SINTEF

SINTEF REPORT

SINTEF ICT

Address:	P.O.Box 124, Blindern 0314 Oslo NORWAY
Location:	Forskningsveien 1
Telephone:	+47 22 06 73 00
Fax:	+47 22 06 73 50

Enterprise No.: NO 948 007 029 MVA

TITLE

Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows

AUTHOR(S)

Olli Bräysy, Wout Dullaert and Michel Gendreau

CLIENT(S)

SINTEF ICT, Research Council of Norway

REPORT NO.	CLASSIFICATION Open	CLIENTS REF. TOP – 140689/420	
CLASS. THIS PAGE Open	ISBN	PROJECT NO.	NO. OF PAGES/APPENDICES 33/0
ELECTRONIC FILE CODE EA_VRPTW_Report.doc		PROJECT MANAGER (NAME, SIGN.) Geir Hasle	CHECKED BY (NAME, SIGN.)
FILE CODE	DATE 2004-06-07	APPROVED BY (NAME, POSITION, SIGN.)	

ABSTRACT

This paper surveys the research on evolutionary algorithms for the Vehicle Routing Problem with Time Windows (VRPTW). The VRPTW can be described as the problem of designing least cost routes from a single depot to a set of geographically scattered points. The routes must be designed in such a way that each point is visited only once by exactly one vehicle within a given time interval; all routes start and end at the depot, and the total demands of all points on one particular route must not exceed the capacity of the vehicle. The main types of evolutionary algorithms for the VRPTW are genetic algorithms and evolution strategies. In addition to describing the basic features of each method, experimental results for the benchmark test problems of Solomon (1987) and Gehring and Homberger (1999) are presented and analyzed.

KEYWORDS	ENGLISH	NORWEGIAN
GROUP 1	Computer Science	Informatikk
GROUP 2	Optimization	Optimering
SELECTED BY AUTHOR	Vehicle routing	Ruteplanlegging
	Evolutionary algorithms	
	Genetic algorithms	

TABLE OF CONTENTS

1 Introduction

Routing and scheduling problems are important elements of many logistics systems. A lot of research has been devoted to finding effective methods to tackle these complex combinatorial problems. One of the major research topics has been the Vehicle Routing Problem with Time Windows (VRPTW). The VRPTW can be defined as follows. Let $G = (V, E)$ be a connected digraph consisting of a set of $n + 1$ nodes, each of which can be reached only within a specified time interval or time window, and a set E of arcs with non-negative weights representing travel distances and associated travel times. Let one of the nodes be designated as the depot. Each node i, except the depot, requests a service of size q_i . In the VRPTW all customers request the same type of service: pickups or deliveries. Moreover, each customer must be serviced once, thus prohibiting split service and multiple visits. The primary objective of the VRPTW is to find the minimum number of tours, K^*, such that each node is serviced within its time window and that the accumulated service up to any node does violate the vehicle capacity Q .The tours correspond to feasible routes starting and ending at the depot. Often a secondary objective is imposed to minimize the total distance traveled or to minimize total schedule time. All problem parameters, such as customer demands, travel times and time windows, are assumed to be known with certainty.

The VRPTW is a basic distribution management problem that can be used to model many real-world problems. Some of the most useful applications of the VRPTW include bank deliveries, postal deliveries, industrial refuse collection, national franchise restaurant services, school bus routing, security patrol services and JIT (just in time) manufacturing. The VRPTW has been the subject of intensive research efforts for both heuristic and exact optimization approaches. Early surveys of solution techniques for the VRPTW can be found in Golden and Assad (1986 and 1988), Desrochers et al. (1988), and Solomon and Desrosiers (1988). The main focus in Desrosiers et al. (1995) and Cordeau et al. (2001a) are exact solution techniques. Further details on these exact methods can be found in Larsen (1999) and Cook and Rich (1999). Because of the high complexity level of the VRPTW and its wide applicability to real-life situations, solution techniques capable of producing high-quality solutions in limited time, i.e. heuristics are of prime importance. Current VRPTW heuristics can be categorized as follows: (i) construction heuristics, (ii) improvement heuristics and (iii) metaheuristics.

Construction heuristics are sequential or parallel algorithms aiming at designing initial solutions to routing problems that can be improved upon by improvement heuristics or metaheuristics.

Sequential algorithms build a route for each vehicle, one after another, using decision functions for the selection of the customer to be inserted in the route and the insertion position within the route. Parallel algorithms build the routes for all vehicles in parallel, using a pre-computed estimate of the number of routes. Different variants of construction heuristics for the VRPTW can be found in Solomon (1987), Potvin and Rousseau (1993), Bramel and Simchi-Levi (1996), Ioannou et al. (2001) and Dullaert and Bräysy (2003).

Most of the recently published VRPTW heuristics use a two-phase approach. First, a construction heuristic is used to generate a feasible initial solution. Second, an iterative improvement heuristic is applied to the initial solution. These route improvement methods iteratively modify the current solution by performing local searches for better neighboring solutions. Generally, a neighborhood comprises the set of solutions that can be obtained by applying a single move to the current solution. For the most successful applications to the VRPTW, see Thompson and Psaraftis (1993), Potvin and Rousseau (1995), Russell (1995), Shaw (1998), Caseau and Laburthe (1999), Cordone and Wolfler-Calvo (2001), and Bräysy et al. (2003). Construction and improvement heuristics are discussed in detail in Bräysy and Gendreau (2002a).

To escape from local optima, the improvement procedure can be embedded in a metaheuristic framework. In general, a metaheuristic is guided by intelligent search strategies to avoid getting trapped in local optima. Metaheuristics such as simulated annealing (Metropolis et al., 1953 and Kirkpatrick et al., 1983) and tabu search (Glover, 1986 and Glover and Laguna, 1997) use myopic moves similar to conventional local search algorithms to move from the incumbent solution to a neighboring solution. Under certain conditions, simulated annealing and tabu search also accept moves to neighboring solutions that worsen the objective value. For the most successful applications to the VRPTW, we refer to Rochat and Taillard (1995), Taillard et al. (1997), Cordeau et al. (2001b), Czech and Czarnas (2002), Li and Lim (2003a), and to the recent survey by Bräysy and Gendreau (2002b). Evolutionary algorithms (Fogel, 1995 and Miettinen et al., 1999) use genetic operators instead of a neighborhood structure to solve optimization problems. Evolutionary algorithms have definitely been among the most suitable approaches for tackling the VRPTW. To our knowledge, these have not been comprehensively surveyed and compared. This survey aims at filling this gap.

This paper does not explore the research on other time-constrained routing problems than the VRPTW, such as the Pickup Delivery Problem with Time Windows (PDPTW). In the PDPTW

(Nanry and Barnes, 2000; Li and Lim 2003b), each transportation request specifies a single origin and a single destination, instead of just transporting between depot and peripheral locations. The VRP with backhauls and time windows (Thangiah et al., 1996; Duhamel et al. 1997) is a special case of PDPTW where the deliveries precede pickups. In Angelelli and Mansini (2001), pick-ups and deliveries at customers are performed simultaneously. Lau and Liu (1999) and Campbell et al. (2001) consider an inventory routing problem with time windows, where inventory management at customer sites is combined with route planning. The Fleet Size and Mix VRP with Time Windows (Liu and Shen, 1999b; Dullaert et al., 2002) combines tactical selection of the heterogeneous vehicle fleet with routing. Cordeau et al. (2001b) consider a VRPTW with multiple depots, and a Periodic VRPTW. Lund et al. (1996), Shieh and May (1998) and Gendreau et al. (1999) study a dynamic VRPTW, where a subset of customers and demands are unknown beforehand. For more details on these time-constrained routing problems, we refer to Bräysy et al. (2002).

The remainder of this paper is organized as follows. The evolutionary algorithms are introduced in Section 2. Section 3 divides evolutionary algorithms developed for the VRPTW into traditional and modern genetic algorithms, evolution strategies and hybrids. The performance of the algorithms is presented and analyzed in Section 4 and Section 5 concludes the paper.

2 Evolutionary Algorithms

Evolution is a phenomenon of adapting to the environment and passing on genetic information to following generations. Darwin (1859) identified three basic principles driving natural evolution: reproduction, natural selection, and the diversity of individuals. These features of natural evolution have found entrance to a broad class of evolutionary algorithms (Fogel, 1995; Bäck et al., 1997; Miettinen et al., 1999) that mimic biological evolution and natural selection. In general, an Evolutionary Algorithm (EA) is characterized by maintaining a set of solution candidates that undergoes a selection process, and is manipulated by genetic operators. By analogy to natural evolution, the solution candidates are called individuals and the set of solution candidates is called the population. Each individual represents a possible solution to the problem at hand. However, an individual is not a decision vector but rather encodes the solution to the optimization problem into a decision vector based on an appropriate structure, e.g., a bit vector or a real-valued vector. Each subsection of the data structure holding the encoded solution (chromosome) is called a gene, and it usually encodes the value of a single parameter.

Selection is a process in which design candidates (parents) are selected for recombination based on their fitness values. Here fitness refers to measure of profit, utility or goodness to be maximized while exploring the solution space. Recombination (or crossover) and mutation are genetic operators aiming at generating new solutions within the search space from existing ones. The crossover operator combines information from a certain number of parents to create a certain number of children (offspring). By contrast, the mutation operator modifies individuals by randomly changing (typically) small parts in the associated decision vectors according to a given probability (mutation rate). Both crossover and mutation work on individuals, not on the decoded decision vectors.

