A Memetic Algorithm for the Vehicle Routing Problem with Time Windows

Jean Berger and Mohamed Barkaoui
Defence Research and Development Canada - Valcartier, Decision Support System Section
2459 Pie-XI Blvd. North, Val-Bélair, PQ, Canada, G3J 1X5
email: jean.berger@drdc-rddc.gc.ca, barkaoui@oricom.ca

Abstract

Serial and parallel versions of a new memetic algorithm to address the Vehicle Routing Problem with Time Windows are presented. The underlying approach involves parallel co-evolution of two populations. The first population evolves individuals to minimize total traveled distance while the second focuses on minimizing temporal constraint violation to generate a feasible solution. New genetic operators have been designed to incorporate key concepts emerging from recent promising techniques such as insertion heuristics, large neighborhood search and ant colony systems to further diversify and intensify the search. The parallel version of the method is based on a master-slave message-passing paradigm. The master controls the execution of the algorithm, synchronizes atomic genetic operations and handles parent selection while the slaves concurrently execute genetic operations. Results from a computational experiment show that the serial version of the proposed technique matches or outperforms the best-known heuristic routing procedures, providing six new best-known solutions. In comparison, the method proved to be fast, cost-effective and highly competitive. Alternatively, simulation results obtained for the parallel version show a significant improvement over the serial algorithm, matching or even improving solution quality. The parallel algorithm shows a speed-up of five in computing solution having near similar quality.

1. **Introduction**

Vehicle routing problems are well known combinatorial optimization problems with considerable economic significance. The Vehicle Routing Problem with Time Windows (VRPTW) has received a lot of attention in the literature recently. This is mostly due to the wide applicability of time window constraints in real-world cases. In VRPTW, customers with known demands are serviced by a homogeneous fleet of vehicles of limited capacity. Routes are assumed to start and end at the central depot. Each customer provides a time interval during which a particular task must be completed such as loading/unloading the vehicle. It is worth noting that the time window requirement does not prevent any vehicle from arriving before the allowed start of service at a customer location. The objective is to minimize the number of tours or routes, and then for the same number of tours, to minimize the total traveled distance, such that each customer is serviced within its time window and the total load on any vehicle associated with a given route does not exceed the vehicle capacity.

A variety of algorithms including exact methods and efficient heuristics have already been proposed for VRPTW. For excellent surveys on exact, heuristic and metaheuristic methods, see [Desrosiers et al., 1995], [Cordeau et al., 2001] and [Bräysy and Gendreau, 2001a and 2001b] respectively. In particular, evolutionary and genetic algorithms have been among the most suitable approaches to tackle the VRPTW, and are of particular interest to us.

Genetic algorithms [Holland, 1975]; [De Jong, 1975] and [Goldberg, 1989] are adaptive heuristic search methods that mimic evolution through natural selection. They work by combining selection, recombination and mutation operations. The selection pressure drives the population toward better solutions while recombination uses genes of selected parents to produce offspring that will form the next generation. Mutation is used to escape from local minima.

Routing techniques based on genetic algorithms to solve VRPTW emerge from the work of [Blanton and Wainwright,1993], [Thangiah, 1995a and 1995b], [Thangiah et al., 1995], [Potvin and Bengio, 1996], [Berger et al., 1998, 1999, 2000] and [Tan et al., 2001]. Alternate methods using evolutionary metaheuristics have been proposed by [Homberger and Gehring, 1999], [Gehring and Homberger, 1999 and 2001], and [Bräysy et al., 2000]. Other recent studies on various metaheuristics for VRPTW can be found in [Rochat and Taillard, 1995], [Taillard et al., 1997], [Chiang and Russell, 1997], [Cordeau et al., 2001] (tabu searches), [Gambardella et al., 1999] (ant colony optimization), and [Liu and Shen, 1999].

