Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows

G. B. Alvarenga
Department of Computer Science
Federal University of Lavras, UFLA, Lavras - Brazil
e.mail: bastos@comp.ufla.br

G. R. Mateus
Department of Computer Science
Federal University of Minas Gerais, UFMG, Belo Horizonte - Brazil
fax.: +55 - 31 – 34995879, e.mail: mateus@dcc.ufmg.br

G. de Tomi
Department of Mining Engineer
University of São Paulo, USP, São Paulo - Brazil
e.mail: gdetomi@usp.br

Abstract

The Vehicle Routing Problem with Time Windows (VRPTW) is a well-know and complex combinatorial problem, which has received considerable attention in recent years. This problem has been addressed using many different techniques including both exact and heuristic methods. The VRPTW benchmark problems of Solomon (1987) have been most commonly chosen to evaluate and to compare all solutions proposed in the literature. Results from exact methods have been improved considerably because of parallel implementations and modern branch-and-cut techniques. However, 25 out of the 56 high order instances from Solomon's test set still remain unsolved. Additionally, in many cases a prohibitive time is needed to find the exact solution. Many efficient heuristic methods have been developed to make possible a good solution in a reasonable amount of time. Unfortunately, while the publications on exact methods have considered the total travel distance as their main objective, almost all the heuristic attempts have considered the total number of vehicles as their main objective. Consequently, it is more difficult to compare and to take advantage of the strong points from each approach. With travel distance as the main objective, this paper proposes a robust heuristic approach for the VRPTW problem using an efficient Genetic Algorithm and a MIP formulation. With this innovative approach it is possible to compare the results with the exact methods published. Additionally, computational results show that the proposed heuristic approach outperforms all previously known heuristic methods published, in terms of the minimum travel distance.

1. Introduction

The VRPTW has been extensively studied in the operations research community. Firstly, because VRPTW is still one of the most difficult problems in combinatorial optimization and consequently presents a great challenge. Secondly, in a more practical aspect, this problem contributes directly to a real opportunity to reduce costs in the important area of logistics. Logistics can be roughly described as the delivery of goods from one place (supplier) to others (consumers). Transportation management, and more specifically vehicle routing, has a considerable economical impact on all logistic systems. In the VRPTW, a fleet of K identical vehicles supplies goods to N customers. All vehicles have the same capacity Q. For each customer i, $i = 1,\ldots,N$, the demand of goods, q_i, the service time s_i, and the time window $[a_i,b_i]$ to meet the demand in i are known. The component s_i represents the unloading service time at the customer i and a_i describes the earliest time when it is possible to start the unloading

service. If any vehicle arrives at customer i before a_i it must wait. The vehicle must start the customer unloading service before b_i. This type of time window constraints is known as a *hard time window*. All vehicle routes start and finish at the central depot. Each customer must be visited once. The locations of the central depot and all customers, the minimal distance d_{ij} and the travel time between all locations t_{ij}, are given. The objective is to find the feasible solution with the minimum total travel distance or with the minimum number of vehicles. In this paper we explore the first objective.

Significant improvements in Solomon's benchmark problem instances were established by Rochat [4] using a taboo search metaheuristic method. In that publication of 1995, Rochat improved or reached 47 heuristic solutions from 56 Solomon's original instances. Another important characteristic was the post-optimization technique used by Rochat. This technique consists of saving all partial solutions identified during the taboo search algorithm for future usage. The routes of each solution are included in the set *T*. Then, after the stop criterion of taboo search has been applied, the best solution that can be built using routes from *T* may be found by solving a set partitioning problem using CPLEX[1] MIP software. Although Rochat et al. have minimized the travel distance in an instance set of Capacitated Vehicle Routing Problem (CVRP), the published results for the Solomon time windows version were obtained using the number of vehicles as the first objective, where the second, subject to optimizing the first, was to obtain the minimum travel distance.

Berger et al. [6] have improved some of the results of Solomon's benchmark using a parallel two-population co-evolution genetic algorithm. In general, heuristic methods have chosen the number of vehicles as their first objective and the total travel distance as the second. However, considering the Solomon's test set results in many research works, the number of vehicles (NV) is the same in almost all instances, so the second objective, travel distance (TD), is the distinguishing criterion. In fact, the number of vehicle and the travel distance represent competitive objectives. Strategies which treat each objective separately are known to have reached the best results. Berger, for example, searches for NV and TD minimization at the same time; however there are two processes where each objective is issued independently, avoiding any eventual competition.

