Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 31 (2004) 2037–2053
computers & operations research
www.elsevier.com/locate/dsw

A parallel hybrid genetic algorithm for the vehicle routing problem with time windows

Jean Berger*, Mohamed Barkaoui

Defence Research and Development Canada Valcartier, Decision Support Systems Section, 2459 Pie-XI Blvd. North, Val-Bélair, PQ, Canada G3J 1X5

Abstract

A parallel version of a new hybrid genetic algorithm for the vehicle routing problem with time windows is presented. The route-directed hybrid genetic approach is based upon the simultaneous evolution of two populations of solutions focusing on separate objectives subject to temporal constraint relaxation. While the first population evolves individuals to minimize total traveled distance the second aims at minimizing temporal constraint violation to generate a feasible solution. Genetic operators have been designed to capture key concepts from successful routing techniques to further enhance search diversification and intensification. A master–slave message-passing paradigm characterizes the parallel procedure. The master component controls the execution of the algorithm, coordinates genetic operations and handles parent selection while the slave elements concurrently execute reproduction and mutation operators. Providing additional speed-up, the parallel algorithm further expands on its sequential counterpart, matching or even improving solution quality. Computational results show the proposed technique to be very competitive with the best-known heuristic routing procedures providing some new best-known solutions.

Keywords: Genetic algorithms; Metaheuristics; Vehicle routing

1. Introduction

Vehicle routing problems are well-known combinatorial optimization problems with considerable economic significance. In vehicle routing problem with time windows VRPTW [1], customers with known demands are serviced by a homogeneous fleet of limited capacity vehicles. 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. The objective is to minimize the

* Corresponding author. Tel.: +1-418-844-4000; fax: +1-418-844-4538.
E-mail address: jean.berger@drdc-rddc.gc.ca (J. Berger).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0305-0548(03)00163-1

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 surveys on exact, heuristic and metaheuristic methods, see Cordeau et al. [1], Desrosiers et al. [2] and Bräysy and Gendreau [3,4], 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 [5–7] 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.

Blanton and Wainwright [8] were the first to apply a genetic algorithm to VRPTW. They hybridized a genetic algorithm with a greedy heuristic. Under this scheme, the genetic algorithm searches for a good ordering of customers, while the construction of the feasible solution is handled by the greedy heuristic. Thangiah [9,10] uses a genetic algorithm to find good clusters of customers within a "cluster first, route second" problem-solving strategy. Thangiah et al. [11] test the same approach to solve vehicle routing problems with time deadlines.

In the algorithm proposed by Potvin and Bengio [12], new offspring are created by connecting two route segments from two parent solutions or by replacing the route of the second parent solution by the route of the first parent solution. Mutation is then used to reduce the number of routes and to locally optimize the solution. Berger et al. [13] present a hybrid genetic algorithm based on removing certain customers from their routes and then rescheduling them with well-known route-construction heuristics. The mutation operators are primarily aimed at reducing solution number of routes through customer rescheduling and local reordering. Bräysy continues the study of Berger et al. [13] by performing a sensitivity analysis, and by creating new crossover and mutation operators. Also Berger et al. [14] and Berger and Barkaoui [15] have further continued the research direction started in Berger et al. [13].

Homberger and Gehring [16] propose two evolutionary metaheuristics based on the class of evolutionary algorithms called evolution strategies and three well-known route improvement procedures, namely, Or-opt [17] λ-interchanges [18] and 2-opt* [19]. Gehring and Homberger [20,21] use a similar approach with parallel tabu search implementation. Bräysy et al. [22] hybridize a genetic algorithm with an evolutionary algorithm consisting of several route construction and improvement heuristics. The recent genetic algorithm by Tan et al. [23] is based on Solomon's insertion heuristic [24], λ-interchanges and the well-known PMX-crossover operator. Other recent studies on various metaheuristics for VRPTW can be found in Rochat and Taillard [25], Taillard et al. [26], Chiang and Russell [27], Cordeau et al. (tabu searches) [28], Gambardella et al. [29] (ant colony optimization), and Liu and Shen [30].

Noticeable from recent surveys [3,4] most metaheuristics proposed so far present some variability in average and best performances. Besides, reported average performance results [4] can hardly support any claims acknowledging a dominating heuristic over the others. Accordingly, no single method consistently matched the best-known minimum number of tours over all problem instances examined. Moreover, computational cost represents a sensitive issue to be satisfactorily addressed

as well. As a result, a more robust, cost effective and stable (minimal variance in computed number of tours) algorithm still remains elusive. The main contribution of this paper is an attempt to design such a new technique. A parallel version of a route-directed hybrid genetic algorithm (RHGA) for the VRPTW is proposed. Lending itself to some form of parallelism due to the very nature of the underlying genetic algorithm, the approach is based on the simultaneous evolution of two populations of solutions 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 fixed number of tours to solution members of a given population, temporal constraint relaxation facilitates escaping local minima while progressively moving toward a better solution. Populations interact with one another whenever a new feasible solution emerges, reducing gradually the number of tours imposed on future solutions. Genetic operators have been designed to capture key concepts from successful routing techniques to further diversify and intensify the exploration of the solution space. Aimed at providing additional speed-up, the parallel algorithm is based upon a master–slave message-passing paradigm. 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.