Proposed metaheuristics so far show significant variability in performance. They often require considerable computational effort and therefore fail to convincingly provide a single robust and successful technique. Recently, a new memetic or parallel hybrid genetic algorithm (PHGA) for the VRPTW has been successfully developed [Berger, Barkaoui and Bräysy, 2002]. Memetic Algorithms is a population-based approach for heuristic search in optimization problems [Moscato, 1989]. They have shown that they are orders of magnitude faster than traditional genetic algorithms for some problem domains. Basically, they combine local search heuristics with mutation and crossover operators. For this reason, some researchers have viewed them as hybrid genetic algorithms or genetic local search. Our approach is based on a new concept that combines constrained parallel co-evolution of two populations and partial temporal constraint relaxation to improve solution quality. The first population evolves individuals to minimize the total traveled distance while the second focuses on minimizing temporal constraint violation in trying to generate a feasible solution. Imposing a constant number of tours for each solution of a given population, temporal constraint relaxation allows escaping local minima while progressively moving toward a better solution. Populations interact with one another whenever a new feasible solution emerges, reducing by one the number of tours imposed on future solutions. New genetic operators have been designed to maximize the number of customers served within their time intervals first, and then temporal constraint relaxation is used to insert remaining unvisited customers. Key principles and variants emerging from recent promising techniques are also captured to further diversify and intensify the search. As a result, even though the algorithm is more robust, efficient, stable and highly competitive, prohibitive computational cost of key genetic operators and overall run-time still remain a sensitive issue to be satisfactorily addressed.

The main contribution of this paper is to further improve the PHGA technique by developing an efficient parallel implementation based upon a master-slave message-passing paradigm (networked parallel computing) in order to significantly reduce run-time. The master processing element controls the execution of the algorithm, synchronizes atomic genetic operations and handles the parent selection process while the slave processing elements concurrently execute reproduction and mutation operators. Current and new genetic operators have been designed and revisited to reduce processor starvation over each generation.

The paper is outlined as follows. Section 2 introduces the basic concepts of the proposed parallel hybrid genetic algorithm. The basic principles and features of the algorithm are first described. Details on the parallel implementation of the algorithm are then given. Section 3 presents the results of a computational experiment to assess the value of the proposed approach and reports a comparative performance analysis to alternate methods. Finally, a summary is presented in Section 4.

2. **Parallel Hybrid Genetic Algorithm**

2.1 ***General Description***

The proposed algorithm relies upon constrained parallel co-evolution and partial constraint relaxation. Two populations Pop_1 and Pop_2, primarily formed of non-feasible solution individuals, are evolving concurrently, each with their own objective functions. Pop_1 contains at least one feasible solution and is used to minimize total traveled distance while Pop_2 focuses on minimizing constraint violation. Constrained to a fixed number of tours over the same population, solution individuals differ by exactly one route across both populations. Parallel evolution is interrupted whenever a new best feasible solution is obtained. Populations are then reinitialized and co-evolution resumed, while decreasing the number of routes associated with solution individuals by one. The number of tours imposed on solution individuals in Pop_1 and Pop_2 are $R_{\min}$ and $R_{\min} - 1$, respectively. $R_{\min}$ refers to the number of routes found in the best feasible solution obtained so far. As a new feasible solution emerges from Pop_2, population Pop_1 is replaced by Pop_2, $R_{\min}$ is updated and, Pop_2 is reinitialized with the revised number of tours ($R_{\min}$ -1), using the RSS_M mutation operator. In addition, a post-processing procedure (RC_M) aimed at reordering customers, is applied to further improve the new best solution. The evolutionary process is repeated until a predefined stopping condition is met.

The proposed approach uses a steady-state genetic algorithm that involves overlapping populations. At first, new individuals are generated and added to the current population Pop_p. The process continues until the overlapping population outnumbers the initial population by n_p. Then, the n_p worst individuals are eliminated to maintain population size using the following individual evaluation:

$$Eval_i = E_i + CV_i, \tag{1}$$

where

$$E_i = r_i - r_m + \frac{d_i}{\max\{d_m, d_i\}}, \tag{2}$$

$$CV_i = \sum_{j=1}^{n} a_j \max\{0, b_j^i - l_j\} + b\ Viol_i \tag{3}$$

r_i = number of routes in individual i,
r_m = lower bound for number of routes (ratio of total demand over vehicle capacity),
d_i = total traveled distance related to individual i,
d_m = average traveled distance over the individuals forming the initial population,

n	= number of customers,
a_j	= penalty associated with temporal constraint violation j,
b_j^i	= scheduled time to visit customer j in individual i,
l_j	= latest time to visit customer j,
b	= penalty associated with number of violated temporal constraints,
$Viol_i$	= number of temporal constraints violated in individual i.