Based on the above concepts, natural evolution is simulated by an iterative computation process. In the beginning a population of candidate solutions to a problem at hand is initialized. This is often accomplished by randomly sampling from the solution space. Then a loop consisting of parent evaluation (fitness assignment), selection, recombination and/or mutation is executed a certain number of times. Each loop iteration is called a generation, and the search is stopped once some convergence criteria or conditions are met. Such criteria might, for instance, refer to a maximum number of generations or the convergence to a homogeneous population composed of similar individuals. Evolutionary algorithms are typically divided into three main subclasses: genetic algorithms (Holland, 1975, De Jong, 1975 and Goldberg, 1989), evolution strategies (Rechenberg, 1973 and Schwefel, 1977) and evolutionary programming (Fogel et al., 1966).

The genetic algorithm is an adaptive heuristic search method developed by Holland (1975). As an EA, a GA is an iterative procedure that maintains a pool of candidates (chromosomes), represented by a fixed-length character (bit) string, simulated over a number of generations. At each generation, chromosomes are subjected to selection, crossover and mutation. Selection and crossover search a problem space by exploiting information present in the chromosomes by selecting and recombining primarily individuals with high fitness values. The mutation is a secondary operator that prevents premature loss of information by randomly changing parts of single individuals. Genetic Algorithms (GA) owe their name to an early emphasis on a (binary) encoding and manipulating the genetic makeup of individuals (genotype) rather than using the physical expression of the genetic makeup (phenotype). The practicality of using the GA to solve complex optimization problems was demonstrated in De Jong (1975) and Goldberg (1989), and for an extensive bibliography, we refer to Alander (2000).

Evolution strategies were originally developed by Rechenberg (1973) and Schwefel (1977) to optimize real-value engineering design problems. There are three important features that distinguish evolution strategies from other evolutionary algorithms. First, ES use a real-coding of the decision vector, and model the organic evolution at the level of individuals' phenotypes. Second, ES depend on a deterministic selection of the individuals for the next generation, and the search is mainly driven by mutation. Third, the individuals' representation includes a vector of so-called "strategy parameters" in addition to the solution vector and both components are evolved by means of recombination and/or mutation operators. An individual in ES is represented as a pair of real vectors, $v = (x, \sigma)$. The first vector, *x*, represents a solution in the search space and consists of real-valued variables. The second vector, σ, represents the strategy parameters. In the real-value case, the strategy parameters represent deviations from normally distributed random variables. Mutation is performed by replacing *x* by $x^{t+1} = x^t + N(0,\sigma)$ where $N(0,\sigma)$ is a random Gaussian number with a mean of zero and standard deviation σ.

Evolutionary programming, originally conceived by L. Fogel (1966), represents individuals phenotypically as finite state machines capable of responding to environmental stimuli, and developing operators (primarily mutation) for effecting structural and behavioral change over time. Evolutionary programming was dormant for many years, but was reintroduced by D. Fogel in the early 1990s. The new evolutionary programming, is nearly identical to evolution strategies, using similar mutation strategies and a slightly different selection process (related to Tournament Selection).

These early characterizations, however, are no longer useful in describing the enormous variety of existing evolutionary algorithms. GA practitioners are seldom constrained to binary implementations. ES practitioners have incorporated recombination operators into their systems. EP is used for much more than just evolution of finite state machines. The literature is filled with new terms and ideas such as memetic algorithms (Moscato, 1989 and 1999) that combine local search with recombination. Most of the evolutionary methods developed for the VRPTW are hybrids, incorporating real-value representation and a set of construction heuristics and local searches. Nevertheless, the authors call them genetic algorithms. As a consequence, labels such as GA are not that helpful in understanding the algorithm in question. In this paper, we try to avoid this problem by focusing on basic elements common to all evolutionary algorithms, and using them to understand and analyze the differences of genetic algorithms, hybrid genetic algorithms and evolution strategies developed for the VRPTW. To this end, the next Section will focus on coding issues, on how the initial population is created and how the individuals are evaluated, selected and modified.

3 EAs for the VRPTW

In this section we review the evolutionary algorithms developed for the VRPTW. The section is divided in four subsections addressing traditional and modern genetic algorithms, evolution strategies and hybrid approaches to the VRPTW.

3.1 Traditional genetic algorithms

Most traditional GAs for the VRPTW apply encoding of the individuals. Some encode solutions into the chromosomes (e.g. Blanton and Wainwright, 1993), while some others encode sectors (e.g. Thangiah 1995a, 1995b) and parameters of a constructive algorithm (e.g., Benyahia and Potvin 1995, Potvin et al. 1996) to build a solution.

Blanton and Wainwright (1993) hybridize a genetic algorithm with a greedy construction 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. The mutation operator randomly exchanges the position of customer indices in the sequence. When creating offspring, the crossover operator takes into account global precedence relationships among customers (related to time, distance to previous customer and capacity) to determine the ordering of customers in the child. For example, it is generally desirable to insert customer c_i before customer c_j during the greedy insertion phase, if the time window at customer c_i occurs before the time window at customer c_j. Accordingly, the crossover operator pushes customers with early time windows to the front of the orderings. The authors use the so-called Davis encoding method, where a chromosome represents a permutation of n customers to be partitioned into m vehicles. The evaluation function assumes that the first m customers of a chromosome are placed into the m vehicles. The remaining $n - m$ customers are examined individually. As each new customer is selected, a possible subtour is evaluated for each vehicle and the best subtour is selected. If some of the customers remain unrouted, the fitness value is the number of these unserviced customers, otherwise the fitness value is based on the total distance of the solution.

Thangiah (1995a) describes a cluster-first, route-second method called GIDEON that assigns customers to vehicles by partitioning the customers into sectors with a genetic algorithm. Customers within each sector are routed using the cheapest insertion heuristic of Golden and Stewart (1985). In the next step, the routes are improved using λ-interchanges introduced in Osman (1993). The two processes are run iteratively for a finite number of times to improve solution quality. During the search, the search strategy also accepts infeasibilities at the expense of a penalty. The search begins by clustering customers either according to their polar coordinate angle or on a random basis. More precisely, the customers are first divided into K sectors, by a set of seed angles in the search space and drawing a ray from the depot to each seed angle. Here K equals the initial number of vehicles required to service the customers according to Solomon's (1987) I1 insertion heuristic. Each seed angle is computed using a fixed angle and an offset from the fixed angle. Then a pseudo-polar coordinate angle is calculated for each customer by normalizing the angles between the customers so that the angular difference between any two adjacent customers is equal. Customer c_i is assigned to vehicle r_k if the pseudo-polar coordinate angle s_i is greater than the seed angle S_k but is less than or equal to seed angle S_{k+1}. The genetic algorithm is used to decide the offset values that allow the sector to encompass a larger or smaller sector area. In the GIDEON system, each chromosome represents a set of possible clustering schemes and the fitness values are based on corresponding routing costs. A standard 2-point crossover operator exchanges a randomly selected portion of the bit string between the chromosomes and mutation is used with a very low probability to randomly change the bit values.

An approach similar to GIDEON, called GenClust is developed in Thangiah (1995b). Instead of sectors, particular geometric shapes are used to cluster customers. In GenClust, each chromosome encodes n_c different circles, one for each cluster, and the GA is then used to search for the set of circles that leads to the best solution. Different heuristic rules are used to associate a customer with a particular cluster, when it is not contained in exactly one circle (i.e., none or several).

Thangiah et al. (1995) use the same approach as Thangiah (1995a) to solve the vehicle routing problems with time deadlines (VRPTD), i.e., a situation in which customers impose a latest time of service but no earliest time of service. In addition, they define two heuristics based on the principles of time-oriented sweep and cheapest insertion procedures for solving the VRPTD, followed by λ-interchanges of Osman (1993). The authors conclude that the genetic algorithm based heuristic does well for problems in which the customers are distributed uniformly and/or with short time deadlines. The other two heuristics perform well for problems in which the customers are tightly clustered or have long deadlines.

Potvin et al. (1996) use the competitive neural network of Potvin and Robillard (1995) to select the seed customers for the modification of Solomon's insertion heuristic (Potvin and Rousseau, 1993) where several routes are constructed simultaneously. The algorithm requires a value for three parameters, α_1, α_2 and μ. The first two constants determine the importance of distance and travel time in the cost function for inserting each unrouted customer. The third parameter is used to weigh the distance savings. A genetic algorithm is used to find values for these three constants. A stochastic selection procedure is applied to the fitness values based on the number of routes and total route time of the best solution produced by the parallel insertion heuristic. A classical 2-point crossover operator is used for recombination. It swaps a segment of consecutive bits between the parents. The mutation changes with very low probability a bit value from 0 to 1 or from 1 to 0. The results are slightly better compared to using the original insertion heuristic without preprocessing.

A GA approach similar to Potvin et al. (1996) is used in Benyahia and Potvin (1995) to optimize the parameter values of sequential and parallel versions of Solomon's (1987) insertion heuristic. However, here seed customers are selected as in Solomon (1987) and Potvin and Rousseau (1993) instead of using neural networks. Moreover, the authors introduce additional cost measures for insertion, involving slack and waiting times, the cost of insertion compared to the cost of servicing the customer in an individual route, and ratio of additional distance to original distance between the pair of consecutive customers.