Homberger et al. [5] have also presented good solutions for many problems of the Solomon's benchmark using two evolutionary metaheuristic methods in a similar two-stage strategy. Few papers on heuristic methods have addressed the travel distance as the first objective to be minimized, as indicated in Rousseau et al. [7] and Tan et al. [8]. In [8] the authors seem to compare their first objective, travel distance , with results from research works in the literature where the number of vehicles was the priority and, consequently, produced worse TD results. Their comparison criteria are not clear.

The minimum number of vehicles has been selected in almost all previous works based on heuristic. In opposite, almost all exact works have elected the travel distance as the unique objective. In fact, the better strategy will be very dependent from the rules and particularities of the business of each logistic company in the real world. In Brazil, for example, a great part of companies, which delivery goods, has used a model in which they are owner from approximately a half part of the fleet. An obligatory

[1] CPLEX is a very popular and commercial package for linear and integer linear programming.

exceeding of goods is always delivered by contractors. Normally, these contractors are little companies or even the owner of the individual truck, namely "aggregated". In that common situation, the logistic company has no compromise with any fixed investment as the truck and the payment rule is based only in the cost of the goods and travel distance predicted by each route. Other scenario, even with no contractor, it's not difficult a day or a week when the total amount of goods is lower than the total capacity of the trucks. In that case, is better any travel distance reduction even it results in all trucks on the road.

Consequently, these scenarios justify the study of new algorithms and techniques to improve the results in terms of travel distance. Additionally, the deficiency of heuristic works in that direction also justifies this work.

2. Hybrid Genetic Algorithm

This approach is based on several island of GA evolutions. The island idea, from the natural evolution theory, is related with a set of individuals where there is the possibility to change theirs genetic material. In the other side, each island evolution is completely isolated from each other. Each island evolution i will produce a subset of "nice" routes T_i. In the first step, for each island, the framework proposed uses a genetic algorithm (GA) to identify several local optima for the VRPTW using $L_{\max}$ independently and sequentially GA evolutions. All routes of the best individual from each GA evolution are included in a subset T_i. The *Algorithm 1* is illustrated as follows:

Algorithm 1

For *t = 1 to* L_{max} ***do {***

Generate a new solution L_t *using GA_algorithm;*

Insert all routes of L_t *in* T_i*;*

};

Then, after the number of L_{max} local minimum were found, a set partitioning formulation is obtained and solved using all routes saved in subset T_i. The set partitioning formulation is solved using an exactly MIP algorithm available in the GLPK 3.2.1 package (GNU library). Consequently, improved solution results are obtained from the best feasible combination of all previous saved routes in T_i. Again, this improved solution is only a local minimum because only a limited set of all possible routes, the subset T_i, is considered in the set partitioning model as follow:

$$\min \quad \sum_{j=1}^{|T_i|} c_j x_j$$

$$\text{subject to} \sum_{j=1}^{|T_i|} a_{ij} x_j = 1 \qquad i = 1,...,N$$

$$x_j \in \{0,1\} \qquad j = 1,....,|T_i|$$

Where the subset T_i includes all routes found, as described above. The number of customers is N, c_j is the cost of j^{th} route in T_i ($j = 1,\ldots,|T_i|$); in this case the cost c_j is the travel distance. For $i = 1,\ldots,N$ and $j = 1,\ldots,|T_i|$:

$$a_{ij} = \begin{cases} 1 & \text{if customer } i \text{ is attended by } j \\ 0 & \text{if customer } i \text{ is not attended by } j \end{cases}$$

The assignment of x_j is an integer variable of the problem and equal to 1 if the route j is present in the solution and 0 otherwise.