The proposed evaluation expression indicates that better individuals generally (but not necessarily) include fewer routes, and smaller total traveled distance, while satisfying temporal constraints. The general algorithm is as follows:

```
Initialization
Repeat
  p=1
  Repeat   {evolve population Pop_p - new generation}
    For j =1..n_p do
      Select two parents from Pop_p
      Generate a new solution S_j using recombination and mutation operators associated with Pop_p
      Add S_j to Pop_p
    end for
    Remove the n_p worst individuals from Pop_p using the evaluation function (1).
    p=p+1
  Until (all populations Pop_p have been considered)
  if (Pop_2 includes a new best feasible solution) then
      {eliminate all Pop_1 individuals}
      Set Pop_1 = Pop_2
      Modify Pop_2 solutions by applying RSS_M {reduces number of routes by one}.
  endif
    Apply RC_M on the best solution {customer reordering}
Until(convergence criteria or max number of generations)
```

Feasible solutions for initial populations are first generated using a sequential insertion heuristic in which customers are inserted in random order at randomly chosen insertion positions within routes. The initialization procedure then proceeds as follows:

```
For  p = 1..2 do  {revisit Pop_1 and Pop_2}
  For  j = 1..n_p do
        Generate a new solution S_j using the EE_M mutator (defined in Section 2.3.2)
        Add S_j in Pop_p
  end for
  Remove the n_p worst individuals from Pop_p using Eval_i
end for
Determine R_min, the minimum number of tours associated with a feasible solution in Pop_1 or Pop_2.
Replicate (if needed) best feasible solution (R_min routes) in Pop_1.
Replace Pop_1 individuals with R_min-route solutions using the procedure RI(R_min).
Replace Pop_2 members with R_min-1 route solutions using the procedure RI(R_min-1).
```

RI(r) is a re-initialization procedure creating an r-route solution. It first generates r one-customer routes formed from randomly selected customers. Then, it uses the insertion procedure proposed by Liu and Shen [Liu and Shen, 1999] to insert as many customers as possible without violating time window constraints. Accordingly, customer route-neighborhoods are repeatedly examined for insertion. The next customer for insertion is selected by maximizing a so-called regret cost function that accounts for multiple route insertion opportunities:

$$\text{regret cost} = \sum_{r \in RN(i)} \{c_i(r) - c_i(r^*)\}, \tag{4}$$

where

$RN(i)$ = route-neighborhood of customer *i*,

$c_i(r)$ = minimum insertion cost of customer *i* within route *r*,

$c_i(r^*)$ = minimum insertion cost of customer *i* over its route-neighborhood.

Remaining unvisited customers (if any) are then inserted in the *r*-tour solution maximizing an extended insertion regret cost function, in which $c_i(r)$ includes an additional contribution reflecting temporal constraint violations:

$$\sum_{j=1}^{n_r} a_j \max\{0, b_j - l_j\} + b\ Viol_r \tag{5}$$

in which

n_r = current number of customers in route *r*,

a_j = penalty associated with temporal constraint violation *j*,

b = penalty associated with the number of violated temporal constraints,

b_j = scheduled time to visit customer *j* in route *r*,

l_j = latest time to visit customer *j*,

$Viol_r$ = current number of temporal constraints violated in route *r*.

2.2 ***Selection***

The selection process consists of choosing two individuals (parent solutions) within the population for mating purposes. The selection procedure is stochastic and biased toward the best solutions using a roulette-wheel scheme [Goldberg, 1989]. In this scheme, the probability of selecting an individual is proportional to its fitness value. An individual fitness value is computed as follows:

Population Pop$_1$:

$$fitness_i = d_i + \sum_{j=1}^{n} a_j \max\{0, b_j^i - l_j\} + b\ Viol_i \tag{6}$$

Population Pop$_2$:

$$fitness_i = \sum_{j=1}^{n} a_j \max\{0, b_j^i - l_j\} + b\ Viol_i \tag{7}$$

The notations are the same as in equations 1–3. Better individuals generally (but not necessarily) tend to include short total traveled distance in Pop$_1$ and satisfy as many temporal constraints as possible in Pop$_2$.

2.3 ***Genetic Operators***

The proposed genetic operators mostly rely on two basic principles. First, for a given number of tours, an attempt is made to construct feasible solutions with as many customer visits as possible. Second, the remaining customers are inserted into existing routes through temporal constraint relaxation. Constraint violation is used to restrict the total number of routes to a constant value. The proposed genetic operators incorporate some key features of the best heuristic routing techniques such as Solomon's insertions heuristic I1 [Solomon, 1987] large neighborhood search [Shaw, 1998] and the route neighborhood-based two-stage metaheuristic (RNETS) [Liu and Shen, 1999]. Details on the recombination and mutation operators used are given in the next sections.