3.2 Modern genetic algorithms

Potvin and Bengio (1996) propose a genetic algorithm called GENEROUS that directly applies genetic operators to solutions, thus avoiding coding issues. The initial population is created with Solomon's (1987) cheapest insertion heuristic and the fitness values of the proposed approach are based on the number of vehicles and the total route time of each solution. The selection process is stochastic and highly biased towards the best solutions. For this purpose a linear ranking scheme is used. The linear ranking scheme prevents individuals, with significantly better fitness values than average, from dominating the selection process. Baker (1985) suggested deriving the probability of an individual being selected for mating from its rank within the population, instead of calculating it directly from the objective value. During the recombination phase, two parent solutions are merged into a single one, so as to guarantee the feasibility of the new solution. Two types of crossover operators are used, namely a route-based and a sequence-based crossover. The route-based crossover replaces one route of parent solution P_2 by a route of parent solution P_1 whereas in the sequence-based crossover only a randomly defined end part in a route of parent-solution P_1 is replaced by a set of customers served by a route of parent solution P_2. A repair operator is used to remove duplicates and insert missing customers into the solution. Mutation operators are aimed at reducing the number of routes by trying to insert the customers of a randomly selected short route into other routes, either directly or by first removing some customer from the target route and inserting it into some other route to make room for the new customer. Finally, in order to locally optimize the solution, a mutation operator based on Or-opt exchanges (Or, 1976) is included.

Berger et al. (1998) hybridize a genetic algorithm with well-known construction heuristics. The authors omit the coding issues and represent a solution by a set of feasible routes. The initial population is created with a nearest neighbor insertion heuristic inspired by Solomon (1987). The fitness values of the individuals are based on the number of routes and total distance of the corresponding solution and for selection purposes the authors use the so-called “roulette-wheel” scheme (Goldberg, 1989). The proposed crossover operator combines iteratively various routes r_1 of parent solution P_1 with a subset of customers, formed by r_2 nearest-neighbor routes from parent solution P_2. A removal procedure is first carried out to remove some key customer nodes from r_1. Then, an insertion heuristic inspired by Solomon (1987) coupled to a random customer acceptance procedure is locally applied to build a feasible route, considering the partial route r_1 as initial solution. Here only the customer nodes in routes r_2 are considered for insertion. The stochastic customer removal procedure removes either randomly specific customers, customers rather distant from their successors, or customers with waiting times. The mutation operators are aimed at reducing the number of routes of solutions having only a few customers by trying to insert them into other routes or locally reordering routes using the Nearest Neighbor heuristic of Solomon (1987). A similar approach is used in Berger et al. (1999) to tackle VRPTW with itinerary constraints (maximum route time).

Bräysy (1999a and 1999b) extends the work of Berger et al. (1998) by proposing several new crossover and mutation operators, testing different forms of genetic algorithms, selection schemes and scaling schemes, as well as the significance of the initial solutions. The best performing recombination operator first removes a set of customers within randomly generated segments from parent solution P_1. Then, the reinsertion of each of the removed customers is performed by considering only partial routes that are geographically close to the route to which the customer belongs to in parent solution P_2. The best-performing mutation operator randomly selects one of the shortest routes and tries to eliminate it by inserting its customers into other, longer routes. The main conclusions regarding the design of genetic algorithms for the VRPTW are that it is important to create many new offspring (children) in each generation and that maintaining a single population (contrary to multiple populations) is sufficient for achieving high quality solutions. Differences between different selection schemes are concluded to be minor. The best results were obtained with the so-called tournament selection that performs the well-known roulette-wheel scheme twice and selects the better out of the two individuals identified by the roulette-wheel scheme. A new scaling scheme based on a weighted combination of number of routes, total distance and waiting time is found to perform particularly well. Finally, to create the initial population, several strategies, such as heuristics of Solomon (1987) and randomly created routes are evaluated. Computational testing revealed that best strategy consists of creating a diverse initial population that also contains some individuals with better fitness scores.

Berger et al. (2003) continue Berger et al. (1998) in an approach that reminiscent of Large Neighborhood Search (Shaw, 1998) by first removing a set of customers from the solution and then reinserting them by a probabilistic insertion criterion. Two populations are evolved in parallel. The first population is used to minimize the total distance and the second population tries to minimize violations of time window constraints. The initial population is created using a random sequential insertion heuristic combined with λ-exchanges and a reinitialization procedure based on the insertion procedure of Liu and Shen (1999a). The first of the two recombination operators is the same as in Berger et al. (1998). The second extends the first operator by also removing illegally routed customers and by using the insertion procedure proposed in Liu and Shen (1999a) instead of Solomon’s (1987) heuristic in the reinsertion phase. Here, the cost function is extended to consider temporal constraint violations. Five mutation operators are randomly applied. The first one is a modification of Large Neighborhood Search (LNS) by Shaw (1998), where the order of reinsertions and the cost function are new. The other mutation operators involve λ-exchanges, exchange of customers served too late in the current solution, elimination of the shortest route using the procedure of Liu and Shen (1999a) and within-route reordering using Solomon’s (1987) heuristic.

Zhu (2000) presents a genetic algorithm based on an integer representation of solutions and two new crossover operators. The initial population is a combination of solutions created by the insertion heuristic of Solomon (1987), a set of randomly created λ-interchange neighbors of the heuristic solution and totally randomly created solutions. The parents are selected with tournament selection and the recombination is based on selecting randomly a cut-off point in both parents and then selecting the customer right after the cut-off point from either parent. This customer is then inserted into the same position in the other parent solution, and corrections are performed to replace the duplicate customer by the replaced one. The next customer to be reinserted is selected based either on distances or latest arrival times with respect to the previously chosen customer. Mutation is based on reversing the order of a pair or sequence of nodes. Moreover, a special hill-climbing technique is used, where a randomly selected part of the population is improved by partial λ-exchanges.

Tan et al. (2001a) introduce a genetic algorithm similar to Zhu's (2000) implementation. The representation, strategy for creating the initial population and selection scheme are the same. The well-known PMX crossover operator is used to interchange gene material between chromosomes and mutation is performed by randomly swapping nodes. The basic idea in the PMX crossover is to use two crossover points which select two sequences of genes in both parents. By mapping the genes at corresponding positions (loci) in the selected sequence in both parents, an interchange mapping is defined. Next, each gene in parent 1 (and 2) is replaced by its counterpart as prescribed by the interchange mapping. Since the authors do not use delimiters to distinguish customers belonging to different routes, one chromosome may be grouped into routes in several different ways. The basic grouping is determined by insertion heuristic I1 of Solomon (1987), and the authors use λ-interchanges to create alternative groupings. After the grouping process, a hill-climbing method similar to the one used by Zhu (2000) is employed to further improve a part of the population.

Tan et al. (2001b) apply the *messy genetic algorithm* (Goldberg et al., 1989) to the VRPTW. The solution is encoded using ordered pairs, consisting of customer and vehicle identification indexes. The initial population is generated randomly, and the construction of new individuals is performed with two operators: cut and splice and thresholding selection. Cut and splice is equivalent to the traditional 1-point crossover, as it switches two sequences of customers to be serviced by two different routes. The goal is to divide customers among vehicles. Then, the routing of the customers for each vehicle is done with Solomon's (1987) cheapest insertion heuristic. Solomon's heuristic is also used to evaluate and check the feasibility of the solution, and create new routes for customers that cannot be included in the same route according to the current division. Thresholding selection is then used to construct a solution by selecting and combining customer sets in the population.

Here more copies are given to strings representing better solutions. A repair algorithm is then used if some customers appear several times in the solution, or some customers are missing.

Rahoual et al. (2001) introduce a multicriteria genetic algorithm, where the evaluation of the individuals is based on a weighted sum of objectives related to violated constraints, number of vehicles and total distance. Solutions are represented as integer strings, describing chronological list of customers for each vehicle. The initial population is generated randomly and crossover is based on 2-opt* (Potvin and Rousseau, 1995), where cutting points are determined randomly. Mutation consists on random reinsertions of customers between routes and the selection is done probabilistically based on the ranks of the individuals. Finally, instead of replacing the preceding population with the current one, the worst individuals in the population are replaced by the best ones found during the search.

Jung and Moon (2002) suggest using the 2D image of a solution for chromosomal cutting within a typical steady-state genetic algorithm. The initial population is created with Solomon’s (1987) I1 insertion heuristic with randomly determined parameter values. Fitness values are based on traveled distance and the selection of parents for mating is performed with the typical binary tournament selection. The recombination is based on dividing the arcs in the selected two solutions in two sets based on different types of curves drawn on the 2D space where customers are located. Then a repair algorithm is used to include missing arcs in a nearest-neighbor manner. In mutation, each route of the offspring is split randomly into at most three routes. After mutation, the offspring is optimized locally using Or-opt (Or, 1976) crossover (swapping end sections of routes, Savelsbergh, 1992) and relocate (moving one customer from one route to another, Savelsbergh,1992) operators.

