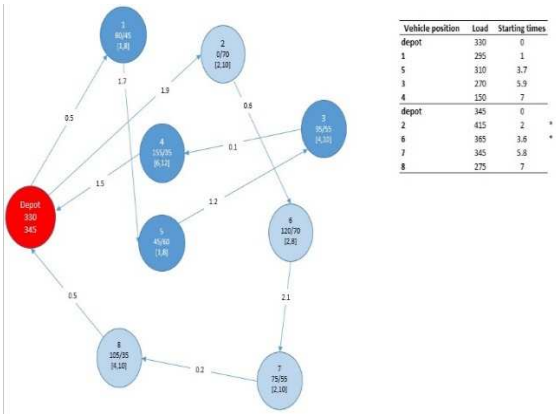
As mentioned in the introductory section of this paper, two local search algorithms are used to solve the VRPSPDTW, namely Tabu Search and Simulated Annealing. One issue that local search algorithms face when solving this problem is that resulting neighbours are not always feasible making the search in the solutions space less effective. For this reason, in this paper we propose a restoration technique that helps to fix neighbours that are not feasible. We explain the restoration technique in the next section.

4. Restoration Technique

After all customers are assigned to a vehicle, we need to make sure that feasible routes can be constructed using the current assignation. Finding the optimal route for each vehicle is a simple task that can be done by using mathematical programming methods, as long as not many customers are involved. In spite of that, we observe that routes in the neighbourhood of a feasible route are, sometimes, not feasible. Thus, we can either look for a feasible neighbour or repair the infeasible neighbour. In this paper we choose to repair those neighbours that are not feasible. The restoration phase we propose in this paper consists on a sequence of 2-opt movements between those customers that violate time windows constraints and/or load capacity constraints. We exchange those conflicting customers among infeasible routes until a feasible route is found. Figure 1 shows an example the restoration phase proposed here.



(a) Vehicle route example. Two infeasible routes are considered. Conflicting customers are identified (customers 2 and 5).



(b) Vehicle route example – restoration phase. A 2-opt like movement is performed between conflicting customers 2 and 5. One of the obtained routes becomes feasible. We then focus on the "new" infeasible route and optimise it.

Figure 1. An example of the restoration phase. Using 2-opt like movements among customers from infeasible routes we obtain routes that are feasible.

As we can see in Figure 1, two vehicle routes are infeasible. Then, we first exchange those "conflicting" customers from each route. For the example in Figure 1, we exchange customers 2 (from route 1) and 5 (from route 2). After doing this, we optimise the route using mathematical programming methods and see if the resulting routes are feasible. For the example in Figure 1, both routes result in feasible routes and no more changes are needed.

Algorithm 4 describes the restoration technique proposed in this paper.

```
Algorithm 4: Restoration Technique
  Input: A set of infeasible routes P
1 Begin
2  For each ρ∈P
3    Θ_ρ ←Identify Conflicting Customers( ρ );
4    Θ=Θ∪Θ_ρ
5  End
6  While ( P is not empty)
7   For each ρ∈P
8    If ρ is not feasible then
9      θ= Random (Θ_ρ)
10     θ'= Random (Θ\{{θ}∪{Θ_ρ})
11     ρ= UpdateRoute( ρ , θ , θ' )
12    optimise( ρ )
13   Else
14     P=P\{ρ}
15   End
16  End
17 End
20 End
```

In the next section we present the obtained results after applying both Tabu Search and Simulated Annealing to the VRPSPDTW.

5. Computational Experiments

As proposed in [2], the set of solutions used to test our algorithms is based on the Solomon's instances for the VRPTW [33]. Here, we assume that values given in Solomon's instances corresponds to the delivery demands d_i, $i=1,\ldots,n$; and pick-up demands p_i, $i=1,\ldots,n$, are as follows

$$p_i := \begin{cases} (1-\alpha)\, d_i & \text{if } i \text{ is even;} \\ (1+\alpha)\, d_i & \text{if } i \text{ is odd} \end{cases}$$

where $0 \le \alpha \le 1$. This new set of instances is similar to the one in [18]. Unlike in [18], in this study we only consider instances where customers are organised in clusters. Parameters for both Tabu Search and Simulated Annealing are as in Table 2. Parameters values in Table 2 were obtained after a trial and error process and, thus, might not be the best values for other instances.