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. Then, the selection scheme, recombination and mutation operators are presented. The combination of concepts derived from well-known heuristics such as large-neighborhood search [31] and route neighborhood-based two-stage metaheuristic [30] are briefly outlined. Details are then provided on the parallel version of the algorithm. 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, some conclusions and future research directions are presented in Section 4.

2. Hybrid genetic approach

2.1. General description

Based on the general principles of natural selection, the proposed algorithm somewhat differs from the typical genetic paradigm in which explicit solution encoding is used for problem representation. Genetic operators are merely applied to a population of solutions rather than a population of encoded solutions (chromosomes). We refer to these solutions as solution individuals.

Our approach is rooted in the concepts of simultaneous 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. 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 (duplication), $R_{\min}$ is updated and, Pop_2 is mutated considering a revised number of tours ($R_{\min}-1$), using the RSR_M mutation

operator. In addition, a post-processing procedure (RC_M) aimed at reordering customers is applied to further improve the new best solution. RSR_M and RC_M mutation operators are both described in Section 2.3.2. 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 $\mathrm{Pop_p}$. 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 function:

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

where

$$E_i = r_i - r_m + \gamma d_i, \tag{2}$$

$$CV_i = \sum_{c=0}^{n} \alpha_c \max\{0, b_c^i - l_c\} + \beta\, Viol_i, \tag{3}$$

where E_i is the basic evaluation of individual (solution) i, r_i the number of routes in individual i, r_m the lower bound for number of routes (ratio of total demand over vehicle capacity), d_i the total traveled distance related to individual i, γ the relative user-defined weight parameter controlling the distance contribution. The value of γ is selected to privilege solutions having a smaller number of routes (e.g. $\gamma = 1/d_m$, where d_m refers to the maximum traveled distance over the individuals forming the initial population), CV_i the temporal constraint violation associated to individual i, n the number of customers ($c = 0$ refers to the depot), α_c the penalty associated with temporal constraint violation for customer c, b_c^i the scheduled time to visit customer c in individual i, l_c the latest time to visit customer c, β the penalty associated with number of violated temporal constraints, and $Viol_i$ the number of temporal constraints violated in individual i.

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

Initialization
Repeat
  $p = 1$
  Repeat {evolve population $\mathrm{Pop_p}$-new generation}
    For $j = 1..\ n_p$ ***do***
      Select two parents from $\mathrm{Pop_p}$
      Generate a new solution S_j using recombination and mutation operators associated with $\mathrm{Pop_p}$
      Add S_j to $\mathrm{Pop_p}$
    end for
    Remove from $\mathrm{Pop_p}$ the n_p worst individuals using the evaluation function (1)
    $p = p + 1$
  Until (all populations $\mathrm{Pop_p}$ have been visited)
  if ($\mathrm{Pop_2}$ includes a new best feasible solution) ***then***
      Set $\mathrm{Pop_1} = \mathrm{Pop_2}$ {extinct $\mathrm{Pop_1}$, duplicate $\mathrm{Pop_2}$ in $\mathrm{Pop_1}$}
      Modify $\mathrm{Pop_2}$ solutions by applying RSR_M {mutate $\mathrm{Pop_2}$ individuals, reducing number of routes by one}
  endif
    Apply RC_M on the best computed 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 from Pop_p the n_p worst individuals using $Eval_i$ (Equation (1))
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 [30] to insert at first, as many customers as possible without violating time window constraints. Accordingly, route neighborhoods associated to unvisited customers are repeatedly examined for customer insertion. Introduced by Liu and Shen [30], this new route-neighborhood structure relates one or multiple routes to individual customers. In our approach, route neighborhood is strictly bounded to two tours, comprising routes whose distance separating their centroid from the customer location is minimal. A route centroid corresponds to a virtual site whose coordinates refer to the average position of its specific routed customers. Each feasible customer insertion opportunity is explored over its entire route neighborhood. The next customer visit is selected by maximizing a so-called regret cost function that accounts for multiple route insertion opportunities:

$$\text{Regret Cost} = \sum_{r \in RN(c)} \{C_c(r) - C_c(r^*)\}, \tag{4}$$

where $RN(c)$ is the route neighborhood of customer c, $C_c(r)$ the minimum insertion cost of customer c within route r, and $C_c(r^*)$ the minimum insertion cost of customer c 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 $C_c(r)$ [30] includes an additional term reflecting temporal constraint violation. This new contribution is given by $CV_{s_r}(c)$ (Eq. (3)) where $S_r(c)$ refers to the partial solution described by route r in which customer c is inserted.