2.3.1 ***Recombination***

The insertion-based IB_X(k) recombination operator creates an offspring by combining, one at a time, k routes of parent solution P_1 with a subset of customers, formed by nearest-neighbor routes $\{r_2\}$ in parent solution P_2. The k routes ($\{r_1\}$) are selected either randomly, with a probability proportional to the relative number of customers or based on the average distance separating consecutive customers on the routes. A removal procedure is first carried out to remove from r_1 some key customers believed to be most suitably relocated within some alternate routes. More precisely, the stochastic customer removal procedure removes either randomly some customers, customers rather distant from their successors, or customers with waiting times. Then, a modified insertion heuristic of [Solomon, 1987] is applied to build a feasible route, considering the modified partial route r_1 as the initial solution and unrouted customers in routes r_2 for insertion. The I1 standard insertion heuristic of [Solomon, 1987] is coupled to a random customer selection procedure, to choose the next candidate customer to be routed. Once the construction of the child route is completed, and reinsertion is no longer possible, a new route construction cycle is initiated. The overall process is repeated for the k routes selected from P_1. Finally, if necessary, the child inherits the remaining "diminished" routes of P_1. If unrouted customers still remain, additional routes are built using a nearest-neighbor procedure of [Solomon, 1987]. The whole process is then iterated once more to generate a second child by interchanging the roles of P_1 and P_2. Further details of the operator may be found in [Berger and Barkaoui, 2000]. In order to keep the number of routes of a child solution identical to its parents, a post-processing procedure is applied. If the solution has a larger number of routes than expected, the RSS_M (Section 2.3.2) procedure is used repeatedly to reduce the number of routes. Conversely, for solutions having a smaller number of routes, new feasible routes are constructed repeatedly by breaking the most populated route in two until the targeted number of routes is obtained.

2.3.2 ***Mutation***

A suite of five mutation operators is proposed, namely LNSB_M, EE_M, IEE_M, RS_M, RSS_M and RC_M. The LNSB_M (Large Neighborhood Search -based) mutation operator relies on the concepts of the Large Neighborhood Search (LNS) proposed by [Shaw, 1998]. The LNS consists of exploring the search space by repeatedly removing related customers and reinserting them using constraint-based tree search (constraint programming). As in [Shaw, 1998], a set of related customers is first removed. In addition, LNSB_M removes customers violating temporal constraints from their routes. The proposed customer re-insertion method differs from the procedure proposed by [Shaw, 1998] in two respects, namely, the insertion cost function used, and the order in which customers are considered for insertion (variable ordering scheme) during the branch-and-bound search process. Unvisited customers (if any) are then reinserted using the same customer re-insertion method while relaxing temporal constraints. Insertion cost is defined by the sum of key contributions referring respectively to increased traveled distance, and delayed service time, as specified in Solomon's procedure I1 (c_{11}+c_{12}), as well as to constraint violation (equation (5)). Concerning customer visit ordering, customers ($\{c\}$) are sorted (*CustOrd*) according to a composite ranking. The ranking is defined as an additive combination of two separate rankings, previously achieved over best insertion costs ($Rank_{Cost}(c)$) on the one hand, and number of feasible insertion positions ($Rank_{|Pos|}$(c)) on the other hand:

$$CustOrd \leftarrow Sort\{c\} \;\; (\; Rank_{Cost}(c) + Rank_{|Pos|}(c) \;) \tag{8}$$

The smaller the insertion cost (short total distance, traveled time) and the number of positions (opportunities), the better (smaller) the ranking. The next customer to be visited within the search process is selected according to the following expression

$$customer \leftarrow CustOrd[INTEGER(L \; rand^{D})] \tag{9}$$

in which

L	=	current number of customers to be inserted,
$rand$	=	real number over the interval [0,1] (uniform random number generator),
D	=	parameter controlling determinism. If D=1 then selection is purely random (default: D=15).

Once a customer is selected, tree search is carried out over its different insertion positions as specified in [Shaw, 1998]. However, the search tree expansion is initiated using a non-constant discrepancy factor, selected randomly over the set {1,2,3}.

The EE_M (edge exchange) and RS_M (repair solution) mutators focus on inter-route improvement. EE_M uses the l-interchange mechanism of [Osman, 1993], performing reinsertions of customer sets over two neighboring routes. Here, route neighborhood is determined by route centroid proximity. Customer exchanges take place as soon as the solution improves, i.e., we use the first-accept strategy. Assuming the notation (x,y) to describe the different sizes of customer sets (l) issued from the neighboring routes, the current operator explores values running over the range (x=1, y=0,1,2). The RS_M mutation operator focuses on exchanges involving one illegal customer. Each illegal customer in a route is exchanged with an alternate legal one or two-customer sequence in order to generate a new set of customers with either violated or non-violated temporal constraints. The objective is to further explore the solution space (diversity) while possibly improving quality. The IEE_M mutation operator is similar to EE_M except for customer reordering in which customer permutations are restricted to the same route.