Table 3 shows the set of instances that are considered in this study. The number of vehicles ranges from 5 to 7 and the number of customers ranges from 31 to 48. The maximum load capacity of a vehicle is set to 200 product units.

Table 2. Parameters for the Local Search Algorithms

Tabu Search	
Parameter	**Value**
maxIter	2000
tabuSize	6
diversLimit	35
listOfCandidates	5

Simulated Annealing	
Parameter	**Value**
maxIter	2000
α	0.85
intialTemp	50

Table 3. Set of Instances considered in this paper.

Instance	n	m	L^{max}
Spdtw1	31	5	200
Spdtw2	32	5	200
Spdtw3	37	5	200
Spdtw4	38	6	200
Spdtw5	43	6	200
Spdtw6	48	7	200

For each instance both algorithms are applied and the obtained results are presented in Tables 4 and 5. In general, Tabu Search tends to find solutions that are better in terms of cost. However, Tabu Search takes longer to converge. This is because it needs to call the restoration module more often, as more neighbours needs to be generated at each iteration.

Table 4. Results obtained by Simulated Annealing with restoration strategy

Instance	VRP Opt	SA II	Time (sec)	SA II %GAP
Spdtw1	672	682.39	1.156	1.54
Spdtw2	784	832.96	1.078	6.24
Spdtw3	669	685.14	1.031	2.41
Spdtw4	805	819.11	0.750	1.75
Spdtw5	742	770.87	1.062	3.89
Spdtw6	1073	1158.17	1.921	7.93

Table 4 shows the results obtained by the Simulated Annealing algorithm with the restoration strategy (SA II). As expected, solutions for the VRPSPDTW has a cost higher than the best solution of the corresponding VRP. The cost of the solutions the Simulated Annealing algorithm found is, in average, 3.96% higher than the cost of optimal solutions of the corresponding VRP problems.

Just as in Table 4, in Table 5 we present the results obtained by the Tabu Search algorithm.

Table 5. Results obtained by TS with restoration strategy

Instance	Opt	TS II	Time	Save %
Spdtw1	672	680.96	3.578	1.33
Spdtw2	784	787.08	3.437	0.39
Spdtw3	669	672.74	10.125	0.55
Spdtw4	805	808.70	4.531	0.45
Spdtw5	742	751.75	12.453	1.31
Spdtw6	1073	1107.45	21.438	3.21

These results considers the restoration strategy proposed in this paper.

On the one hand Tabu Search takes, in average, almost 8 seconds more than the Simulated Annealing algorithm to converge. On the other hand, Tabu Search finds solutions that are more than 50% cheaper than the ones found by the Simulated Annealing algorithm.

Figure 2 shows an instance from Table 5 that is solved in this paper. Circles represent the customers' location and the red triangle represents the depot position.

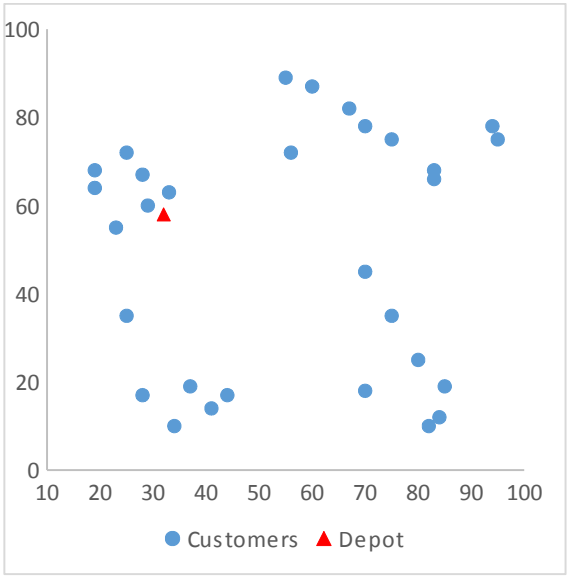


Figure 2. An example of customers and depot position in the Euclidean space.