2.2. Selection

Selection consists in choosing two parent solutions within the population for mating purposes. The selection procedure is stochastic and biased toward the best solutions using a roulette-wheel scheme [7]. In this scheme, the probability to select an individual is proportional to its fitness value.

Individual fitness is computed as follows:

$$\text{Population Pop}_1 \quad fitness_i = d_i + \sum_{c=0}^{n} \alpha_c \max\{0, b_c^i - l_c\} + \beta\, Viol_i, \tag{5}$$

$$\text{Population Pop}_2 \quad fitness_i = \sum_{c=0}^{n} \alpha_c \max\{0, b_c^i - l_c\} + \beta\, Viol_i. \tag{6}$$

The notations are the same as in Eqs. (1)–(3). Better individuals generally (but not necessarily) tend to include short total traveled distance and minimal temporal constraint violation 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, remaining unvisited customers are inserted within existing routes through temporal constraint relaxation. Constraint violation is used to restrict the total number of routes to a fixed value. The proposed genetic operators incorporate some key features of the best heuristic routing techniques such as Solomon's insertion heuristic I1 [24], large neighborhood search [31] and the route neighborhood-based two-stage metaheuristic (RNETS) [30]. Details on the recombination and mutation operators used are given in the next sections.

2.3.1. Recombination

The insertion-based IB_X crossover operator creates an offspring by combining, one at a time, k routes (R_1) of parent solution P_1 with a subset of customers, formed by nearest-neighbor routes (R_2) from parent solution P_2. The neighborhood R_2 includes the routes of P_2 whose centroid is located within a certain range of $r_1 \in R_1$ (centroid). This range corresponds to the average distance separating r_1 from the routes defining P_2. The routes of R_1 are selected either randomly, with a probability proportional to the number of customers characterizing a tour or based on average distance separating consecutive customers over a route.

A stochastic removal procedure is first carried out to remove from r_1, customers likely to be migrated to alternate routes. Targeted customers are either selected according to waiting times, distance separating them from their immediate neighbors, or randomly. Then, using a modified insertion heuristic inspired from Solomon [24] a feasible child tour is constructed, expanding the altered route r_1 by inserting customer visit candidates derived from the nearest-neighbor routes R_2 defined earlier. The proposed insertion technique consists in adding a stochastic feature to the standard customer insertion heuristic I1 [24], by selecting randomly the next customer visit over the three best candidates with a bias toward the best. 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 of R_1. Finally, the child inherits the remaining "diminished" routes (if any) of P_1. If unvisited customers still remain, additional routes are built using a nearest-neighbor procedure.

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 [15]. 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 tours than expected, it is repeatedly mutated using the RSR_M (Section 2.3.2) procedure to reduce the number of routes. Conversely, for solutions having a smaller number of routes, new feasible routes are constructed repeatedly breaking down the most populated route in two until the targeted number of routes is obtained.

2.3.2. Mutation

A suite of six mutation operators is proposed, namely LNSB_M(d), EE_M, IEE_M, RS_M, RSR_M(I) and RC_M(I).

The LNSB_M(d) (large neighborhood search based) mutation operator relies on the concept of the Large Neighborhood search (LNS) method proposed by Shaw [31]. LNS consists in exploring the search space by repeatedly removing related customers and reinserting them using constraint-based tree search (constraint programming). Customer relatedness defines a relationship linking two customers based upon specific properties (e.g. proximity and/or identical route membership), such that when both customers are considered simultaneously for a visit, they can compete with each other for reinsertion creating new opportunities for solution improvement. Therefore, customers close to one another naturally offer interchange opportunities to improve solution quality. Similarly, solution number of tours is more likely to decrease when customers sharing route membership are removed all together. As stated in Shaw [31], a set of related customers is first removed. Then, customers violating temporal constraints are taken out as well before proceeding to the reinsertion phase. The proposed customer reinsertion method differs from the procedure proposed by Shaw [31] in two respects, namely, the insertion cost function used, and the order in which customers are visited for insertion (variable ordering scheme) through the branch-and-bound search process. Insertion cost is defined by the sum of key contributions referring, respectively, to traveled distance increase and delayed service time, as prescribed in Solomon's procedure I1, as well as to constraint violation (Eq. (3)). As for customer ordering, customers ($\{c\}$) are sorted ($CustOrd$) according to a composite ranking, departing from the myopic scheme originally proposed by Shaw. The ranking is defined as an additive combination of two separate rankings, previously achieved over best insertion costs ($Rank_{\mathrm{Cost}}(c)$) on the one hand, and number of feasible insertion positions ($Rank_{|\mathrm{Pos}|}(\mathrm{c})$) on the other hand:

$$CustOrd \leftarrow Sort(Rank_{\mathrm{Cost}}(c) + Rank_{|\mathrm{Pos}|}(c)). \tag{7}$$

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

$$customer \leftarrow CustOrd[INTEGER(L \times rand^{D})] \tag{8}$$

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

Once a customer is selected, branch-and-bound search is carried out over its different insertion positions as specified in [31], exploiting limited discrepancy search [32]. In LNSB_M, tree expansion is achieved using a non-constant discrepancy factor d, selected randomly (uniform probability

distribution) over the set $\{1,2,3\}$ to further reduce execution time and provide diversification. Despite the implicit contribution of constraint violation in the insertion method, all legal customer visits are first successfully explored. Unvisited customers (if any) are then routed relaxing temporal constraints.