The RSS_M (reinsert shortest Solomon) mutation operator eliminates the shortest route (smallest number of customers) of the solution, decreasing by one the total number of routes. Customers from the shortest route are first removed. Then, following an iterative process, unvisited customers are re-inserted into existing routes using the insertion procedure proposed by [Liu and Shen, 1999] in which the regret cost function (equation (4)) has been extended to include a constraint violation contribution (equation (5)). The entire iterative process is repeated over *I* different sets (e.g. *I*=20) of randomly generated parameter values.

The RC_M (reorder customers) mutation operator is an intensification procedure that tries to reduce the total distance of feasible solutions by reordering customers within a route. The procedure consists of repeatedly reconstructing a new tour using the sequential insertion procedure I1 of Solomon [Solomon, 1987] over *I* different sets (e.g. *I*=2) of randomly generated parameter values.

2.4 ***Parallel Implementation***

The parallel implementation consists in using the parallel virtual machine PVM [Geist and al., 1994] software based on a master-slave message-passing paradigm that enables a collection of heterogeneous computers to be used as a single coherent and flexible computational resource supporting concurrency. In the current version, the master processing element supervises and controls the execution of the algorithm, handles the parent selection process, population replacement and the emergence of a new feasible solution, selects and synchronizes (activation and completion) atomic genetic operations, finalizes construction of new generations while the slave processing elements supervised by the master concurrently execute reproduction and mutation operators. Current and new genetic operators have been revisited or configured to reduce processor starvation over each generation. In a nutshell, the master component of the networked parallel implementation presents similar characteristics to the sequential version of the algorithm except that genetic operators are concurrently executed as atomic operations on multiple processors. The algorithm has been implemented in C++, using a modified version of the GAlib genetic algorithm library of [Wall, 1995], on a 19-computer cluster architecture: Linux platform environment, 19 Athlon 1.2 GHz processors (a master and 18 slaves) with 768M of RAM with a 100M/sec communication bandwidth, and a 3 module switch including 41 ports. The overall run-time for the algorithm has been limited to two minutes.

3. **Computational Experiment**

A computational experiment has been conducted to compare the performance of the parallel version of the proposed algorithm with some of the best techniques designed recently for VRPTW. The algorithm has been tested with 56 VRPTW benchmark problems of Solomon [Solomon, 1987]. Each problem involves 100 customers, randomly distributed over a geographical area. The travel time separating two customers corresponds to their relative Euclidean distance. Customer locations for a problem instance are either generated randomly using a uniform distribution (problem data sets R1 and R2), clustered (problem data sets C1 and C2) or mixed, combining randomly distributed and clustered customers (problem data sets RC1 and RC2). The experiment consisted in performing three simulation runs for each problem instance in a given data set.

3.1 *Configuration*

Parameter setting and simulation configuration for the investigated algorithm are specified as follows:

Populations: 2 (Pop$_1$ and Pop$_2$)
Run-time: 120 seconds
Recombination and mutation rates: 100%
Recombination operator: IB_X
Mutation operators: LNSB_M(*d*), EE_M, IEE_M, RS_M, RSS_M and RC_M

Within the LNSB_M(d) mutation operator the number of customers considered for elimination varies within the range [12, 17]. The discrepancy factor *d* is randomly chosen over {1,2,3}. In fitness, evaluation and insertion cost functions:

$$a_j = 100, \ \forall j$$

$$a_0 = 1000$$

$$b = 100$$

The probabilities and parameter values for the proposed genetic operators are defined as follows. For all data sets except C1 and C2:

Population size: 25
Pop$_1$:
Population overlap per generation: n_1=1
LNSB_M(*d*) (100%)
EE_M (50%) + IEE_M(50%)
Pop$_2$:
Population overlap per generation n_2=17.
LNSB_M(*d*) (100%)
RS_M + EE_M + IEE_M (33%), RS_M + EE_M (33%) and RS_M + IEE_M (33%)

For data sets C1 and C2:
Population size: 25
Pop$_1$:
Population overlap per generation: n_1=15
IB_X(*k*=2) (100%) (for C2: *k*=1)
RC_M(*I*=2) (100%)
Pop$_2$:
Population overlap per generation n_2=3.
IB_X(*k*=2) (100%) (for C2: *k*=1)
RC_M(*I*=2) (100%)

Because of limited computational resources, the parameter values were determined by trying just a few intuitively selected combinations, and selecting the one that yielded the best average output. This is justified by the fact that the sensitivity of the results with respect to changes in the parameter values such as recombination and mutation rates was found to be generally quite small. Population size was chosen empirically to balance intensification and diversification.