3.3 Evolution strategies

Homberger and Gehring (1999) propose two evolution strategies for the VRPTW. The individuals of a starting population are generated by means of a stochastic approach that is based on the savings algorithm of Clarke and Wright (1964). In this approach, the stochastic element consists of the random selection of savings elements from the savings list. Selection of the parents is done randomly and only one offspring is created through the recombination of pair of parents. This way, a number $\lambda > \mu$ offspring is created, where μ is the population size. At the end, fitness values are used to select μ offspring for the next generation. The fitness values are based on the number of routes, total travel distance and a criterion that determines how easily the shortest route of the solution (in terms of the number of customers on the route) can be eliminated. The mutation is based on local searches of Or-opt (Or, 1976), 2-opt* (Potvin and Rousseau, 1995) and λ–interchange-move (Osman, 1993) with $\lambda = 1$. In addition, a special Or-opt based operator is used to reduce the number of routes. The first out of the two proposed metaheuristics, evolution strategy ES1, skips the recombination phase. The second evolution strategy, ES2, uses the uniform order-based crossover (Goldberg, 1989) to modify the initially randomly created mutation codes. More specifically, the mutation code consists of customer numbers and the first occurrence of a customer number identifies a removal operation and the second an insertion operation. The mutation code is used to control a set of removal and insertion operators performed by the Or-opt operator. The strategy parameters refer to how often a randomly selected local search operator is applied and to a binary parameter used to alternate the search between minimizing the number of vehicles and total distance.

Mester (2002) has also experimented with evolution strategies similar to Homberger and Gehring's. In the beginning, all customers are served by separate routes. Then, a set of six initial solutions is created using cheapest reinsertions of single customers with varying insertion criteria. The best initial solution obtained is used as a starting point for the ES. The multi-parametric mutation consists of removing a set of customers from a solution randomly, based on the distance to the depot or by selecting one customer from each route. Then, a cheapest insertion heuristic is used to reschedule the removed customers. After mutation, the solution vector is improved using the same three improvement heuristics as in Homberger and Gehring (1999) described above, and, in addition, with the GENIUS heuristic of Gendreau et al. (1992). To solve large-scale problems, a set of strategies for dividing a problem into parts are proposed, and the author concludes that sorting problem data according to time windows improves the initial solution quality.

3.4 Hybrids

In this section we review heuristic search methods that hybridize ideas of evolutionary computation with some other search techniques, such as tabu search or simulated annealing. Most of the hybrids presented in this section use local search mutation instead of the random mutation operators typical for the traditional GAs.

A number of metaheuristics based on a two-phase approach are developed in Thangiah et al. (1994). In the first phase, an initial solution is created by either the cheapest insertion heuristic or the sectoring based genetic algorithm GIDEON (Thangiah, 1995a). The second phase applies one of the following search procedures that use the λ-interchange mechanism: a local search descent procedure, a simulated annealing algorithm or a hybrid simulated annealing and tabu search, where tabu search is combined with the simulated annealing-based acceptance criterion to decide which moves to accept from the candidate list. The main feature of the local search procedures is that infeasible solutions with penalties are allowed if considered attractive.

In Gehring and Homberger (1999) a two-phase approach is introduced: in the first phase, the evolution strategy ES1 of Homberger and Gehring (1999) described earlier is applied with a population size of one to minimize the number of routes; in the second phase, the total distance is minimized using a tabu search algorithm utilizing the same local search operators as ES1. The approach is parallelized using the concept of cooperative autonomy, i.e., several autonomous sequential solution procedures cooperate through the exchange of solutions. The cooperating slave processes are configured in different ways using different seeds for random number generators to create diversity in the search. The authors also develop a new set of larger benchmark problems that are based on Solomon's (1987) benchmark problems.

Gehring and Homberger (2001) introduce some improvements to the parallel method of Gehring and Homberger (1999) described above. First, population size is set greater than one. In the evaluation of individuals, capacity related information is also used to determine the route for elimination. The procedure first identifies the route with the most slack (empty space with respect to capacity). Then the used capacity of all other routes is compared to the (maximum) empty slack. The route whose used capacity least exceeds the capacity constraint of the route with most slack is the route selected for route elimination. Additional improvements include new termination criteria for both phases, where the search is stopped once the minimum number of vehicles required is obtained or no improvement can be found over a certain number of iterations. A single processor implementation of a similar algorithm, using other than capacity criteria to select the routes for removal, is also presented in Homberger and Gehring (2001).

Bräysy et al. (2000) hybridize a genetic algorithm and an evolutionary algorithm consisting of several local search and route construction heuristics inspired from Solomon (1987) and Taillard et al. (1997). In the first phase, a genetic algorithm based on Berger et al. (1998) and Bräysy (1999a) is used to obtain a feasible solution. The algorithm uses a random heuristic to create the initial population, and a Large Neighborhood Search (LNS) based-strategy of Shaw (1998) within the recombination and mutation phase. The evolutionary algorithm used in the second phase picks every combination of two routes in random order and applies randomly one out of four local search operators or route construction heuristics (modified versions of CROSS exchanges (Taillard et al., 1997), Or-Opt (Or, 1976) and insertion heuristics (Solomon, 1987)). Offspring routes generated by these crossover operators are mutated according to a user-defined probability by selecting randomly one out of two new local search operators. Selecting each possible pair of routes, mating and mutation operators are repeatedly applied for a certain number of generations and finally a feasible solution is returned. To escape from a local minimum, arcs longer than average are penalized if they appear frequently during the search (Voudouris, 1997 and Voudouris and Tsang, 1998).

Wee Kit et al. (2001) describe a hybrid genetic algorithm, where a simple tabu search based on cross, exchange, relocate and 2-opt neighborhoods, is applied on individual solutions in the later generations to intensify the search. The genetic algorithm is based on random selection of parent solutions and two new crossover operators. The first operator tries to modify the order of the customers in the first parent by trying to create consecutive pairs of customers according to the second parent. The second crossover operator tries to copy common characteristics of parent solutions to offspring by modifying the seed selection procedure and cost function of an insertion heuristic similar to Solomon' (1987). The authors do not define the mutation and initial solution procedures used.

Bräysy and Dullaert (2003) continue the work of Bräysy et al. (2000). Here the genetic algorithm of the first phase is replaced with a two-stage multi-start local search (Yagiura and Ibaraki, 2001). In the first stage, a set of initial solutions is created using a sequential cheapest insertion heuristic of Bräysy (2003). After creating an initial solution, an attempt is made to reduce the number of routes, using a special ejection chain based technique (Glover, 1991 and 1992). The main innovation of this approach is reordering the routes within the ejection chain to reduce the lateness. In the second phase an evolutionary algorithm is used to minimize the total distance. The differences with respect to the previous study lie in the number and type of crossover and mutation operators used. Here just two crossovers operators are used. The first is an extension of CROSS-exchanges of Taillard (1997), and the other one is similar to the Large Neighborhood Search of Shaw (1998), where the search is restricted to two routes only. Finally, the offspring routes generated by crossover operators are mutated using modified intra-route CROSS-exchanges.

Mester and Bräysy (2003) hybridize the evolution strategies of Mester (2002) with Guided Local Search metaheuristic. The authors call the created new metaheuristic Active Guided Evolution Strategies. It is based on an iterative two-stage procedure, where Guided Local Search is used to regulate a composite local search in the first stage, and the objective function and the neighborhood of the modified Evolution Strategies local search algorithm of Mester (2002) in the second stage. The two stages are repeated iteratively until the stopping criterion is met. The composite local search is based on well-known relocate, 1-interchange and 2-opt* neighborhoods and the initial solution is created with the cheapest insertion heuristic of Mester (2002). After creating the initial solution, an attempt is made to reduce the number of routes with the filling procedure of Bent and Van Hentenryck (2002). Contrary to Mester (2002), the problem decomposition and GENIUS procedures are not used.

Le Bouthillier and Crainic (2003) present a parallel co-operative methodology in which several agents communicate through a pool of feasible solutions. The agents consist of simple construction and local search algorithms and four different metaheuristic methods, namely two evolutionary algorithms and two tabu searches. The evolutionary algorithms use a probabilistic mutation and the well-known edge recombination and order crossovers, while the tabu search procedures are adaptations of the TABUROUTE method of Gendreau et al. (1994) and unified tabu search of Cordeau et al. (2001b). The fitness value of solutions is based on the number of vehicles, distance and waiting times. The pool is initialized with a set of four simple construction heuristics: least successor, double-ended nearest neighbor, multiple fragments (which adds sequentially the shortest arcs) and shortest arc hybridizing (probabilistic version of the previous). The created initial and final solutions are post-optimized with an ejection chain procedure and well-known 2-opt, 3-opt and Or-opt improvement heuristics.

4 Analysis of Results

In this section we present and analyze the results obtained with the above-described evolutionary algorithms to the Solomon’s (1987) 100 customer benchmark problems and to the extended benchmark problems developed by Gehring and Homberger (1999).

4.1 Solomon’s problem instances

Solomon’s (1987) test problems have been the most common way to assess and compare the value of the various heuristic approaches proposed in the literature. These problems have a hundred customers, a central depot, capacity constraints, time windows on the time of delivery, and a total route time constraint. The C1 and C2 classes have customers located in clusters and in the R1 and R2 classes the customers are at random positions. The RC1 and RC2 classes contain a mix of both random and clustered customers. Each class contains between 8 and 12 individual problem instances and all problems in any one class have the same customer locations and the same vehicle capacities; only the time windows differ. In terms of time window density (the percentage of customers with time windows), the problems have 25%, 50%, 75%, and 100% time windows. The C1, R1 and RC1 problems have a short scheduling horizon and require 9 to 19 vehicles. Short horizon problems have vehicles that have small capacities and short route times, and therefore cannot service many customers. Classes C2, R2 and RC2 are more representative of "long-haul" delivery with longer scheduling horizons and fewer (2–4) vehicles. Both travel time and distance are given by the Euclidean distance between points.