Exactly as in natural biology, any genetic combination is only possible between individuals in that particular island, because other individuals are geographically isolated. In the present work, because the set partitioning model combines all solutions in T_i, the concept of island evolution is extended to the entire evolution represented by *Algorithm 1* and *Algorithm 2* together. Then, to further improve the local minimum reached in *Algorithm 1*, a second step algorithm is used. In this algorithm, the starting point is the best solution from the set partitioning problem. Many (S_{max}) incomplete subsets of routes from the current solution are chosen and the previous GA subroutine is used to try improving the cost of these specified routes further, once the order of the problem was reduced. The *Algorithm 2* is outlined as follows:

Algorithm 2

For *s = 1 to* S_{max} ***do***

For *r= 0 to Number_of_Routes* ***do {***

Initialize *an auxiliary subset of routes* R_{aux} *to null* ***;***

If *random(Number_of_Routes) < (ReducedOrderFactor* Number_of_Routes)* ***then***

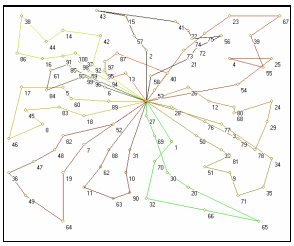
Insert all customer from route r in a subset R_{aux}*;*

}

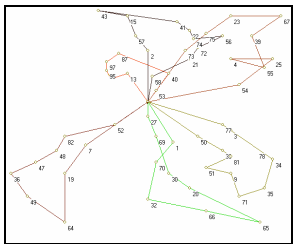
Generate a new solution L_s *using the same GA_algorithm, considering only customers in* R_{aux}*;*

Insert all routes of L_s *in* T_i*;*

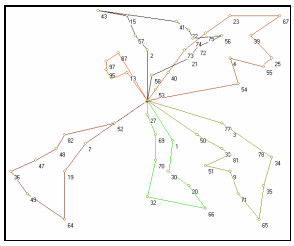
} // End procedure



Phase I – Best solution from the Set Partitioning Problem over T_i



Phase II – A subset of routes randomly chosen from Phase I.



Phase III – A hypothetical routes improved from Phase II.

Fig.1. Action of Algorithm 2 in a current solution example, when each loop step solves a lower order problem, may find an improved subset of routes. Note the solution in III can be updated in the original solution because there is no conflict with any other route.

The *Algorithm 2,* in the line 4, randomizes a number from 0 to *Number_of_Routes* (number of routes in the current solution from the *Algorithm* 1). Considering the constant *SubproblemOrderFactor* less than 1 (0.6, for example), the number of customer utilized in the *Algorithm 2* will be reduced, as desired**.** The Fig.1 illustrates the result of each interaction in the *Algorithm 2.* After *Algorithm 2*, the set partitioning model described is solved again, but now T_i has many additional routes, resulting from *Algorithm 2*. Many routes identified from *Algorithm 2* are similar or exactly the same as previous routes inserted in T_i, for example, when the same solution was found by both algorithms.

The process described, resulting each island in this context, is summarized in Fig.2, T_1 represents the total set of routes present during all evolution over the island 1, that is, i=1.

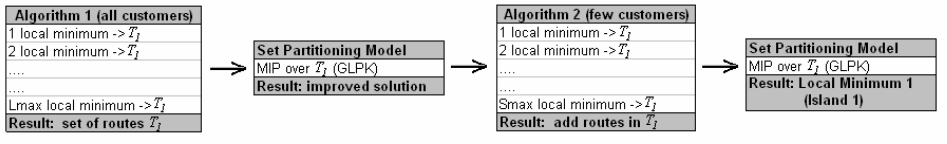


Fig.2. Island evolution 1: all steps, generating new "nice" routes to T_i until reach a new improved local minimum

The process described has the characteristic of finding a good local minimum in a reasonable amount of time. The number of GA evolutions used in *Algorithm 1* was 10, L_{max}*=10*. This parameter is fixed along all tested instances. That choice was based on experimentation. The number of GA evolutions in *Algorithm 2* was 25, S_{max}*= 25,* also fixed after experimentation.

In this research work, all attempts to increase the number of local minimum in *Algorithm 1* or *2* and all attempts to increase the number of populations in each GA evolution resulted in significantly higher execution time with nothing or only very little gain in terms of quality of the solution. However, a significative solution improvement was achieved when many independent islands were executed and, again, all results included in a new set partitioning problem, as shown in Fig.3. The main idea of the proposed framework is motivated by the perception that in any reasonable local minimum it is possible to find many routes also present in the global minimum (exactly the best solution). It is easier to produce several local minimum rather than commit hard computations efforts to avoid these ones. Additionally, an exact method to build the best combination of all routes identified is possible and fast to execute.