The EE_M (edge exchange) and RS_M (repair solution) mutators focus on inter-route improvement. EE_M attempts to shift customers to alternate routes as well as to exchange sets of customers between two routes. It is inspired from the λ-interchange mechanism of Osman [18], performing reinsertions of customer sets over two neighboring routes. In the proposed mutation procedure, each customer is explored for reinsertion in its surrounding route neighborhood made up of two tours. Tours are being selected such that the distance separating their centroid from customer location is minimal. Customer exchanges occur as soon as the solution improves, i.e., we use a "first admissible" improving solution strategy. Assuming the notation (x, y) to describe the different sizes of customer sets to be exchanged over two routes, the current operator explores values running over the range $(x = 1, y = 0, 1, 2)$. As for the RS_M mutation operator, it focuses on exchanges involving one illegal customer visit. In that scheme, each illegal visit in a route is exchanged with an alternate legal one or two-customer visit sequence in order to generate a new set of customers with either violated or non-violated temporal constraints. The RS_M mutator differs from EE_M in that the former explicitly attempts to generate feasible visits for illegal customers only, even at the expense of solution quality degradation. Therefore, permutations occur even if emerging solutions do not improve. The objective is to further explore the solution space (diversity). The IEE_M (intra-route edge exchange) mutation operator is similar to EE_M except that customer migration is restricted to the same route.

The RSR_M(I) (reinsert shortest route) mutation operator eliminates the shortest route of the solution, reducing by one the total number of routes. Customers from the shortest route are first removed. Then, following an iterative process, various solutions are explored, in which unvisited customers are reinserted within existing routes using the insertion heuristic presented for the RI re-initialization procedure described in Section 2.1. The entire iterative process is repeated over I different sets (e.g. $I = 20$) of randomly generated parameter values, and returns the best solution computed.

The RC_M(I) (reorder customers) mutation operator is an intensification procedure intended to reduce total traveled distance of feasible solutions by reordering customers within a route. The procedure consists in repeatedly reconstructing a new tour using the sequential insertion heuristic I1 [24] over I different sets (e.g. $I = 2$) of randomly generated parameter values, returning the best solution generated shall an improved one emerge.

2.4. Parallel version

Further benefits toward reducing computational cost and limiting run-time impact on performance can be clearly anticipated from the inherent structure of the approach. Taking advantage of the natural propensity of genetic algorithms toward parallelism, a simple parallel procedure based upon a master–slave message-passing paradigm is proposed. The architecture is depicted in Fig. 1.

The master–slave scheme 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 (genetic controller) supervises and controls the execution of the algorithm, handles the parent selection process, population replacement and the emergence of a new feasible

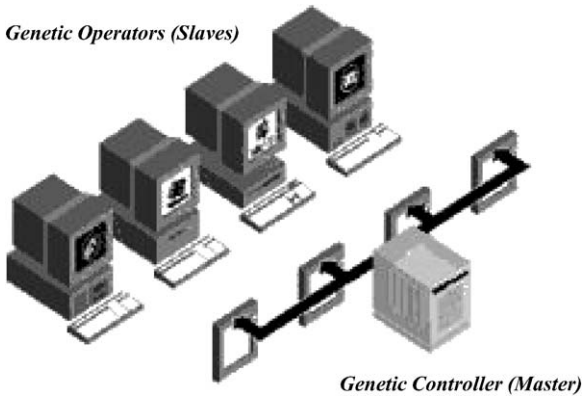

Fig. 1. Master–slave parallel architecture.

solution, selects and synchronizes atomic genetic operator activation and completion, and finalizes construction of new generations. Supervised by the master component, the slave processing elements (genetic operators) concurrently execute reproduction and mutation operators as atomic tasks on multiple processors. A genetic operation task assigned to a slave component by the genetic controller results in the creation of new individuals. Populations evolve concurrently, and execution stops after a pre-determined run-time period (typically 2 min).

The proposed basic architecture is primarily directed toward achieving a quick speed-up from a natural and low-cost parallel implementation effort. Therefore, issues such as design efficiency in properly dealing with the intrinsic sequential component of the parallel method, parallel efficiency, scalability and, communication and synchronization costs incurred while generating individuals or constructing a new population, are disregarded.

3. Computational experiment

A computational experiment has been conducted to compare the performance of the proposed approach with some of the best heuristic methods recently designed for VRPTW. The algorithm has been tested on the VRPTW benchmark proposed by Solomon [24], which includes 56 problem instances. Each problem involves 100 customers 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.