The results are usually ranked according to a hierarchical objective function, where the number of vehicles is considered as the primary objective, and for the same number of vehicles, the secondary objective is often either total traveled distance or total duration of routes. Therefore, a solution requiring fewer routes is always considered better than a solution with more routes, regardless of the total traveled distance.

The evolutionary algorithms described above are compared in Table 1 against some of the other recent metaheuristics by Rochat and Taillard (1995), Taillard et al. (1997), Liu and Shen (1999a), Gambardella et al. (1999), Cordeau et al. (2001b) and Rousseau et al. (2002). Rochat and Taillard (1995) presented a probabilistic tabu search that uses adaptive memory to record and combine the best routes produced during the search. Taillard et al. (1997) also describe a tabu search with adaptive memory, but employ a different neighborhood structure, based on exchanges of consecutive customers (or segments) between routes. Liu and Shen (1999a) developed a route-neighborhood-based metaheuristic that constructs routes in nested parallel manner, and uses several different improvement heuristics. Gambardella et al. (1999) developed an ant colony system consisting of two separate ant colonies with different objectives, guiding a well-known nearest neighbor heuristic for solution construction, and the CROSS-exchanges of Taillard et al. (1997) for improvement. Cordeau et al. (2001b) introduce a tabu search based on simple customer relocations and allowance of infeasible solutions during the search process, and Rousseau et al. (2002) used a variable neighborhood descent scheme and new large-neighborhood operators within a constraint-programming framework.

Table 1: Comparison of evolutionary algorithms and other metaheuristics.

Reference	R1	R2	C1	C2	RC1	RC2	CNV/CTD	TIME
Tan et al.	**14.42**	**5.64**	**10.11**	**3.25**	**14.63**	**7.00**	**525**	Pentium II 266 MHz,
(2000)	1314.79	1093.37	860.62	623.47	1512.94	1282.47	62901	–, –
Tan et al.	**13.17**	**5.00**	**10.11**	**3.25**	**13.50**	**5.00**	**478**	Pentium II 330 MHz,
(2001a)	1227	980	861	619	1427	1123	58605	–, 25 min.
Tan et al.	**12.91**	**5.0**	**10.00**	**3.00**	**12.60**	**5.80**	**471**	Pentium II 330 MHz,
(2001b)	1205.0	929.6	841.96	611.2	1392.3	1080.1	56931	–, 25 min.
Jung and Moon	**13.25**	**5.36**	**10.00**	**3.00**	**13.00**	**6.25**	**486**	Pentium III 1 GHz,
(2002)	1179.95	878.41	828.38	589.86	1343.64	1004.21	54779	100 runs, 0.8 min.
Thangiah et al.	**12.33**	**3.00**	**10.00**	**3.00**	**12.00**	**3.38**	**418**	NeXT 25 MHz,
(1994)	1227.42	1005.00	830.89	640.86	1391.13	1173.38	58905	–, 30 min.
Thangiah	**12.75**	**3.18**	**10.00**	**3.00**	**12.50**	**3.38**	**429**	Solbourne 5/802,
(1995a)	1300.25	1124.28	892.11	749.13	1474.13	1411.13	65074	–, 2.1 min.
Potvin and Bengio	**12.58**	**3.00**	**10.00**	**3.00**	**12.13**	**3.38**	**422**	Sun Sparc 10,
(1996)	1296.83	1117.64	838.11	590.00	1446.25	1368.13	62634	–, 25 min.
Berger et al.	**12.58**	**3.09**	**10.00**	**3.00**	**12.13**	**3.50**	**424**	Sun Sparc 10,
(1998)	1261.58	1030.01	834.61	594.25	1441.35	1284.25	60539	–, 1–10 min.
Bräysy	**12.58**	**3.09**	**10.00**	**3.00**	**12.13**	**3.38**	**423**	Sun Ultra Enterprise
(1999)	1272.34	1053.65	857.64	624.31	1417.05	1256.80	60962	450, 5 runs, 17 min.
Homberger and Gehring	**11.92**	**2.73**	**10.00**	**3.00**	**11.63**	**3.25**	**406**	Pentium 200 MHz,
(1999)	1228.06	969.95	828.38	589.86	1392.57	1144.43	57876	10 runs, 13 min.
Gehring and Homberger	**12.42**	**2.82**	**10.00**	**3.00**	**11.88**	**3.25**	**415**	4×Pentium 200 MHz,
(1999)	1198	947	829	590	1356	1140	56942	1 run, 5 min.
Bräysy et al.	**12.42**	**3.09**	**10.00**	**3.00**	**12.13**	**3.38**	**421**	Celeron 366 MHz,
(2000)	1213.86	978.00	828.75	591.81	1372.20	1170.23	57857	5 runs, 15 min.
Gehring and Homberger	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**	4×Pentium 400 MHz,
(2001)	1217.57	961.29	828.63	590.33	1395.13	1139.37	57641	5 runs, 15.1 min.
Homberger and Gehring	**11.92**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**	Pentium 400 MHz,
(2001)	1212.73	955.03	828.38	589.86	1386.44	1123.17	57309	–, –
Wee Kit et al.	**12.58**	**3.18**	**10.00**	**3.00**	**12.75**	**3.75**	**432**	Digital Workstation
(2001)	1203.32	951.17	833.32	593.00	1382.06	1132.79	57265	433a, –, 147.4 min.
Rahoual et al.	**12.92**	—	**10.00**	—	**12.63**	—	—	—
(2001)	1362		887		1487			
Mester	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**	Pentium III 450 MHz,
(2002)	1208	954	829	590	1387	1119	57219	–, 150.2 min.
Bräysy and Dullaert	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**	AMD 700 MHz,
(2003)	1220.14	977.57	828.38	589.86	1397.44	1140.06	57870	3 runs, 9.1 min.
Le Bouthillier and Crainic	**12.08**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**407**	5×Pentium 850 MHz,
(2003)	1209.19	963.62	828.38	589.86	1389.22	1143.70	57412	1 run, 12 min.
Berger et al.	**11.92**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**	Pentium 400 MHz,
(2003)	1221.10	975.43	828.48	589.93	1389.89	1159.37	57952	–, 30 min.
Rochat and Taillard	**12.25**	**2.91**	**10.00**	**3.00**	**11.88**	**3.38**	**415**	Silicon Graphics
(1995)	1208.50	961.72	828.38	589.86	1377.39	1119.59	57231	100 MHz, –,–
Taillard et al.	**12.17**	**2.82**	**10.00**	**3.00**	**11.50**	**3.38**	**410**	Sun Sparc 10, –,–
(1997)	1209.35	980.27	828.38	589.86	1389.22	1117.44	57523	
Liu and Shen	**12.17**	**2.82**	**10.00**	**3.00**	**11.88**	**3.25**	**412**	HP 9000/720,
(1999a)	1249.57	1016.58	830.06	591.03	1412.87	1204.87	59318	3 runs, 20 min.
Gambardella et al.	**12.00**	**2.73**	**10.00**	**3.00**	**11.63**	**3.25**	**407**	Sun Ultrasparc 1,
(1999)	1217.73	967.75	828.38	589.86	1382.42	1129.19	57525	167 MHz, –,–
Cordeau et al.	**12.08**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**407**	Sun Ultra 2
(2001)	1210.14	969.57	828.38	589.86	1389.78	1134.52	57556	300 MHz, –,–
Rousseau et al.	**12.08**	**3.00**	**10.00**	**3.00**	**11.63**	**3.38**	**412**	Sun Ultra 10,
(2002)	1210.21	941.08	828.38	589.86	1382.78	1105.22	56953	10 runs, 183.3 min.

The first column to the left lists the authors. Columns R1, R2, C1, C2, RC1 and RC2 give the average number of vehicles and average total distance with respect to six problem groups of

Solomon (1987). The CNV/CTD column indicates the cumulative number of vehicles (CNV) and cumulative total distance (CTD) over all 56 test problems. Finally, the rightmost column describes the computer, number of independent runs, and the CPU time, as reported by the authors. The Table is divided into three parts. The first two parts present the reviewed evolutionary algorithms and the third results of the other recent metaheuristics. The evolutionary algorithms are divided into two parts according to the objective function used. In the top of Table 1, results of four genetic algorithms that consider the total distance as the only objective are shown. For other methods in the two lower parts of the Table, a hierarchical objective function is used, where the number of routes is considered as the primary objective and, for the same number of routes, the secondary objective is to minimize the total traveled distance. An exception is found in Potvin and Bengio (1996), where the second objective is to minimize the total duration of routes. All algorithms in Table 1, except Liu and Shen (1999a), are stochastic and most of the methods are implemented in C. The only exceptions are Wee Kit et al. (2001) and Mester (2002) that are coded in Java and Visual Basic, respectively. Tan et al. (2000) do not report the programming language used.