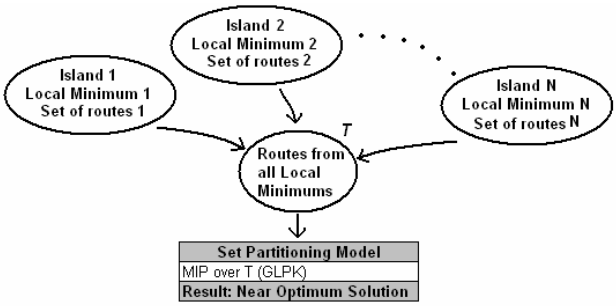


Fig.3. Global framework overview: many local minimums to obtain a near optimal end solution for the VRPTW.

2.1. Genetic Algorithm

The first algorithms that use a natural evolution as the central strategy to solve problems were published in the 90's, such as Fraser [9] and Box [10]. In 1966 Forgel et al. [11] proposed a method called Evolutionary Programming. Following that, in 1973, Rochenberg [12] introduced the method called Evolution Strategies. The proper Genetic Algorithm, or simply GA, was proposed by Holland [13] in 1975. All these proposals were based in the natural reproduction, selection and evolution theory from Darwin [14], 1859. Since then, the genetic algorithm has been popular because it can contribute in finding good solutions for complex mathematical problems, like the VRP and others NP-complete problems.

The GA is based in the chromosomes or individuals population, where each individual represents a solution proposed to the problem issued. When new chromosomes or individuals are created, new proposals for the solution are generated by combining the chromosomes present in the current individuals. The individuals are grouped in generations. The set of all individuals is called population. The most important individual property is the fitness. The individual fitness is the objective function which is evaluated using the parameters values coded in each individual.

Generally, GA works on maximization, therefore the higher is the objective function the better; the higher the results from a specific individual parameters value, the better will be the fitness of this individual. Additionally, a higher fitness value results in a larger chance for the individual to participate in a crossover (with its parents), which will generate the individuals in the next generations. In fact, a higher fitness value reflects how intensive the local search in that region in the search space was. Therefore, an individual with good fitness induces a search in that direction. The crossover is done after some kind of individuals selection, partially random based, and partially based on the quality of the individual fitness. The simplest way to do a crossover is breaking up two chromosomes in a random point and exchange them sideways, as illustrated in Fig.4:

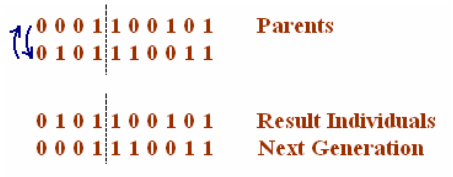


Fig.4. Simple crossover

Successive generations are made up until some stop criterion is reached, normally, the computational time spent or the numbers of individuals generated. The basic GA is described as follows:

Algorithm 3

```
function GA : individual;
{
        generation_number = 1;
        Start_initial_population;
```

```
        Population_evaluation;
        while (generation_number < MAX_GENERATION) do {
                inc (generation_number );
                selection;
                crossover;
                mutation;
                Population_evaluation;
        }
        return(best_individual_current_population);
}
```

2.1.1. Chromosome and Individual representation

The representation of the GA chromosome in the present work is very simple. Each customer has a unique integer identifier i, $i = 1,\ldots,N$, where N is the number of customers. The chromosome is defined as a string of integers as shown in Fig.5. For each vehicle with at least one customer to be visited, one chromosome is allocated. An individual, who represents a complete solution, and consequently many routes, is a set of chromosomes. The central depot is not considered in this representation, because all routes necessarily start and end in it.

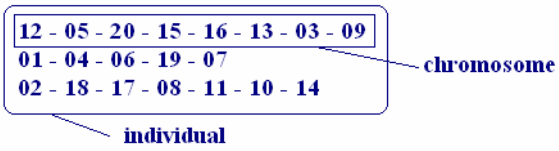


Fig.5. An individual with 3 chromosomes, or 3 vehicles (routes). It represents a solution for a problem instance with 20 customers.