Population overlaps (n_1 and n_2) were selected such that the sum ($n_1 + n_2$) matches the maximum number of slave processors in order to generate a new population as quickly as possible while minimizing processor starvation and, alleviate the impact of the intrinsic serial part of the parallel program. Population overlaps have been determined according to the most prominent characteristic of a data set. As we aimed at computing the minimum number of routes first, over a limited time, the idea consists in allocating a maximum number of processors to Pop$_2$, as number of tours minimization generally represents the most difficult task to achieve. But, for cases where computing the minimum number of routes does not present a major challenge, more computational resources are allocated to reduce total traveled distance. Consequently, more processing power were devoted to evolve Pop$_1$ for clustered data sets (n_1=15, n_2=3), emphasizing total traveled distance minimization. Alternatively, computational resources were primarily dedicated to Pop$_2$ evolution for alternate problem instances (n_1=1, n_2=17), stressing number of routes minimization. Variations over computed solution quality regarding population overlap parameters are negligible as far as we exploit basic data set characteristics (clustered versus non-clustered distribution). Communication and synchronization cost combined to processor starvation, following genetic operations when constructing an entire new generation, impose inevitable performance limits on parallel computation. In order to counter this adverse condition, processor starvation was minimized by configuring genetic operators to keep processing time as uniform as possible through genetic operations suite during child computation.

For a matter of run-time convenience, different parameter settings are proposed for C1 and C2, as opposed to other data sets. The parameters instantiation was inspired from a previous genetic algorithm by [Berger and Barkaoui, 2000]. In fact, this class of problem instances does not present a real challenge for most VRPTW metaheuristics as convergence generally occurs very quickly.

3.2 *Results*

The results for the six problem data sets are summarized in Tables 1-2 for some of the best reported methods for VRPTW, namely, GTA [Gambardella et al., 1999], RT [Rochat and Taillard, 1995], SW [Shaw, 1998], TB [Taillard et al., 1997], CR [Chiang and Russell, 1997], LS [Liu and Shen, 1999], HG [Homberger and Gehring, 1999], CLM [Cordeau et al., 2001] and,

Table 1: Best performance comparison among VRPTW algorithms.

Problem		**RT**	**LS**	**CR**	**TB**	**GTA**	**HG (ES1)**	**HG (ES2)**	**BB1-2**
R1	**Vehicles**	12.25	12.17	12.17	12.17	12.00	11.92	12.00	11.92
	Distance	1208.50	1249.57	1204.19	1209.35	1217.73	1228.06	1226.38	1221.1
R2	**Vehicles**	2.91	2.82	2.73	2.82	2.73	2.73	2.73	2.73
	Distance	961.72	1016.58	986.32	980.27	967	969.95	1033.58	975.43
C1	**Vehicles**	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	Distance	828.38	830.06	828.38	828.38	828.38	828.38	828.38	828.48
C2	**Vehicles**	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
	Distance	589.86	591.03	591.42	589.86	589.86	589.86	589.86	589.93
RC1	**Vehicles**	11.88	11.88	11.88	11.50	11.63	11.63	11.50	11.50
	Distance	1377.39	1412.87	1397.44	1389.22	1382.42	1392.57	1406.58	1389.89
RC2	**Vehicles**	3.38	3.25	3.25	3.38	3.25	3.25	3.25	3.25
	Distance	1119.59	1204.87	1229.54	1117.44	1129.19	1144.43	1175.98	1159.37
ALL	**Vehicles**	**415**	**412**	**411**	**410**	**407**	**406**	**406**	**405**
	Distance	57231	59317	58502	57522	57516	57876	58921	57952

BB1 [Berger, Barkaoui and Bräysy, 2002] and BB2 for the sequential and parallel versions of the parallel hybrid genetic algorithm respectively. The results are usually ranked according to a hierarchical objective function, where the number of vehicles is the primary objective and, for the same number of vehicles, the secondary objective is total traveled distance.