Exploiting the PVM (parallel virtual machine) software [33], the parallel procedure has been implemented in C++, using a modified version of the GAlib genetic algorithm library of Wall [34], on a 19-computer cluster architecture: Linux platform environment, 19 Athlon 1.2 GHz processors (a master and 18 slaves) each with 768 M of RAM, a 100 M/s communication bandwidth and a three module switch including 41 ports.

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 s
Recombination and mutation rates: 100%
Recombination operator: IB_X
Mutation operators: LNSB_M(d), EE_M, IEE_M, RS_M, RSR_M(I) and RC_M(I)

In the LNSB_M(d) mutation operator the number of customers considered for elimination runs in the range [12,17]. The discrepancy factor d is randomly selected over $\{1,2,3\}$. For fitness, evaluation and insertion cost functions:

$$\alpha_c = 100, \quad \forall c$$

$$\alpha_0 = 1000, \quad \beta = 100$$

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%)
RSR_M($I = 20$)—when a new best feasible solution is found.

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%)
RSR_M($I = 20$)—when a new best feasible solution is found.

Given limited computational resources and combinatorial complexity, parameter values were determined empirically over a few intuitively selected combinations, choosing the one that yielded the best average output. This is justified by the fact that the sensitivity of the results with respect to

Table 1
Average performance comparison among VRPTW algorithms

Problem		GTA	RT	TB	LS	GH	BB^s	BB^p
R1	Vehicles	12.38	12.58	12.33	12.25	12.41	12.17	12.08
	Distance	1210.83	1197.42	1220.35	1253.68	1201	1251.40	1288.35
	Time	1800	2700	13774	2495	300	1800	120
R2	Vehicles	3.00	3.09	3.00	2.82	2.91	2.73	2.73
	Distance	960.31	954.36	1013.35	1022.08	945	1056.59	1097.98
	Time	1800	9800	20232	403	300	1800	120
C1	Vehicles	10.00	10.00	10.00	10.00	10.00	10.00	10
	Distance	828.38	828.45	828.45	841.33	829	828.50	828.95
	Time	1800	3200	14630	1459	300	1800	120
C2	Vehicles	3.00	3.00	3.00	3.00	3.00	3.00	3.00
	Distance	591.85	590.32	590.91	591.03	590	590.06	601.23
	Time	1800	7200	16375	224	300	1800	120
RC1	Vehicles	11.92	12.33	11.9	12.00	12.00	11.88	11.88
	Distance	1388.13	1269.48	1381.31	1416.11	1356	1414.86	1456.49
	Time	1800	2600	11264	1869	300	1800	120
RC2	Vehicles	3.33	3.62	3.38	3.25	3.25	3.25	3.25
	Distance	1149.28	1139.79	1198.63	1230.31	1140	1258.15	1388.18
	time	1800	7800	11596	430	300	1800	120

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. Population overlaps were determined according to the most prominent characteristic of a data set. As we aimed at computing the minimum number of routes first, the general strategy consisted in allocating a maximum number of processors to Pop_2, as number of tours minimization generally represents the hardest task to achieve. In cases where minimum number of routes computation has not appeared as a major challenge, more computational resources were allocated to the task of reducing total traveled distance. Therefore, more processing power were devoted to evolve Pop_1 for clustered data sets ($n_1 = 15, n_2 = 3$) in contrast to Pop_2 evolution for alternate problem instances ($n_1 = 1, n_2 = 17$), emphasizing total traveled distance and number of routes minimization respectively. Mainly for a matter of run-time convenience, a different parameter setting was proposed for C1 and C2. The parameters were instantiated based upon a previous genetic algorithm proposed by Berger and Barkaoui [15]. In fact, this class of problem instances does not seem very difficult, as convergence generally occurs very quickly for most VRPTW metaheuristics.

3.2. Results

The results for the six problem data sets are summarized in Tables 1–3 for some of the best-reported heuristic methods for VRPTW, namely GTA [29], RT [25], TB [26], LS [30], GH [20], RGP [35], CR [27], CLM [28], HG [16] and, BB^s and BB^p for the sequential and parallel versions of the hybrid genetic algorithm, respectively. The results are usually ranked according to a hierarchical

Table 2
Best performance comparison among VRPTW algorithms

Problem		RT	LS	RGP	CR	TB	CLM	GTA	HG(ES1)	BB
R1	Vehicles	12.25	12.17	12.08	12.17	12.17	12.08	12.00	11.92	11.92
	Distance	1208.50	1249.57	1210.21	1204.19	1209.35	1210.14	1217.73	1228.06	1221.1
R2	Vehicles	2.91	2.82	3.00	2.73	2.82	2.73	2.73	2.73	2.73
	Distance	961.72	1016.58	941.08	986.32	980.27	969.57	967	969.95	975.43
C1	Vehicles	10.00	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	828.38	828.48
C2	Vehicles	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
	Distance	589.86	591.03	589.86	591.42	589.86	589.86	589.86	589.86	589.93
RC1	Vehicles	11.88	11.88	11.63	11.88	11.50	11.50	11.63	11.63	11.50
	Distance	1377.39	1412.87	1382.78	1397.44	1389.22	1389.78	1382.42	1392.57	1389.89
RC2	Vehicles	3.38	3.25	3.38	3.25	3.38	3.25	3.25	3.25	3.25
	Distance	1119.59	1204.87	1105.22	1229.54	1117.44	1134.52	1129.19	1144.43	1159.37
ALL	Vehicles	415	412	412	411	410	407	407	406	405
	Distance	57231	59317	56953	58502	57522	57556	57516	57876	57952