From Figure 2 we can note that the customers are distributed in clusters. In particular, there are four clusters in the example in Figure 2. Figure 3 shows the routes for all five vehicles needed to solve the instance Spdtw1 using the Simulated Annealing algorithm and the restoration strategy proposed in Section 3.

It is interesting to note that vehicles do not necessarily visit the closest customer from their current position. This is because sometimes the capacity constraints make such movements infeasible. Moreover, when we compare the routes generated by both Tabu Search and Simulated Annealing from Figures 3 and 4, we can see that the algorithms converge to different solutions, even though they start from the same initial solution. This is because both algorithms explore the decision space differently.

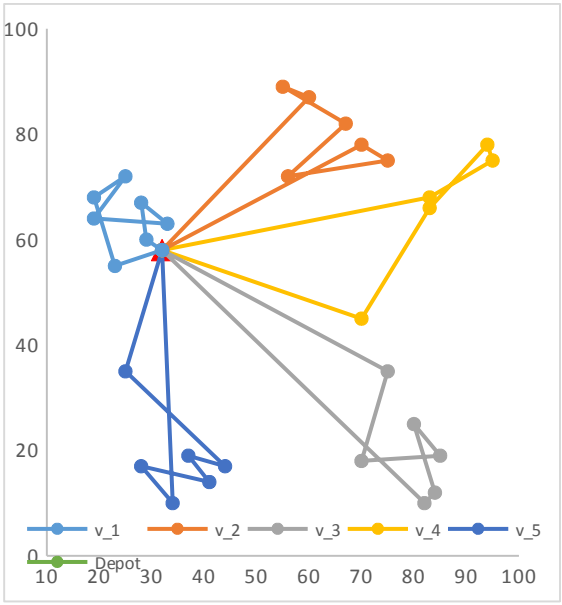


Figure 3. Vehicles routes for the instance Spdtw1 using the Tabu Search algorithm including restoration phase.

Figure 4 shows the routes found by the Tabu Search algorithm using the restoration strategy proposed in Section 3.

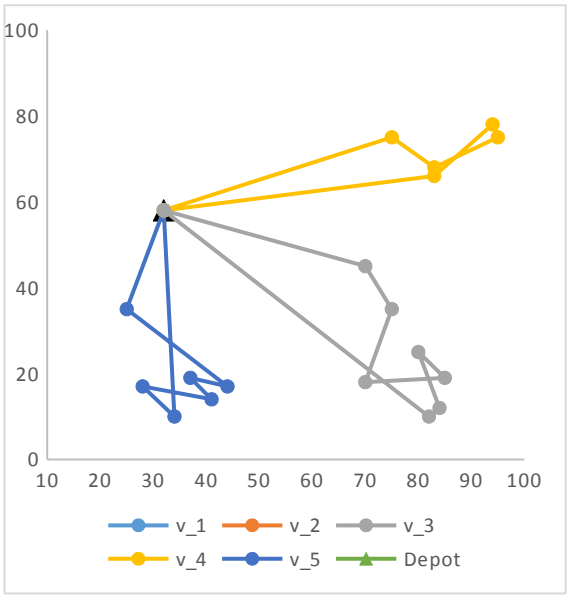


Figure 4. Vehicles routes for the instance Spdtw1 with a vehicle capacity equal to 200. All five available vehicles are used (v_1 to v_6).

Finally, Tables 6 and 7 shows the improvements on both algorithm after including the restoration strategy.

While for the Tabu Search including the restoration strategy proposed in this paper leads to solutions that are, in average, 3.9% cheaper, for the Simulated Annealing savings are equal to 5.9%. However, the restoration strategy makes Tabu Search considerably slower. As mentioned before, this is because Tabu Search needs to generate many neighbours and often they are not feasible, thus the restoration strategy needs to be applied.