The best computed results are shown in Table 1. Each entry refers to the best performance obtained with a specific technique over a particular data set. The first column describes the various data sets and corresponding measures of performance defined by the average number of routes (or vehicles), and total traveled distance. The following columns refer to particular problem-solving methods. The performance of our PHGA is depicted in the last column (BB1-2). Results indicate that PHGA matches or outperforms the best-known heuristic routing procedures. The last row refers to the cumulative number of routes and traveled distance over all problem instances. The total number of tours computed over all problem data sets outperform by one the best-computed result so far, reported by Homberger and Gehring [Homberger and Gehring, 1999]. In addition, PHGA is the only method that found the minimum number of tours consistently for all problem data sets. PHGA also succeeded in improving six of the best-known solutions. Accordingly, Table 2 provides six new best-known solutions and compares them with the previous best-known solutions. Details of the new solutions can be made available by the authors.

Table 2: New best computed solutions for some Solomon problem instances

Problem	**Best-Known Solutions**		**New Best Solutions**		
	Vehicles	**Distance**	**Reference**	**Vehicles**	**Distance**
R108	9	963.99	SW	9	960.88
R110	10	1125.04	CLM	10	1119
RC105	13	1637.15	HG	13	1629.44
RC106	11	1427.13	CLM	11	1424.73
R210	3	955.39	HG	3	954.12
R211	2	910.09	HG	2	906.19

Overall, computational results for BB2 have shown a significant improvement over the sequential algorithm (BB1), mostly matching or even improving solution quality. The parallel implementation of the PHGA algorithm has generally shown a speed-up of five over the sequential version, in computing solutions having near similar quality, that is solutions presenting the same minimum number of tours and comparable traveled distance.

Even though solution quality is expected to grow with the number of processing elements, the parallel implementation involves important limitations such as processor waiting time (starvation), communication cost and the inherent ratio of the serial/parallel code attached to the proposed method. Intrinsic sequential contributions include synchronization constraints involved in the computation and completion of a new generation, fitness computation and population replacement scheme. However, the current experiment shows that run-time associated with computation of solutions presenting similar or comparable quality, steadily decreases with the number of processors. Saturation on performance is eventually expected to happen as the solution space is further explored, but this did not occur within the context of the current experiment (19-processor (maximum number) cluster environment).

4. **Conclusion**

A parallel implementation of a promising hybrid genetic algorithm (PHGA) targeted to the vehicle routing problem with time windows has been successfully developed. The implementation of the technique is based upon a master-slave message-passing paradigm (networked parallel computing) using the PVM software within a 19-computer cluster environment. The master processing element controls the execution of the algorithm, synchronizes atomic genetic operations and handles the parent selection process while the slave processing elements concurrently execute reproduction and mutation operators. Genetic operators have been designed and revisited to further reduce processor starvation over each generation. Results from a computational experiment show a significant speed-up of the method over its sequential version mostly matching or even improving solution quality. Accordingly, a new best result has been computed for the R1 Solomon's data set, while improving best-known solutions for some instances of the R2 data set.

Future work will focus on comparative average performance analysis while extending computational experiments to larger VRPTW problem instances to better characterize the limitations of the approach. Variants of the parallel algorithm, addressing design limitations with respect to processor starvation, communication cost and current serial/parallel code ratio, might be investigated as well.

5. **References**

Berger, J., M. Salois and R. Begin (1998), “A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows”, *Lecture Notes in Artificial Intelligence* 1418, AI’98, Advances in Artificial Intelligence, Vancouver, Canada, 114–127.

Berger J., M. Sassi and M. Salois (1999), “A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows and Itinerary Constraints”, In *Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, USA, 44–51.

Berger, J. and M. Barkaoui (2000), “An Improved Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows”, International ICSC Symposium on Computational Intelligence, part of the *International ICSC Congress on Intelligent Systems and Applications (ISA'2000*), University of Wollongong, Wollongong, Australia.

Berger J., M. Barkaoui and O. Bräysy (2002), "A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows", to be published.

Blanton, J.L. and R.L. Wainwright (1993), “Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms”, In *Proceedings of the 5th International Conference on Genetic Algorithms*, S. Forrest (editor), 452–459 Morgan Kaufmann, San Francisco.

Bräysy, O., J. Berger and M. Barkaoui (2000), “A New Hybrid Evolutionary Algorithm for the Vehicle Routing Problem with Time Windows”, Presented in Route 2000 Workshop, Skodsborg, Denmark.

Bräysy, O. and M. Gendreau (2001a), “Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms”, Working Paper, SINTEF Applied Mathematics, Department of Optimisation, Norway.

Bräysy, O. and M. Gendreau (2001b), “Vehicle Routing Problem with Time Windows, Part II: Metaheuristics”, Working Paper, SINTEF Applied Mathematics, Department of Optimisation, Norway.