Table 3
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

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. Many of the methods depicted in Table 2 cannot be listed in Table 1 because they do not report run-time or average results. On the other hand, alternate methods focusing on a different objective function (e.g. traveled distance minimization only) have also been deliberately omitted for comparison purposes.

Table 1 presents the average results of various well-known procedures. The average is computed over all problem instances of the corresponding data set. Each entry refers to the best average 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), total traveled distance and, run-time expressed in seconds. The following columns refer to particular problem-solving methods. The performance of the parallel RHGA is depicted in the last column (BB^p). Related computer platforms include Sun UltraSparc 1 167 MHz (70 Mflops/s) for GTA, Silicon Graphics 100 MHz (15 Mflop/s) for RT, Sun Sparc 10 50 MHz (10 Mflops/s) for TB, HP 9000/720 (17 Mflop/s) for LS, Pentium 200 MHz (24 Mflop/s) for GH,

Pentium 400 MHz (54 Mflops/s) for BB^s and, Athlon1.2 GHz processors (computer network) for BB^p, respectively.

The results of the experiment do not show any conclusive evidence to support a dominating heuristic over the others. But, on average, RHGA proves to be fast, cost effective and highly competitive as it mostly matches the performance of best-known heuristic routing procedures. Computational results for the parallel algorithm show some improvement over its sequential counterpart. Accordingly, a new best result for R1 has been obtained (12.08 tours) in terms of number of routes for near similar or comparable solution quality while providing an approximate speed-up of five (1800 s/(120 s*1.2 GHz/0.4 GHz)). However, this run-time gain is a bit overestimated, as more computational resource would normally be required to match solution quality in terms of total traveled distance. The slightly degraded solution shown for BB^p regarding traveled distance is mainly due to a larger computational effort explicitly devoted to Pop_2 over Pop_1 evolution for minimizing solution number of routes, and indirectly determined by population overlap parameters n_1 and n_2. This imbalance somewhat undermines BB^p's ability to equally perform on distance. The parallel algorithm nonetheless shows a relative advantage over the sequential technique, as number of routes minimization remains the primary objective to be pursued.

The best-computed results are shown in Table 2. Results indicate that RHGA, referred as BB in Table 2, matches or outperforms the best-known heuristic routing procedures in terms of solution number of tours. The last row refers to the cumulative number of routes and traveled distance for all problem instances. The total number of tours computed over all problem data sets outperforms by one the best computed result so far, reported by Homberger and Gehring [16]. In addition, RHGA is the only method that found the minimum number of tours consistently for all problem data sets. In other respect, CLM, GTA and HG appear to perform better than RHGA regarding the total distance criterion, especially in problem groups R2 and RC2. On the average the differences in total distance are, however, quite small (0.76%). RHGA also succeeded in improving six of the best-known solutions. Table 3 provides six new best-known solutions and compares them with the previous best-known solutions. Details of the new solutions are presented in Appendix A.

4. Conclusion

A parallel version of a promising hybrid genetic algorithm targeted to the vehicle routing problem with time windows has been successfully developed. The proposed approach involves concurrent evolution of two populations dealing with separate objectives and temporal constraint relaxation. A first population evolves individuals to minimize total traveled distance whereas the second aims at minimizing temporal constraint violation to generate a feasible solution. Genetic operators have been designed to incorporate key features from recent routing techniques to further diversify and intensify search while exploring the solution space. The parallel method 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 manages high-level 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. Results from a computational experiment show the proposed technique to be very competitive with the best-known heuristic routing procedures providing some new best-known solutions. The parallel version of the

algorithm further expands on its sequential counterpart, 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 particular instances. The parallel algorithm shows a speed-up of five in computing solution having near similar quality. Future work will further explore applications and variants of the parallel algorithm. In that regard, performance studies on larger problem instances and targeted real-time applications will be undertaken.

Acknowledgements

The authors are very grateful and wish to express their sincere appreciation to Dr. Olli Bräysy, a researcher with SINTEF Applied Mathematics, Oslo, Norway, for his valuable comments, suggestions and feedback over the whole duration of this work.

Appendix A.