2.1.2. Initial population

A fast and simple heuristic procedure to distribute all customers in the vehicles can reduce significantly the time of convergence of the main optimization algorithm. The heuristic method proposed by Solomon [15], namely PFIH (*Push Forward Insertion Heurist*), has been frequently used by many researchers in the past.. For a detailed description of the PFIH method, see [16]. In the present work PFIH was utilized; however a total randomized choice is made for the first customers, in each new route, as opposed to the equation proposed originally. This is necessary to increase the diversification in the first generation of the GA. Once the first customer in a new route is a randomized choice, the second customer will be the one with the minimal cost increase from all free customers. Each feasible inserted position in a route is evaluated. A new routed is created only when no new free customer is feasible.

2.1.3. Selection

Through this step, pairs of parents are selected to crossover. There are many methods proposed in the literature for selection step in genetic algorithms. The two most popular are roulette wheel selection

and tournament selection. In the roulette wheel selection method, the probability of an individual to participate in a crossover is directly proportional to his relative fitness. This method is very sensitive to details of the evaluation function and almost always some additional control is necessary, for example fitness scaling.

In the work, a k-way tournament selection method was used in the GA. In a k-way tournament, k individuals are selected randomly. Then, the individual who presents the highest fitness is the winner. This process is repeated until the number of selected individuals necessary to crossover has been reached.

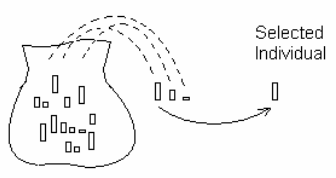


Fig.6. The k-way tournament selection. After the pool has been chosen, the winner is the larger fitness

2.1.4. Fitness

The fitness function to evaluate the individuals is commonly obtained from the objective function of the optimization problem but not necessarily coincide with it. However, in the present work a slight modification was necessary, to enable the GA to work with maximization. The inverse of the travel distance was used to calculate the fitness of the individuals.

2.1.5. Crossover

An important characteristic to be considered is the possibility of generating infeasible individuals. If infeasible individuals are generated and there is no way to bring them back to the feasible region, the algorithm may fail. Additionally, it should also be considered how much effort is required for searching in the infeasible region. Berger [6] used parallel co-evolution of two populations. In the first population, only feasible individuals are accepted and the objective is the minimization of the travel distance. However, Berger key strategy is to force the number of vehicles to be one unit less than the current feasible solution and to minimize the total time constraint violation in order to generate a new feasible solution. Both alternatives can be found in the literature, both with consistent results.

In this research work, the search space was limited to the feasible region. Therefore every individual is always feasible. A careful consideration is necessary in the crossover and mutation operators in this way for the VRPTW, because a simple exchange between two customers can violate time and capacity constraints. Consequently, a more complex operator is necessary to be introduced without bringing a bias in any particular direction. A new crossover algorithm was written in order to address the following features:

- Maximize all the original routes present in the new chromosome, exactly as found in its parents;

- Equalize the number of routes inherited from each parent;
- Avoid exchanging customers which significantly increase the cost of the solutions previously reached.
- Avoid new individuals with a number of vehicles higher than their parents.

In order to achieve these objectives the following algorithm was written:

Algorithm 4 – Crossover Algorithm

```
Procedure Crossover( Indiv1, Indiv2 : Individual; NewIndiv : ^Individual );
{
  // First Step: it's possible entire Parents routes insertion in New Individual
   repeat
        EntireRouteIndiv1Possible = False;
        repeat
              Route1 = Indiv1.NewRoute; {this procedure return only not used routes}
              If Route1 is possible in NewIndiv then EntireRouteIndiv1Possible = True;
        until (Route1=nil) or (EntireRouteIndiv1Possible = True);
        if Route1 <> nil then InsertNewIndiv(Route1);
        EntireRouetIndiv2Possible = False;
        repeat
              Route2 = Indiv2.NewRoute; {this procedure return only not used routes}
              If Route2 is possible in NewIndiv then EntireRouteIndiv2Poissible = True;
        until (Route1=nil) or (EntireRouteIndiv2Possible = True);
        if Route1 <> nil then InsertNewIndiv(Route2);
   until (EntireRouteIndiv1=False) and (EntireRouteIndiv2=False);
   // Second Step: insert not routed customers in previous routes
   for i= 1 to N do {
       if Customer[i] not routed in NewIndiv then {
             for v= 1 to MaxVehicle do {
                   possible = Insert (Customer[i], NewIndiv, Vehicle[v]);
                   if possible then {
                         set Customer[v] routed;
                         break;
                   }
             }
       }
   }
   // Third Step: Create new routes for remainders
   Insert Remainder Customers in new routes using PFIH;
} // End procedure;
```

In the first step, the algorithm makes a random route choice from each parent individual in turn. After all feasible routes have been inserted; each remaining customer is tested against the possibility of inserting it in a non empty route of the new individual (second step). In the event of some customers still continuing to be not routed, then any feasible route can be inserted, and there is no other option other than inserting new vehicles (routes) in that new individual.