Chiang, W.-C. and R.A. Russell (1997), “A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows”, *INFORMS Journal on Computing* 9, 417–430.

Cordeau, J.F., G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis (2001), “The VRP with Time Windows”, To appear in *The Vehicle Routing Problem*, Chapter 7, P. Toth and D. Vigo (eds), SIAM Monographs on Discrete Mathematics and Applications.

Cordeau, J.-F., G. Laporte and A. Mercier (2001), “A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows”, *Journal of the Operational Research Society* 52, 928–936.

Jong De, K. A. (1975), An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Dissertation, University of Michigan, U.S.A.

Desrosiers, J., Y. Dumas, M.M. Solomon and F. Soumis (1995), “Time Constrained Routing and Scheduling”, In *Handbooks in Operations Research and Management Science*, *Vol. 8. Network Routing*, M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (eds), North-Holland, Amsterdam, 35–139.

Gambardella, L. M., E. Taillard, and G. Agazzi (1999), “MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows”, In *New Ideas in Optimization*, D. Corne, M. Dorigo and F. Glover (eds), 63–76, McGraw-Hill, London

Gehring, H. and J. Homberger (1999), “A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows”, *Proceedings of EUROGEN99 - Short Course on Evolutionary Algorithms in Engineering and Computer Science, Reports of the Department of Mathematical Information Technology Series*. No. A 2/1999, University of Jyväskylä, Finland, K. Miettinen, M. Mäkelä and J. Toivanen (eds), 57–64.

Gehring, H. and J. Homberger (2001), “Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows”, *Asia-Pacific Journal of Operational Research* 18, 35–47.

Geist, Al et al. (1994), A Users' Guide and Tutorial for Networked Parallel Computing, MIT Press Scientific and Engineering Computation, Janusz Kowalik Editor, Massachusetts Institute of Technology, Boston, (http://www.netlib.org/pvm3/book/pvm-book.html).

Goldberg, D.E (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York.

Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Homberger, J. and H. Gehring (1999), “Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows”, *INFOR* 37, 297–318.

Liu, F.-H. and S.-Y. Shen (1999), “A Route-Neighborhood-based Metaheuristic for Vehicle Routing Problem with Time Windows”, *European Journal of Operational Research* 118, 485–504.

P. Moscato, P. (1989). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, *Concurrent Computation Program, C3P Report 826*, Caltech, Pasadena, U.S.A.

Osman, I.H.(1993), “Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem”, *Annals of Operations Research* 41, 421–451.

Potvin, J-Y. and S. Bengio (1996), “The Vehicle Routing Problem with Time Windows Part II: Genetic Search”, *INFORMS Journal on Computing* 8, 165–172.

Rochat, Y. and E. Taillard (1995), “Probabilistic Diversification and Intensification in Local Search for Vehicle Routing”, *Journal of Heuristics* 1, 147–167.

Solomon, M.M. (1987), “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints”, *Operations Research* 35, 254–265.

Shaw, P. (1998), “Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems”, In *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science*, M. Maher and J.-F. Puget.(eds.), 417–431, Springer-Verlag, New York.

Taillard, É., P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin (1997), “A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows”, *Transportation Science* 31, 170–186.

Tan, K.C., L.H. Lee and K. Ou (2001), “Hybrid Genetic Algorithms in Solving Vehicle Routing Problems with Time Window Constraints”, *Asia-Pacific Journal of Operational Research* 18, 121–130.

Thangiah, S.R., I.H. Osman, R. Vinayagamoorthy and T. Sun (1995), “Algorithms for the Vehicle Routing Problems with Time Deadlines”, *American Journal of Mathematical and Management Sciences* 13, 323–355.

Thangiah, S. (1995a), “Vehicle Routing with Time Windows Using Genetic Algorithms”, In *Application Handbook of Genetic Algorithms*: *New Frontiers, Volume II*, 253–277, L. Chambers (editor), CRC Press, Boca Raton.

Thangiah, S.R. (1995b), “An Adaptive Clustering Method using a Geometric Shape for Vehicle Routing Problems with Time Windows”, In *Proceedings of the 6th International Conference on Genetic Algorithms*, L.J. Eshelman (editor), 536–543 Morgan Kaufmann, San Francisco.

Wall, M. (1995), *GAlib - A C++ Genetic Algorithms Library, version 2.4*. (http://lancet.mit.edu/galib-2.4/), MIT, Boston.