The new best-known solutions for Solomon's (1987) data sets are specified as follows:

R108:
Routes: 9
Total Traveled Distance: 960.876
1. 73 72 75 56 23 67 39 55 25 54 26
2. 92 98 91 44 14 38 86 16 61 85 100 37
3. 27 69 1 53 40 21 4 74 22 41
4. 28 12 80 76 3 79 78 34 29 24 68 77
5. 50 33 81 51 9 35 71 65 66 20
6. 2 57 15 43 42 87 97 95 94 13 58
7. 6 96 59 93 99 5 84 17 45 83 60 18 89
8. 52 7 48 82 8 46 47 36 49 19
9. 31 88 10 62 11 64 63 90 32 30 70

R110:
Routes: 10
Total Traveled Distance: 1119
1. 2 41 22 74 73 40 53 26 54 24
2. 31 11 63 90 10 20 66 65
3. 52 82 8 18 7 48 46 45 60 89
4. 21 72 75 56 23 67 39 25 55 4
5. 59 98 44 16 86 38 14 43 42 13 58
6. 95 15 57 87 94 97 92 37 100 91 93
7. 27 69 30 51 9 71 35 34 78 33 1
8. 88 62 19 47 36 49 64 32 70
9. 28 76 12 29 81 79 3 50 77 68 80
10. 83 5 17 84 61 85 99 96 6

RC105:
Routes: 13
Total Traveled Distance: 1629.44
1. 42 61 8 6 46 4 3 1 100
2. 98 14 47 15 16 9 10 13 17
3. 33 76 89 48 21 25 24
4. 72 71 81 41 54 96 94 93
5. 90 53 66 56
6. 39 36 44 38 40 37 35 43 70
7. 31 29 27 30 28 26 32 34 50 80
8. 63 62 67 84 51 85 91
9. 65 82 12 11 87 59 97 75 58
10. 83 19 23 18 22 49 20 77
11. 2 45 5 7 79 55 68
12. 69 88 78 73 60
13. 92 95 64 99 52 86 57 74

RC106:
Routes: 11
Total Traveled Distance: 1424.73
1. 14 11 87 59 75 97 58 74
2. 72 71 67 30 32 34 50 93 80
3. 95 62 63 85 76 51 84 56 66
4. 82 52 99 86 57 22 49 20 24 91
5. 2 45 5 8 7 6 46 4 3 1 100
6. 15 16 47 78 73 79 60 55 70
7. 42 44 39 40 36 38 41 43 37 35
8. 69 98 88 53 12 10 9 13 17
9. 33 31 29 27 28 26 89
10. 92 61 81 90 94 96 54 68
11. 65 83 64 19 23 21 18 48 25 77

R210:
Routes: 3
Total Traveled Distance: 954.121
1. 95 92 42 15 23 67 39 75 22 41 57 87 99 6 94 53 40 21 73 72 74 56 4 55 25 54 26 58
2. 28 69 1 30 65 71 33 50 76 12 29 3 79 78 81 9 51 20 32 90 63 10 31 70 66 35 34 24 80 68 77
3. 27 52 7 47 36 64 11 62 88 18 45 16 44 14 38 86 61 84 5 60 83 8 82 48 19 49 46 17 85 98 37 97 13 2
43 100 91 93 59 96 89

R211:
Routes: 2
Total Traveled Distance: 906.192
1. 95 92 98 42 15 2 21 72 73 39 67 23 75 22 41 57 87 40 53 12 76 3 29 79 33 81 51 30 71 65 35 34 78 9 66 20 32 10 70 1
50 77 68 80 24 25 55 54

2. 28 27 69 31 52 83 5 61 16 44 14 38 86 85 99 18 82 7 88 62 19 11 90 63 64 49 36 47 48 46 8 45 84 6 94 96 59 93 37 97
13 58 26 4 56 74 43 100 91 17 60 89

References

[1] Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM, Soumis F. The VRP with time windows. In: Toth P, Vigo D, editors. The vehicle routing problem, SIAM monographs on discrete mathematics and applications, Society for Industrial & Applied Mathematics, Philadelphia, USA, 2002 [Chapter 7].

[2] Desrosiers J, Dumas Y, Solomon MM, Soumis F. Time constrained routing and scheduling. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL, editors. Handbooks in operations research and management science, network routing, vol. 8. Amsterdam: North-Holland, 1995. p. 35–139.

[3] Bräysy O, Gendreau M. Vehicle routing problem with time windows, part I: route construction and local search algorithms. Internal Report STF 42 A01024, SINTEF Applied Mathematics, Department of Optimization, Norway, 2001.

[4] Bräysy O, Gendreau M. Vehicle routing problem with time windows, Part II: metaheuristics. Internal Report STF 42 A01025, SINTEF Applied Mathematics, Department of Optimization, Norway, 2001.

[5] Holland JH. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press, Ann Arbor, 1975.

[6] Jong De KA. An analysis of the behavior of a class of genetic adaptive systems. PhD dissertation, University of Michigan, USA, 1975.

[7] Goldberg DE. Genetic algorithms in search, optimization, and machine learning. New York: Addison-Wesley, 1989.

[8] Blanton JL, Wainwright RL. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In: Forrest S, editor. Proceedings of the 5th International Conference on Genetic Algorithms. San Francisco: Morgan Kaufmann, 1993. p. 452–9.