2.1.6. Mutation

A total of 8 different operators were used in this phase of the genetic algorithm proposed. The mutation process is necessary for inserting new characteristics that are eventually not present in all current individuals. Without considering mutation, the genetic algorithm search is limited to a very small area in the total feasible region. In the present work, some very specialized operators were created to improve the evolution of the individuals, as explained below.

Mutation_01 (Random Customer Migration): This operator chooses a random vehicle and a random customer associated to it; it then tries an insertion of this customer in other non-empty vehicles in this individual. If the insertion is feasible it is accepted, independently of related new costs.

Mutation_02 (Bringing the Best Customer): This operator chooses a random vehicle and searches for the customer from others vehicles which represents the minimal increase in its route cost (travel distance).

Mutation_03 (Re-insertion using PFIH): This operator chooses a random route in a random individual and it applies the PFIH modified procedure, as described in section 2.1.2.

Mutation_04 (Similar Customer Exchange): This operator chooses a random route in a random individual and it searches for a similar customer in terms of time window (minimal time distance) and then it tries to exchange it.

Mutation_05 (Exchanging Customer with Positive Gain): This operator is very expensive in terms of time demand (N^2), but nevertheless it is very important. It verifies each possibility to exchange any two customers and it only carries it out if a reduction in the total travel distance is obtained.

Mutation_06 (Merging Two Routes): This operator chooses two random routes and it tries to merge then in a random way. Very often, the remaining customers are inserted in other routes or in new ones.

Mutation_07 (Reinserting Customer): This operator chooses a random customer, removes it, and re-inserts in a better position (with a minimal distance traveled).

Mutation_08 (Route Partitioning): This operator chooses a random vehicle, a random customer and it divides this route in two others, using this customer as point of reference;

2.1.7. Elitism

In order to guarantee that GA does not compromise the quality of the results, an elitism strategy was adopted. This method consists in making the best individual from the last population always present in the new population.

2.1.8. General Parameters Tuning

Appropriate tuning of parameters in genetic algorithms can make a significant difference in terms of performance. Some values can provide very high performances in a specific instance while giving premature convergence in others, even over the same kind of problems. Today, robust genetic algorithms with very specialized and complex operators are developed for many problems. For the VRPTW, this is very critical, because there is not only one kind of instance but many classes of problems. For example, some instances are very relaxed in terms of time windows constraints. In others instances, customers sequences are absolutely necessaries to satisfy time restrictions.

In this work it was possible to define a satisfactory set of operators which allowed a robust and fast algorithm. However, considering the set partitioning problem used to complete the search, an emphasis in a fast convergence is predominant. This is because the resulting algorithm does not intend to find a global optimal as a stand-alone GA evolution, but it guarantees a sufficient proximity from it.

All parameters were fixed in all instances as shown below:

Number of generation in GA Algorithm 1 (`generation_number`)	170
Number of generation in GA Algorithm 2 (`generation_number`)	120
Number of individuals in a Population in Algorithm 1 and 2	31
Number of Local Minimal (L_{max}) found by Island in GA algorithm 1	8
Number of Local Minimal (L_{max}) found by Island in GA algorithm 2	25
Number of Island executed (Improved Local Minimal)	8
Mutation01 (attempt, a positive result is not guaranteed)	20
Mutation02 (attempt, a positive result is not guaranteed)	10
Mutation03 (attempt, a positive result is not guaranteed)	1
Mutation04 (attempt, a positive result is not guaranteed)	20
Mutation05 (attempt, a positive result is not guaranteed)	2
Mutation06 (attempt, a positive result is not guaranteed)	1
Mutation07 (attempt, a positive result is not guaranteed)	30
Mutation08 (attempt, a positive result is not guaranteed)	1
k used in the “k-way tournament”	3