Table 6. Simulated Annealing before (SA) and after (SA II) considering the restoration strategy proposed in this paper.

Instance	SA	SA II	Save %
Spdtw1	698.87	682.39	2.36%
Spdtw2	855.87	832.96	2.68%
Spdtw3	777.79	685.14	11.91%
Spdtw4	851.21	819.11	3.77%
Spdtw5	825.26	770.87	6.59%
Spdtw6	1251.60	1158.17	7.46%

Although this situation also occurs in the Simulated Annealing algorithm, since this algorithm generates fewer neighbours per iteration the impact in the computational time required to converge is marginal.

Table 7. Tabu Search before (TS) and after (TS II) considering the restoration strategy proposed in this paper.

Instance	TS	TS II	Save %
Spdtw1	681.17	680.96	0.03%
Spdtw2	830.81	787.08	5.26%
Spdtw3	741.89	672.74	9.32%
Spdtw4	818.06	808.70	1.14%
Spdtw5	790.61	751.75	4.91%
Spdtw6	1144.48	1107.45	3.24%

5. Conclusions

In this paper we have presented a comparison between two well-known local search algorithms, namely Tabu Search and Simulated Annealing. Results confirm the effectiveness of these two heuristics when solving the VRPSPDTW. Although both algorithms find solutions that are quite competitive to other previously proposed strategies (such as the ant colony optimisation in [18]) within an acceptable time, we noted that the algorithms have some problems when dealing with infeasible neighbours. As a consequence, the algorithms sometimes took too long to find a feasible neighbourhood. To solve this issue, a generic restoration strategy is proposed in this paper. Results show that local search algorithms considered in this paper consistently obtain better results when they make use of the restoration strategy.

As a future work we expect to try the restoration strategy within heuristic algorithms other than Tabu Search and Simulated Annealing. For instance, variable neighbourhood search as well as population based algorithms such as particle swarm optimisation and ant colony optimisation might be considered.

REFERENCES

1. AI, T. J., V. A. KACHITVICHYANUKUL, **Particle Swarm Optimization for The Vehicle Routing Problem with Simultaneous Pickup and Delivery**, Computers & Operations Research, vol. 36(5), 2009, pp. 1693-1702.
2. ANGELELLI, E., R. MANSINI, **The Vehicle Routing Problem with Time Windows and Simultaneous Pick-up and Delivery. Quantitative Approaches to Distribution Logistics and Supply Chain Management**, Lecture Notes in Economics and Mathematical Systems, vol. 519, 2002, pp. 249-267.
3. AREIBI, S., G. GREWAL, D. BANERJI, P. DU, **Hierarchical FPGA Placement**. Canadian Journal of Electrical and Computer Engineering, vol. 32(1), 2007, pp. 53-64.
4. AVCI, M., S. TOPALOGLU, **An Adaptive Local Search Algorithm for Vehicle Routing Problem with Simultaneous and Mixed Pickups and Deliveries**. Computers & Industrial Engineering, vol. 83(C), 2015, pp. 15-29.
5. BARANY, M., Z. TUZA, **Circular Coloring of Graphs via Linear Programming and Tabu Search**. Central European Journal of Operations Research, vol. 23(4), 2015, pp. 833-848

6. BRAEKERS, K., K. RAMAEKERSA, I. V. NIEUWENHUYSEC, **The Vehicle Routing Problem: State of the Art Classification and Review**. Computers & Industrial Engineering, to be published.

7. CABRERA, G., S. Roncagliolo, J. P. RIQUELME, C. CUBILLOS, R. SOTO, **A Hybrid Particle Swarm Optimization-Simulated Annealing Algorithm for The Probabilistic Travelling Salesman Problem**, Studies in Informatics and Control, vol. 21, 2012, pp. 49-58.