[9] Thangiah S. Vehicle routing with time windows using genetic algorithms. In: Chambers L, editor. Application handbook of genetic algorithms: New Frontiers, vol. II. Boca Raton: CRC Press, 1995. p. 253–77.

[10] Thangiah SR. An adaptive clustering method using a geometric shape for vehicle routing problems with time windows. In: Eshelman LJ, editor. Proceedings of the 6th International Conference on Genetic Algorithms. San Francisco: Morgan Kaufmann, 1995. p. 536–43.

[11] Thangiah SR, Osman IH, Vinayagamoorthy R, Sun T. Algorithms for the vehicle routing problems with time deadlines. American Journal of Mathematical and Management Sciences 1995;13:323–55.

[12] Potvin J-Y, Bengio S. The vehicle routing problem with time windows Part II: genetic search. INFORMS Journal on Computing 1996;8:165–72.

[13] Berger J, Salois M, Begin R. A hybrid genetic algorithm for the vehicle routing problem with time windows. Lecture Notes in Artificial Intelligence, vol. 1418. AI'98, Advances in Artificial Intelligence, Vancouver, Canada, 1998. p. 114–27.

[14] Berger J, Sassi M, Salois M. A hybrid genetic algorithm for the vehicle routing problem with time windows and itnerary constraints. In: Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, USA, 1999. p. 44–51.

[15] Berger J, Barkaoui M. 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, 2000.

[16] Homberger J, Gehring H. Two evolutionary metaheuristics for the vehicle routing problem with time windows. INFOR 1999;37:297–318.

[17] Or I. Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. PhD thesis, Northwestern University, Evanston, USA, 1976.

[18] Osman IH. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operations Research 1993;41:421–51.

[19] Potvin J-Y, Rousseau J-M. An exchange heuristic for routing problems with time windows. Journal of the Operational Research Society 1995;46:1433–46.

[20] Gehring H, Homberger J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Miettinen K, Mäkelä M, Toivanen J, editors. 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, 1999. p. 57–64.
[21] Gehring H, Homberger J. Parallelization of a two-phase metaheuristic for routing problems with time windows. Asia-Pacific Journal of Operational Research 2001;18:35–47.
[22] Bräysy O, Berger J, Barkaoui M. A new hybrid evolutionary algorithm for the vehicle routing problem with time windows. Presented in Route 2000 Workshop, Skodsborg, Denmark, 2000.
[23] Tan KC, Lee LH, Ou K. Hybrid genetic algorithms in solving vehicle routing problems with time window constraints. Asia-Pacific Journal of Operational Research 2001;18:121–30.
[24] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 1987;35:254–65.
[25] Rochat Y, Taillard E. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1995;1:147–67.
[26] Taillard E, Badeau P, Gendreau M, Guertin F, Potvin J-Y. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 1997;31:170–86.
[27] Chiang W-C, Russell RA. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. INFORMS Journal on Computing 1997;9:417–30.
[28] Cordeau J-F, Laporte G, Mercier A. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society 2001;52:928–36.
[29] Gambardella LM, Taillard E, Agazzi G. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: Corne D, Dorigo M, Glover F, editors. New ideas in optimization. London: McGraw-Hill, 1999. p. 63–76.
[30] Liu F-H, Shen S-Y. A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. European Journal of Operational Research 1999;118:485–504.
[31] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget J-F, editors. Principles and practice of constraint programming, Lecture Notes in Computer Science. New York: Springer, 1998. p. 417–31.
[32] Harvey WD, Ginsberg ML. Limited discrepancy search. In: Proceedings of the 14th IJCAI, Montreal, Canada, 1995.
[33] Geist Al, et al. A users' guide and tutorial for networked parallel computing. In: Kowalik J, editor. MIT Press scientific and engineering computation, Massachusetts Institute of Technology, Boston, 1994. (http://www.netlib.org/pvm3/book/pvm-book.html).
[34] Wall M. GAlib - A C++ Genetic Algorithms library, version 2.4. (http://lancet.mit.edu/galib-2.4/), Boston: MIT, 1995.
[35] Rousseau L-M, Gendreau M, Pesant G. Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of Heuristics 2002;8:43–58.

Jean Berger is a defence scientist with the Decision Support Systems Section of Defence Research and Development Canada—Valcartier, working in the field of information technology. He received BS and MS degrees in engineering physics from Ecole Polytechnique de Montreal, Canada. His research interests include artificial intelligence and operations research applied to intelligent control, planning, routing and scheduling problems.

Mohamed Barkaoui is a research assistant associated with the Decision Support Systems Section of Defence Research and Development Canada—Valcartier, working in the field of resource planning and intelligent control using operations research and artificial intelligence technologies. He holds a Master's degree in Computer Science from Laval University, Quebec City, Canada and, Engineering in Computer Science from University of Tunis, Tunisia. His research interests include computer modeling and simulation.