According to Table 1, it seems that the evolution strategies by Homberger and Gehring (1999) and Mester (2002), hybrids of evolution strategies and tabu search by Gehring and Homberger (2001) and Homberger and Gehring (2001), and the genetic algorithm of Berger et al. (2003) produce the best results in terms of the hierarchical objective function. Homberger and Gehring (2001) reports the best results for R1 and RC1, while Mester (2002) performs best in R2 and RC2. For clustered problem groups C1 and C2 most of the recent papers report optimal solutions. The differences between the best methods in terms of solution quality are small, only less than 0.5% in the cumulative number of vehicles (CNV) and about 1% in cumulative total distance (CTD). These evolutionary algorithms also outperform all other recent metaheuristics. Berger et al. (2003) and Homberger and Gehring (2001) are able to obtain the lowest known cumulative number of vehicles, 405, and the distance values are very close to the best known. The maximum difference in the number of vehicles is about 6% among the algorithms with hierarchical objective, while the greatest difference in CTD is about 13.7%. According to Table 1, the results of the traditional genetic algorithm of Thangiah (1995a) are inferior with respect to recent genetic algorithms, evolution strategies and hybrids. In general, different hybrids, combining well-known route construction and improvement heuristics, seem to show the best performance. As for the distance optimization, it seems that the recent genetic algorithm of Jung and Moon (2002) performs best. The differences with other competing approaches using the same objective vary from 3.9 to 14.8%. The difference in CTD with respect to Homberger and Gehring (2001) is 4.6%, while Jung and Moon (2002) require 20% more vehicles, which clearly illustrates the conflicting nature of route number and distance optimization and the importance of short or long run cost minimization. Distance values are thus comparable only for an equal number of vehicles. Since only a few of the authors report the number of runs required to obtain the reported results, it is impossible to draw any final conclusions regarding which method performs best. Another comparison is provided in Table 2, where only results obtained with limited computational effort are considered.

We consider in Table 2 only results for which the CPU time consumption and the number of runs are described. To facilitate the comparison, the effect of different hardware is normalized to equal Sun Sparc 10 using Dongarra's (1998) factors. In addition, if the reported results are the best ones over multiple experiments, we multiplied the computation times by this number to account for the real computational effort. The number of runs is greater than one only if the reported result is the best over multiple executions of the given algorithm and the CPU time is reported only for a single run. Two CPU time values are described in the TIME column: the one reported by the authors and in the parenthesis the normalized CPU time. Details for calculating these normalized CPU times can be found in Bräysy (2001).

Table 2: Comparison of results obtained with limited computational effort for Solomon's instances.

Reference	R1	R2	C1	C2	RC1	RC2	CV/CD	TIME
Homberger and Gehring	**11.92**	**2.73**	**10.00**	**3.00**	**11.63**	**3.25**	**406**	Pentium 200 MHz,
(1999)	1228.06	969.95	828.38	589.86	1392.57	1144.43	57876	10 runs, 13 (312) min.
Gehring and Homberger	**12.42**	**2.82**	**10.00**	**3.00**	**11.88**	**3.25**	**415**	4×Pentium 200 MHz,
(1999)	1198	947	829	590	1356	1144	56946	1 run, 5 (48) min.
Bräysy	**12.58**	**3.09**	**10.00**	**3.00**	**12.13**	**3.38**	**423**	Sun Ultra Enterprise 450,
(1999)	1272.34	1053.65	857.64	624.31	1417.05	1256.80	60962	5 runs, 17 (374) min.
Bräysy et al.	**12.42**	**3.09**	**10.00**	**3.00**	**12.13**	**3.38**	**421**	Celeron 366 MHz,
(2000)	1213.86	978.00	828.75	591.81	1372.20	1170.23	57857	5 runs, 15 (360) min.,
Gehring and Homberger	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**	4×Pentium 400 MHz,
(2001)	1217.57	961.29	828.63	590.33	1395.13	1139.37	57641	5 runs, 13.5 (1458) min.
Homberger and Gehring	**12.08**	**2.82**	**10.00**	**3.00**	**11.50**	**3.25**	**408**	Pentium 400 MHz,
(2001)	1211.67	950.72	828.45	589.96	1395.93	1135.09	57422	5 runs, 17.5 (473) min.
Le Bouthillier and Crainic	**12.08**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**407**	5×Pentium 850 MHz,
(2003)	1209.19	963.62	828.38	589.86	1389.22	1143.70	57412	1 run, 12 (702) min.
Berger et al.	**12.17**	**2.73**	**10.00**	**3.00**	**11.88**	**3.25**	**411**	Pentium 400 MHz,
(2003)	1251.40	1056.59	828.50	590.06	1414.86	1258.15	60200	1 run, 30 (162) min.
Bräysy and Dullaert	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**	AMD 700 MHz,
(2003)	1220.14	977.57	828.38	589.86	1397.44	1140.06	57870	3 runs, 9.1 (374) min.
Rochat and Taillard	**12.58**	**3.09**	**10.00**	**3.00**	**12.38**	**3.62**	**427**	Silicon Graphics 100,
(1995)	1197.42	954.36	828.45	590.32	1369.48	1139.79	57120	1 run, 92.2 (138) min.
Taillard et al.	**12.33**	**3.00**	**10.00**	**3.00**	**11.90**	**3.38**	**417**	Sun Sparc 10,
(1997)	1220.35	1013.35	828.45	590.91	1381.31	1198.63	58614	1 run, 248 (248) min.
Liu and Shen	**12.17**	**2.82**	**10.00**	**3.00**	**11.88**	**3.25**	**412**	HP 9000/720,
(1999a)	1249.57	1016.58	830.06	591.03	1412.87	1204.87	59318	3 runs, 20 (102) min.
Gambardella et al.	**12.38**	**3.00**	**10.00**	**3.00**	**11.92**	**3.33**	**418**	Sun Ultrasparc 1,
(1999)	1210.83	960.31	828.38	591.85	1388.13	1149.28	57583	1 run, 30 (210) min.

According to Table 2, the algorithms by Homberger and Gehring (1999), Gehring and Homberger (2001) and Bräysy and Dullaert (2003) show the best overall performance. It is difficult to name the best method: Gehring and Homberger (2001) report the best solution quality, but their procedure is clearly slower than the others. The evolutionary algorithms in Table 2 seem to outperform also the other best performing metaheuristics in terms of efficiency. Regarding individual problem groups, practically all approaches yield excellent results to problem groups C1 and C2 with clustered customers. For other problem groups, the best results are obtained with the evolution strategies and hybrids of Homberger and Gehring. For detailed best-known results to Solomon’s benchmarks, we refer the reader to the web site http://www.top.sintef.no/.

4.2 Gehring and Homberger problem instances

The Gehring and Homberger (1999) extended benchmark problems were constructed similar to the 100 customer problem instances by Solomon (1987). They consist of 5 sets of 200, 400, 600, 800 and 1000 customers, with 60 instances in each set.

In Table 3, we present results for the 300 extended problems developed by Gehring and Homberger (1999). For each problem size of 200, 400, 600, 800 and 1000 customers, the Table shows the cumulative number of vehicles (CNV), cumulative total distance (CTD) and total time consumption in minutes (Cumulative Computer Time, CCT). This total time consumption was normalized to Sun Sparc 10, as in Table 2. So far, only a few authors have tackled these extended benchmarks. In Table 3, we consider all methods we are aware of reporting results and the computational effort to solve all extended problems. These methods are hybrids of evolution strategies and tabu search by Gehring and Homberger (1999, 2001) and Homberger and Gehring (2001), the hybrid of evolution strategies and guided local search by Mester and Bräysy (2003), the hybrid of two evolutionary algorithms and a tabu search by Le Bouthillier and Crainic (2003), the simulated annealing of Li and Lim (2003a), and the multi-start local search of Bräysy et al. (2003). All methods consider a hierarchical objective function. According to this objective, Mester and Bräysy (2003) report the best results to 200-, 400- and 1000-customer problems, whereas Gehring and Homberger (2001) shows the best performance for 600- and 800-customer problems. These two evolutionary algorithms also seem to outperform the methods of Li and Lim (2003a) and Bräysy et al. (2003) but they require a considerable computational effort.

Table 3: Comparison of results for the extended benchmark problems of Gehring and Homberger (1999).

Reference		200	400	600	800	1000
Gehring and Homberger (1999)	CNV	**694**	**1390**	**2082**	**2770**	**3461**
	CTD	176, 180	412, 270	867, 010	1, 515, 120	2, 276, 390
	CCT	*96*	*192*	*288*	*384*	*480*
Gehring and Homberger (2001)	CNV	**696**	**1392**	**2079**	**2760**	**3446**
	CTD	179, 328	428, 489	890, 121	1, 535, 849	2, 290, 367
	CCT	*136*	*460*	*836*	*1503*	*1950*
Homberger and Gehring (2001)	CNV	**699**	**1397**	**2088**	**2773**	**3459**
	CTD	180, 602	431, 089	890, 293	1, 516, 648	2, 288, 819
	CCT	*26*	*83*	*167*	*295*	*497*
Le Bouthillier and Crainic (2003)	CNV	**694**	**1390**	**2088**	**2766**	**3451**
	CTD	173 061	408 281	836 261	1475 281	2225 366
	CCT	*650*	*1300*	*1950*	*2600*	*3250*
Mester and Bräysy (2003)	CNV	**694**	**1389**	**2082**	**2765**	**3446**
	CTD	168 573	390 386	796 172	1361 586	2078 110
	CCT	*256*	*544*	*1280*	*4640*	*19 200*
Li and Lim (2003a)	CNV	**707**	**1414**	**2112**	**2802**	**3490**
	CTD	172, 472	405, 656	843, 320	1, 416, 531	2, 176, 398
	CCT	*3606*	*7124*	*7916*	*10, 155*	*12, 005*
Bräysy et al. (2003)	CNV	**695**	**1391**	**2084**	**2776**	**3465**
	CTD	172, 406	399, 132	810, 662	1, 384, 306	2, 133, 376
	CCT	*98*	*322*	*661*	*1069*	*1616*

If one analyzes the five methods showing the best performance in Tables 1–3 (Homberger and Gehring, 2001; Berger et al., 2003; Gehring and Homberger, 2001; Bräysy and Dullaert, 2003; Mester, 2002), it must be noted that they all hybridize several well-known search techniques for routing problems and are rather complex to implement. Three of these methods are based on evolution strategies. Berger et al. (2003) call their method a hybrid genetic algorithm, but the search is mainly driven by the mutation consisting of several well-known construction and improvement routing heuristics and metaheuristics. Bräysy and Dullaert's (2003) method can be interpreted as an iterative multi-phase and multi-start local search, consisting of several move operators. All five methods work on a population of several feasible solutions, except for Bräysy and Dullaert (2003) who define each feasible route as an individual. The population size is very small in all cases (1–30) compared to the traditional genetic algorithms, and instead of random sampling, the initial population is created with well-known construction heuristics. The search is mainly driven by mutation in all other methods, except for Bräysy and Dullaert (2003), and the interaction between the solutions in the population is thus rather small. All methods use a set of several previously developed improvement heuristics, based on a small neighborhood. All methods have also special operators and/or strategies for reducing the number of routes.