3. Computational Results

The results presented in Table 1 are the best from 3 runs using the framework proposed for each instance in the Solomon’s test set problems. In fact, the algorithm has proven to be very robust, with

results always less than 1% from the optimum, when an optimal solution is known (from some exact methods in the literature). The column "Best" in Table 1 shows the exact solutions whenever they exist. As described bellow the exact algorithms in the literature have used 1 decimal of precision. The procedure is then to multiply all input data for 10 and truncate it. This has enabled the researchers to work with integers only. When the exact solution is not known, the "Best" column of Table 1 contains the best known results, obtained with heuristic methods with double precision. In Table 1, these cases were marked with a (*).

The Table 1 shows that many previously known optimal solutions from exact methods have been reached, 22 out of 31 (62.8%). It is possible to know the behavior of the heuristic method proposed with the same assumptions with the same approach, that is, with travel distance as the first objective and working with integers, 1 decimal truncated.

When the exact solution is missing, the results of the hybrid GA commonly overmatch the results from heuristic methods. However, there are two considerations to be made. The comparisons are carried out with different precision algorithms and there are only few research works in which the travel distance is minimized using heuristics, [1], [7] and [8].

	Best Publis.		Hybrid GA			Best Publis.		Hybrid GA			Best Publis.		Hybrid GA	
Prob.	NV	TD	NV	TD	Prob.	NV	TD	NV	TD	Prob.	NV	TD	NV	TD
R101	20	1637.7	***20***	***1637.7***	C108	10	827.3	***10***	***827.3***	C206	3	586.0	***3***	***586.0***
R102	18	1466.6	***18***	***1466.6***	C109	10	827.3	***10***	***827.3***	C207	3	585.8	***3***	***585.8***
R103	14	1208.7	***14***	***1208.7***						C208	3	585.8	***3***	***585.8***
R104	10	982.01*	11	978.6	R201	8	1143.2	8	1179.8	RC101	15	1619.8	16	1627.1
R105	15	1355.3	15	1356.0	R202	4	1088.07*	6	1040.8	RC102	14	1457.4	14	1461.9
R106	13	1234.6	***13***	***1234.6***	R203	4	912.98*	6	891.7	RC103	11	1258.0	12	1273.6
R107	11	1064.6	11	1066.8	R204	3	824.62*	4	739.9	RC104	10	1135.48*	10	1144.9
R108	9	960.88*	10	949.4	R205	3	994.42*	5	966.9	RC105	15	1513.7	***15***	***1513.7***
R109	13	1148.7	***13***	***1148.7***	R206	3	906.14*	5	904.4	RC106	12	1384.92*	13	1375.1
R110	12	1068.0	12	1083.2	R207	3	814.84*	4	813.4	RC107	11	1230.48*	12	1209.3
R111	12	1048.7	***12***	***1048.7***	R208	3	708.78*	3	726.5	RC108	10	1139.82*	11	1115.5
R112	10	953.63*	11	981.8	R209	4	901.88*	5	894.2	RC201	9	1261.8	7	1315.2
C101	10	827.3	***10***	***827.3***	R210	3	939.34*	5	950.8	RC202	4	1165.57*	7	1103.4
C102	10	827.3	***10***	***827.3***	R211	2	794.46*	3	844.8	RC203	3	1013.99*	4	959.1
C103	10	826.3	***10***	***826.3***	C201	3	589.1	***3***	***589.1***	RC204	3	798.41*	4	810.2
C104	10	822.9	***10***	***822.9***	C202	3	589.1	***3***	***589.1***	RC205	5	1286.70*	7	1177.5
C105	10	827.3	***10***	***827.3***	C203	3	588.7	***3***	***588.7***	RC206	3	1146.32*	6	1110.4
C106	10	827.3	***10***	***827.3***	C204	3	590.60*	3	596.7	RC207	3	1061.14*	6	997.6
C107	10	827.3	***10***	***827.3***	C205	3	586.4	***3***	***586.4***	RC208	3	828.14*	5	838.1

Table 1 Comparisons of results from the literature. Bold means an optimal solution was reached. The symbol (*) when the optimal solution is not known, and that result comes from heuristic methods, using double precision. NV – Number of Vehicles, TD –Travel distance.