8. CAO, E., M. LAI, **An Improved Differential Evolution Algorithm for the Vehicle Routing Problem with Simultaneous Delivery and Pick-up Service**, Proceedings of the 3rd International Conference on Natural Computation (ICNC), 2007, pp. 436-440.

9. CHEN, J. F., T. H. WU, **Vehicle Routing Problem with Simultaneous Deliveries and Pickups**. Journal of the Operational Research Society, vol. 57, (5), 2006, pp 579-587.

10. COSSI, A. M., J. R. S. MANTOVANI, **Integrated Planning of Electric Power Distribution Networks**. IEEE Latin American Transactions, vol 7(2), 2009, pp. 203-210.

11. DANTZIG, G. B., J. H. RAMSER, **The Truck Dispatching Problem**. Management Science, vol. 6(1), 1959, pp. 80-91.

12. DETHLOFF, J., **Vehicle Routing and Reverse Logistics: The Vehicle Routing Problem with Simultaneous Delivery and Pick-Up**. OR-Spektrum, vol. 23(1), 2001, pp. 79-96.

13. GAJPAL, Y., P. ABAD, **An Ant Colony System (ACS) for Vehicle Routing Problem with Simultaneous Delivery and Pickup**, Computers & Operations Research, vol. 36(1), 2009, pp. 3215-3223.

14. GAN, X., Y. WANG, S. LI, B. NIU, **Vehicle Routing Problem with Time Windows and Simultaneous Delivery and Pick-Up Service Based on MCPSO**. Mathematical Problems in Engineering, vol. 2012, Article ID 104279, 2012, pp. 1-11. doi:10.1155/2012/104279.

15. GLOVER, F., **Artificial Intelligence, Heuristic Frameworks and Tabu Search**. Managerial and Decision Economics, vol. 11(1), 1990, pp. 365-375.

16. GOKSAL, F. P., I. Karaoglan, F. ALTIPARMAK, A **Hybrid Discrete Particle Swarm Optimization for Vehicle Routing Problem with Simultaneous Pickup and Delivery**. Computer and Industrial Engineering, vol. 65(1), 2013, pp. 39-53. doi: 10.1016/j.cie.2012.01.005.

17. HIERMANN, G., M. PRANDSTETTER A. RENDL, J. PUCHINGER, G. R. RAIDL, **Metaheuristics for Solving A Multimodal Home-Healthcare Scheduling Problem**. Central European Journal of Operations Research, vol. 23(1), 2015, pp. 89-113.

18. JOHNSON, F., J. VEGA, G. CABRERA, E. CABRERA, **Ant Colony System for a Problem in Reverse Logistic**, Studies in Informatics and Control, vol. 24(2), 2015, pp. 133-140.

19. KASSEM, S., M. CHEN, **Solving Reverse Logistics Vehicle Routing Problems WITH Time Windows**. The International Journal of Advanced Manufacturing Technology, vol. 68(1), 2013, pp. 57–68, doi: 10.1007/s00170-012-4708-9.

20. KIRKPATRICK, S., **Optimization by Simulated Annealing: Quantitative Studies**. Journal of Statistical Physics, vol. 34, no. 5-6, 1984, pp. 975-986.

21. KIRKPATRICK, S., C. D. GELATT, M. P. VECCHI, **Optimization by Simulated Annealing**. Science, vol. 220, no. 4598, 1983, pp. 671-680.

22. KOLEN, A. W. J., A. H. G. RINNOOY KAN, H. W. J. M. TRIENEKENS, **Vehicle Routing with Time Windows**. Operations Research, vol. 35(2), 1987, pp. 266-273.

23. LAGOS, C., B. CRAWFORD, E. CABRERA, R. SOTO, J. M. RUBIO, F. PAREDES, **Combining Tabu Search and Genetic Algorithms to Solve the Capacitated Multicommodity Network Flow Problem**. Studies in Informatics and Control, vol. 23(3), 2014,pp. 265-276.