5 Conclusions

NP-hardness of the vehicle routing problem requires heuristic solution strategies for most real-life instances. In the previous sections, we have comprehensively surveyed the evolutionary algorithms for the vehicle routing problem with time windows. To summarize, it seems that it is important to include special methods for route reduction, combine several different search methods, employ some improvement heuristics, and create a high quality initial population. Small neighborhood and population size seem to result in efficient implementations. Finally, operating at the level of feasible solutions works well, and interaction between the solutions does not seem to be crucial to get good results. Currently, genetic algorithm of Berger et al. (2003), the evolution strategies of Mester (2002) and hybridizations of evolution strategies with tabu search and guided local search by Homberger and Gehring (2001) and Mester and Bräysy (2003), respectively, seem to achieve best performance. These evolutionary algorithms also outperform any other heuristic or metaheuristic approach we are aware of.

Acknowledgements

This work was supported in part by the Emil Aaltonen Foundation, Liikesivistysrahasto Foundation, the Canadian Natural Science and Engineering Research Council and the TOP program funded by the Research Council of Norway. This support is gratefully acknowledged.

6 References

Alander, J.T. (2000). “An Indexed Bibliography of Genetic Algorithms in Operations Research.” Technical Report 94-1-OR. University of Vaasa, Finland.

Angelelli, E. and R. Mansini (2001). “The Vehicle Routing Proble with Time Windows and Simultaneous Pick-up and Delivery.” In: Klose, A., Speranza, M.G. and L.N. Van Wassenhove (eds.) *Quantitative Approaches to Distribution Logistics and Supply Chain Management*. Lectures Notes in Economics and Mathematical Systems. New York: Springer-Verlag, pp. 249–268.

Baker, J.E. (1985). “Adaptive Selection Methods for Genetic Algorithms.” In J.J. Grefenstette (ed.) *Proceedings of an International Conference on Genetic Algorithms and their Applications*. New York: Lawrence Erlbaum Associates, pp. 101–111.

Bent R and P. Van Hentenryck (2002). ”A two-stage hybrid local search for the vehicle routing problem with time windows.” To Appear in *Transportation Science*.

Benyahia, I. and J.-Y. Potvin. (1995). “Generalization and Refinement of Route Construction Heuristics Using Genetic Algorithms.” In *Proceedings of the 1995 IEEE International Conference on Evolutionary Computation*. Pistacaway: IEEE Service Center, pp. 39–43.

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*, 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 Genetic and Evolutionary Computation Conference*. San Francisco: Morgan Kaufman Publishers, pp. 44–51.

Berger, J., M. Barkaoui and O. Bräysy. (2003). “A Route-directed Hybrid Genetic Approach for the Vehicle Routing Problem with Time Windows”, *Information Systems and Operations Research* 41, 179-194.

Blanton, J.L. and R.L. Wainwright. (1993). “Multiple Vehicle Routing with Time and Capacity Constraints Using Genetic Algorithms.” In *Proceedings of the Fifth International Conference on Genetic Algorithms,* San Mateo: Morgan Kaufmann Publishers, pp. 452–459.

Bramel, J. and D. Simchi-Levi. (1996). “Probabilistic Analyses and Practical Algorithms for the Vehicle Routing Problem with Time Windows.” *Operations Research* 44, 501–509.

Bräysy, O. (1999). “A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows.” Licentiate thesis, University of Vaasa, Finland.

Bräysy, O. (1999). “A New Algorithm for the Vehicle Routing Problem with Time Windows Based on the Hybridization of a Genetic Algorithm and Route Construction Heuristics.” *Proceedings of the University of Vaasa, Research papers* 227.

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

Bräysy, O. (2001). “Local Search and Variable Neighborhood Search Algorithms for the Vehicle Routing Problem with Time Windows.” Doctoral Dissertation, University of Vaasa, Finland.

Bräysy, O. and M. Gendreau. (2002a). “Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms”, To appear in *Transportation Science*.

Bräysy, O. and M. Gendreau. (2002b). “Tabu Search Heuristics for the Vehicle Routing Problem with Time Windows”, *Top* 10, 211-238.

Bräysy O., M. Gendreau, G. Hasle and A. Lokketangen. (2002). “A Survey of Rich Vehicle Routing Models and Heuristic Solution Techniques.” Working Paper, SINTEF Applied Mathematics, Department of Optimization, Norway. Presented at Optimization Days 2002, Montreal, Canada.

Bräysy, O. (2003). “A Reactive Variable Neighborhood Search for the Vehicle Routing Problem with Time Windows.” *INFORMS Journal on Computing* 15, 347-368.

Bräysy, O. and W. Dullaert (2003). “A Fast Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows.” *International Journal on Artificial Intelligence Tools*, 12, 153-172.

Bräysy, O., G. Hasle and W. Dullaert. (2003). “A Multi-Start Local Search Algorithm for the Vehicle Routing Problem with Time Windows”, To Appear in *European Journal of Operational Research*.

Bäck, T., D.B. Fogel and Z. Michalewicz. (1997). *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press.

Campbell A., L. Clarke and M. Savelsbergh. (2001). Inventory Routing in Practice. In: Toth P. and Vigo D. (eds.), *The Vehicle Routing Problem*. Philadelphia: SIAM, pp. 309–330.

Caseau, Y. and F. Laburthe. (1999). “Heuristics for Large Constrained Vehicle Routing Problems.” *Journal of Heuristics* 5, 281–303.

Clarke, G. and J.W. Wright. (1964). “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points.” *Operations Research* 12, 568–581.

Cook, W. and J.L. Rich. (1999). “A Parallel Cutting-Plane Algorithm for the Vehicle Routing Problems with Time Windows.” Working Paper, Department of Computational and Applied Mathematics, Rice University, Houston, U.S.A.

Cordeau, J.-F., G. Desaulniers, J. Desrosiers, M.M. Solomon and F. Soumis. (2001a). “The VRP with Time Windows.” In P. Toth and D. Vigo (eds.), *The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications*. Philadelphia: SIAM, pp. 157–194.

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

Cordone, R. and R. Wolfler-Calvo. (2001). “A Heuristic for the Vehicle Routing Problem with Time Windows.” *Journal of Heuristics* 7, 107–129.

Crainic, T. and G. Laporte. (1997). “Planning Models for Freight Transportation.” *European Journal of Operational Research* 97, 409–438.

Czech, Z.J. and P. Czarnas. (2002). “Parallel Simulated Annealing for the Vehicle Routing Problem with Time Windows.” In *Proceedings of 10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing*. Canary Islands, Spain, pp. 376–383.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection*. London: Murray.

De Backer, B. and V. Furnon. (1997). “Meta-Heuristics in Constraint Programming Experiments with Tabu Search on the Vehicle Routing Problem.” Presented at the Second International Conference on Metaheuristics, Sophia Antipolis, France.

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

Desrochers, M., J.K. Lenstra, M.W.P. Savelsbergh and F. Soumis. (1988). “Vehicle Routing with Time Windows: Optimization and Approximation.” In B. Golden and A. Assad, (eds.), *Vehicle Routing*: *Methods and Studies*. Amsterdam: Elsevier Science Publishers, pp. 65–84.

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

Dongarra, J. (1998). “Performance of Various Computers Using Standard Linear Equations Software.” Technical Report CS-89-85, Department of Computer Science, University of Tennessee, U.S.A.

Duhamel C., J.-Y. Potvin and J.-M. Rousseau. (1997). “A Tabu Search Heuristic for the Vehicle Routing Problem with Backhauls and Time Windows.” *Transportation Science* 31, 49–59.

Dullaert, W., G.K. Janssens, K. Sörensen and B. Vernimmen. (2002). “New heuristics for the fleet size and mix vehicle routing problem with time windows”, *Journal of the Operational Research Society*, 53, pp. 1232-1238.

Dullaert, W. and O. Bräysy (2003). “Routing relatively few customers per route.” *TOP* 11, 325-336.

Fogel, L., A. Owens and M. Walsh. (1966). *Artificial Intelligence Through Simulated Evolution*. Chichester: John Wiley & Sons.

Fogel, D.B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. New York: IEEE Press.

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

Gehring, H. and J. Homberger. (1999). “A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows.” In *Proceedings of EUROGEN99*. Jyväskylä: University of Jyväskylä, pp. 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.

Gendreau, M., A. Hertz and G. Laporte. (1992). ”A New Insertion and Post-Optimization Procedures for the Traveling Salesman Problem.” *Operations Research* 40, 1086–1093.