The same Hybrid GA was tested using real arithmetic (double precision). The results are summarized in Table 2 . The results are presented in terms of arithmetic averages for each Solomon class (R1, C1, RC1, R2, C2, and RC2).

It's possible to see that the algorithm proposed continues to be very competitive in terms of total travel distance.

	Taillard et al. [1]		Rousseau et al.[7]		Hybrid GA (real)		Hybrid GA (trunc)	
Class	TD	NV	TD	NV	TD	NV	TD	NV
R1	1209,35	12,17	1201,10	12,83	**1200,13**	13,33	1180,07	13,33
C1	828,38	10,00	828,38	10,00	**828,38**	10,00	826,70	10,00
RC1	1389,22	11,50	1370,26	12,50	**1344,89**	13,00	1340,14	12,88
R2	980,27	2,82	966,94	3,18	**910,17**	4,64	904,24	4,82
C2	**589,86**	3,00	594,01	3,00	590,62	3,00	588,45	3,00
RC2	1117,44	3,38	1113,29	3,75	**1032,88**	6,00	1038,94	5,75

Table 2 Arithmetic averages of travel distance (TD) and number of vehicles (NV) from previous works and the Hybrid GA proposed. The results from the Hybrid GA are shown in double precision (real) and 1 decimal truncated (trunc).

4. Conclusions

This work presents a contribution to overcome two main weaknesses in the VRPTW literature. Firstly, the results reported here have improved the literature benchmarks in terms of total travel distance minimization. Secondly, the behavior of heuristic methods was compared with exact methods in terms of global optimum found, robustness and computational effort, using the same assumptions for the precision of the calculation and the same objective function defined in almost all papers reporting solutions and results by exact methods.

5. References

[1] E. Taillard, P. Badeau, M. Gendreau, F. Geurtin, and J.Y. Potvin, "A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows," *Transportation Science*,31,170-186, 1997.

[2] P. Shaw, "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems," *In Principles and Practice of Constraint Programming* - CP98, Pisa, Italy, 1998.

[3] J. Larsen. "Parellellization of the vehicle routing problem with time windows*." Ph.D. Thesis IMM-PHD-1999-62*, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.

[4] Y. Rochat and E.D. Taillard, "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing*," Journal of Heuristics* 1, 147-167, 1995.

[5] J. Homberger and H. Gehring, "Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows," INFOR, VOL. 37, 297-318, 1999.

[6] J. Berger, M. Barkaoui and O. Bräysy, "A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows," Working paper, Defense Research Establishment Valcartier, Canada, 2001.

[7] L. M. Rousseau, M. Gendreau and G. Pesant. "Using Constraint-based Operators with Variable Neighborhood Search to Solve the Vehicle Routing Problem with Time Windows" Presented at the CP-AI-OR'99 Workshop, February 25.-26., University of Ferrara, Italy, 1999.

[8] K,C. Tan, L.H. Lee, Q.L. Zhu and K. Ou, "Heuristic methods for vehicle routing problem with time windows" *Artificial Intelligence in Engineering,* 15, 281-295, 2001.

[9] A. S., Fraser. "Simulation of Genetic System by Automatic Digital Computers". Australian Journal of Biological Science, Vol. 10, 484-491, 1957.

[10] G. E. P. Box. "Evolutionary Operation: a Method of increasing industrial productivity". Applied Statistics, Vol. 6 , 81 -101, 1957.

[11] L. J. Forgel, A.J. Owens e M.J. Walsh. "Artificial Intelligence through Simulated Evolution". New York: Wiley Publishing. 1966.

[12] I. Rochenberg. “Evolutions strategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution”. Frommann-Holzboog, Stuttgart. 1973.

[13] J. H. Holland. “Adaptation in Natural and Artificial System”. Ann Arbor, Michigan: The University of Michigan Press, 1975.

[14] C. Darwin. “On The Origin of Species”. 1st edition, Harvard University Press, MA. 1859.

[15] M. M. Solomon. “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints”, Operations Research 35 (2), 254-265, 1987.

[16] S. R. Thangiah, I. H. Osman, T. Sun, Hybrid Genetic Algorithm Simulated Annealing and Tabu Search Methods for Vehicle Routing Problem with Time Windows, Technical Report 27, Computer Science Department, Slippery Rock University, 1994.