24. LAGOS, C., B. CRAWFORD, R. SOTO, E. CABRERA, J. VEGA, F. JOHNSON, F. PAREDES, **Improving Tabu Search**

Performance by Means of Automatic Parameter Tuning, Canadian Journal of Electrical and Computer Engineering, vol. 39(1), 2016, pp. 51-58.

25. LIU, R., X. XIE, V. AUGUSTO, C. RODRIGUEZ, **Heuristic Algorithms for a Vehicle Routing Problem with Simultaneous Delivery and Pickup and Time Windows in Home Healthcare**. European Journal of Operational Research, vol. 230(3), 2013, pp. 475-486.
26. LUNDY, M., A. MEES, **Convergence of an Annealing Algorithm**. Mathematical Programming, vol. 34(1), 1986, pp. 111-124.
27. METROPOLIS, N., A. W. ROSENBLUTH, M. N. ROSENBLUTH, A. H. TELLER, E. TELLER, **Equation of State Calculations by Fast Computing Machines**. The Journal of Chemical Physics vol. 21(6), 1953, pp. 1087-1092.
28. MIN, H. **The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Points**. Transportation Research Part A General 23(5), 1989, pp. 377-386.
29. MLADENOVIĆ, N., P. HANSEN, **Variable Neighborhood Search**. Computers & Operations Research, vol. 24(11), 1997, pp. 1097-1100.
30. NAGY, G., S. SALHI, **Heuristic Algorithms For Single And Multiple Depot Vehicle Routing Problems With Pickups And Deliveries**. European Journal of Operational Research, vol. 162(1), 2005, pp. 126-141.
31. RIECK, J., J. ZIMMERMANN, **Exact Solutions to the Symmetric and Asymmetric Vehicle Routing Problem with Simultaneous Delivery and Pick-Up**. Business Research, vol. 6(1), 2013, pp. 77-92.
32. RODRIGUEZ, D. A., A. C. OLIVERA, N. B. BRIGNOLE, **Vehicle Routing for Public Transport with Adapted Simulated Annealing**. Latin American Applied Research, vol. 44(3), 2014, pp. 247-252.
33. SOLOMON, M. M., **Algorithms for the Vehicle Routing Problem with Time Windows**. Transportation Science, vol. 29(2), 1995, pp. 156-166.
34. SUBRAMANIAN, A., L. S. OCHI, **New Lower Bounds for the Vehicle Routing Problem with Simultaneous Pickup and Delivery**, Technical Report - RT 01/10, Universidade Federal Fluminense, Niteri-RJ, Brazil, 2010.
35. TANG, F. A., R. D. GALVAO, **A Tabu Search Algorithm for the Vehicle Routing Problem with Simultaneous Pick-Up and Delivery Service**. Computers & Operations Research, vol. 33, 2006, pp. 595-619
36. TOADER, F. A., **A Hybrid Algorithm for Job Shop Scheduling Problem**. Studies in Informatics and Control, vol. 24(2), 2015, pp. 171-180.
37. WANG, C., F. ZHAO, D. MU, J. W. SUTHERLAND, **Simulated Annealing for a Vehicle Routing Problem with Simultaneous Pickup-Delivery and Time Windows**. Advances in Production Management Systems. Sustainable Production and Service Supply Chains, vol. 415(2), 2013, pp 170-177.
38. WILHELM, M. R., T. L. WARD, **Solving Quadratic Assignment Problems by Simulated Annealing**, IIE Transactions, vol. 19(1), 1987, pp. 107-119.
39. ZACHARIADIS, E., C. D. TARANTILIS, C. KIRANOUDIS, **A Hybrid Metaheuristic Algorithm for the Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Service**. Expert Systems with Applications, vol. 36(2), 2009, pp. 1070-1081
40. ZHU, N., C. SHAO, **Vehicle Routing Problem with Simultaneous Delivery and Pick-up Based on the Improved Genetic Algorithm**, in Proc. of the 4th International Conference on Genetic and Evolutionary Computing (ICGEC), Shenzhen, China, 2010, pp. 312-316.