Gendreau, M., A. Hertz and G. Laporte. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science* 40, 1276–1290.

Gendreau, M., F. Guertin, J.-Y. Potvin and E. Taillard. (1999). “Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching.” *Transportation Science* 33, 381–390.

Glover F. (1986). “Future Paths for Integer Programming and Links to Artificial Intelligence.” *Computers and Operations Research* 13, 533–549.

Glover, F. (1991). “Multi-Level Tabu Search and Embedded Search Neighborhoods for the Travelling Salesman Problem.” Working Paper, College of Business & Administration, University of Colorado, Boulder.

Glover, F. (1992). New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems. In O. Balchi, S. Ramesh and S.A. Zenios (eds.), *Computer Science and Operations Research: New Developments in Their Interfaces*. Oxford: Pergamon Press, pp. 449–509.

Glover F. and M. Laguna (1997). *Tabu Search*. New York: Kluwer Academic Publishers.

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

Goldberg, D., B. Korb and K. Deb. (1989). “Messy Genetic Algorithms: Motivation, Analysis and First Results.” *Complex Systems* 3, 493–530.

Golden, B.L. and W. R. Stewart. (1985). “Empirical Analysis of Heuristics.” In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.), *The Traveling Salesman Problem*. Chichester: John Wiley & Sons, pp. 207–249.

Golden, B.L. and A.A. Assad. (1986). “Perspectives on Vehicle Routing: Exciting New Developments.” *Operations Research* 34, 803–809.

Golden, B.L. and E.A. Wasil. (1987). “Computerized Vehicle Routing in the Soft Drink Industry.” *Operations Research* 35, 6–17.

Golden, B.L. and A.A. Assad. (1988). *Vehicle Routing*: *Methods and Studies*. Amsterdam: Elsevier Science Publishers.

Halse, K. (1992). “Modeling and Solving Complex Vehicle Routing Problems.” Ph.D. thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

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

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

Homberger, J. and H. Gehring. (2001). “A Two-Phase Hybrid Metaheuristic for the Vehicle Routing Problem with Time Windows.” Working Paper, Department of Economics, University of Hagen, Germany.

Ioannou, G., M. Kritikos and G. Prastacos. (2001). “A Greedy Look-Ahead Heuristic for the Vehicle Routing Problem with Time Windows.” *Journal of the Operational Research Society* 52, 523–537.

Jung, S. and B.-R. Moon. (2002). “A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows.” In *Proceedings of Genetic and Evolutionary Computation Conference*. San Francisco: Morgan Kaufmann Publishers, pp. 1309–1316.

King, G.F. and C. F. Mast. (1997). “Excess Travel: Causes, Extent and Consequences.” *Transportation Research Record* 1111, 126–134.

Kirkpatrick, S., C.D. Gelatt and P.M. Vecchi. (1983). “Optimization by Simulated Annealing.” *Science* 220, 671–680.

Larsen, J. (1999). “Parallelization of the Vehicle Routing Problem with Time Windows.” Ph.D. thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Lau, H.C. and Q.Z. Liu. (1999). “Collaborative Model and Algorithms for Supporting Real-Time Distribution Logistics Systems.” Working Paper, School of Computing, National University of Singapore.

Le Bouthillier, A. and T.G. Crainic (2003). “Co-operative Parallel Meta-Heuristic for the Vehicle Routing Problems with Time Windows.” Working paper, University of Montreal, Montreal, Canada.

Li, H. and A. Lim (2003a). “Local Search with Annealing-Like Restarts to Solve the VRPTW.” *European Journal of Operational Research* 150, 115–127.

Li, H. and A. Lim (2003b). “A Metaheuristics for the Pickup and Delivery Problem with Time Windows.” *International Journal on Artificial Intelligence Tools* 12, 173–186.

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

Liu, F.-H. and S.-Y. Shen. (1999b). “The Fleet Size and Mix Vehicle Routing Problem with Time Windows.” *Journal of the Operational Research Society* 50, 721–732.

Lund K., O.B.G. Madsen and J.M. Rygaard. (1996). “Vehicle Routing Problems with Varying Degree of Dynamism.” Technical Report, The Department of Mathematical Modelling, Technical University of Denmark.

Mester, D. (2002). “An Evolutionary Strategies Algorithm for Large Scale Vehicle Routing Problem with Capatitate and Time Windows Restrictions.” Working Paper, Institute of Evolution, University of Haifa, Israel.

Mester, D. and O. Bräysy (2003). “Active Guided Evolution Strategies for Large-scale Vehicle Routing Problems with Time Windows.” To Appear in *Computers & Operations Research*.

Metropolis, W., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. (1953). “Equation of the State Calculations by Fast Computing Machines.” *Journal of Chemical Physics* 21, 1087–1092.

Miettinen K., M. Mäkelä, P. Neittaanmäki and J. Périaux. (1999). *Evolutionary Algorithms in Engineering and Computer Science*. Chichester: John Wiley & Sons.

Moscato, P. (1989). “On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Toward Memetic Algorithms.” Technical Report 790, California Institute of Technology, Pasadena.

Moscato, P. (1999). “Memetic Algorithms: A Short Introduction.” In D. Corne, M. Dorigo and F. Glover (eds.), *New Ideas in Optimization*. London: McGraw-Hill, pp. 219–243.

Nanry W.P. and J.W. Barnes. (2000). “Solving the Pickup and Delivery Problem with Time Windows using Reactive Tabu Search.” *Transportation Research, Part B* 34, 107–121.

Or, I. (1976). “Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking.” Ph.D. thesis, Northwestern University, Evanston, Illinois.

Osman, I.H. (1993). “Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problems.” *Annals of Operations Research* 41, 421–452.

Potvin, J.-Y. and J.-M. Rousseau. (1993). “A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows.” *European Journal of Operational Research* 66, 331–340.

Potvin, J.-Y. and J.-M. Rousseau. (1995). “An Exchange Heuristic for Routeing Problems with Time Windows.” *Journal of the Operational Research Society* 46, 1433–1446.

Potvin, J.-Y. and C. Robillard. (1995). “Clustering for Vehicle Routing with a Competitive Neural Network.” *Neurocomputing* 8, 125–139.

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.

Potvin, J.-Y., D. Dubé and C. Robillard. (1996). "A Hybrid Approach to Vehicle Routing Using Neural Networks and Genetic Algorithms." *Applied Intelligence* 6, 241-252.

Rahoual, M., B. Kitoun, M.-H. Mabed, V. Bachelet and F. Benameur. (2001). "Multicriteria Genetic Algorithms for the Vehicle Routing Problem with Time Windows." Presented at MIC'2001, 4th Metaheuristic International Conference, Porto, Portugal.

Rechenberg, I. (1973). *Evolutionsstrategie*. Stuttgart: Fromman-Holzboog.

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

Russell, R.A. (1995). "Hybrid Heuristics for the Vehicle Routing Problem with Time Windows." *Transportation Science* 29, 156–166.

Savelsbergh, M.W.P. (1992). The vehicle routing problem with time windows: minimizing route duration. *Journal on Computing* 4, 146–154.

Schwefel, H.-P. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutions-strategie*. Basel: Birkhäuser.

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

Shieh, H.M. and M.D. May. (1998). "On-line Vehicle Routing with Time Windows: Optimization-Based Heuristics Approach for Freight Demands Requested in Real-Time." *Transportation Research Record* 1617, 171–178.

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

Solomon, M.M. and J. Desrosiers. (1988). "Time Window Constrained Routing and Scheduling Problems." *Transportation Science* 22, 1–13.

Taillard, E., 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.Q. Zhu. (2000). "Heuristic Methods for Vehicle Routing Problem with Time Windows." Presented at the *6th International Symposium on Artificial Intelligence & Mathematics*, Ft Lauderdale, Florida.

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

Tan, K.C., T.H. Lee, K. Ou and L.H. Lee. (2001b). “A Messy Genetic Algorithm for the Vehicle Routing Problem with Time Window Constraints.” In *Proceedings of the 2001 Congress on Evolutionary Computation*. Pistacaway: IEEE Service Center, pp. 679–686.

Thangiah, S., I. Osman and T. Sun. (1994). “Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows.” Technical Report UKC/IMS/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury.

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

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

Thangiah, S., 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.R., J.-Y. Potvin and T. Sun. (1996). “Heuristic Approaches to Vehicle Routing with Backhauls and Time Windows.” *Computers & Operations Research* 23, 1043–1057.

Thompson, P. M. and H.N. Psaraftis. (1993). “Cyclic Transfer Algorithms for Multivehicle Routing and Scheduling Problems.” *Operations Research* 41, 935–946.

Voudouris, C. (1997). *Guided Local Search for Combinatorial Problems*. Ph.D. thesis, University of Essex, Colchester, UK.

Voudouris, C. & E. Tsang (1998). Guided local search. *European Journal of Operations Research* 113, 80–119.

Wee Kit, H., J. Chin and A. Lim. (2001). “A Hybrid Search Algorithm for the Vehicle Routing Problem with Time Windows.” *International Journal on Artificial Intelligence Tools* 10, 431–449.

Yagiura, M. and T. Ibaraki. (2001). “On Metaheuristic Algorithms for Combinatorial Optimization Problems.” *Systems and Computers in Japan* 32, 33–55.

Zhu, K.Q. (2000). “A New Genetic Algorithm for VRPTW.” Presented at IC-AI 2000, Las Vegas, U.S.A.