Evolutionary Algorithms for Vehicle Routing

Jean-Yves Potvin
Département d'informatique et de recherche opérationnelle and Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport, Université de Montréal, C.P. 6128, succ. Centre-Ville, Montréal, Canada, potvin@iro.umontreal.ca

This paper is a survey of the literature on applications of evolutionary algorithms for vehicle routing problems. It reports about genetic algorithms, evolution strategies and particle swarm optimization, when applied to the classical capacitated vehicle routing problem and many of its variants. The performance of evolutionary algorithms is also compared with the best alternative problem-solving approaches on benchmark instances.

Key words: evolutionary algorithms; genetic algorithms; evolution strategies; particle swarm optimization; vehicle routing
History:

---

1. Introduction

This paper reviews applications of evolutionary algorithms (EAs) for solving vehicle routing problems. EAs are stochastic search methods that operate on a population of solutions by simulating, at a high level of abstraction, the evolution processes observed in nature. The survey excludes the work done on the Traveling Salesman Problem (TSP), a canonical combinatorial optimization problem that has been used as a testbed for many new methodologies, including EAs. We thus focus on extensions of the TSP, where a fleet of vehicles, starting from one or a number of depots, must visit a set of customers at minimum cost, subject to side constraints. A precise definition of the classical capacitated vehicle routing problem and its variants is given in the next section.

This review is divided along methodological lines rather than vehicle routing types. It just happens that the same researchers have often addressed different variants of the vehicle routing problem with the same basic problem-solving methodology, slightly adapted or extended to address the specific characteristics of a given variant. It is thus easier to follow the work done by these researchers.

In the following, Section 2 first provides a description of the main vehicle routing variants addressed by EAs. Sections 3 to 5 are then devoted to genetic algorithms, evolution strategies

and particle swarm optimization, respectively. Section 6 summarizes the work reported in the previous sections by collecting all references in a single table for each type of EA and vehicle routing variant. A computational comparison of EAs on the most popular variants is then reported in Section 7. A conclusion follows in Section 8.

Although this survey is the only one that provides an exhaustive account of applications of EAs to vehicle routing problems, the reader can find overviews, surveys and classification of metaheuristics for solving combinatorial optimization problems at large in [21, 29, 54, 192], and reviews of metaheuristics for vehicle routing problems in [26, 27, 50, 51, 52, 53, 55, 135].

2. The Vehicle Routing Problem

In the Vehicle Routing Problem (VRP), least-cost delivery or collection routes must be designed for a set of customers, using a fleet of vehicles located at a central depot, subject to side constraints. More formally, let $G = (V, A)$ be a graph with $V = \{1, ..., n\}$ the vertex set. In this graph, vertex 1 stands for the depot, while the remaining vertices are customers. Also, with every arc $(i, j) \in A$, $i \neq j$, is associated a non negative distance matrix $C = (c_{ij})$ (in some contexts, c_{ij} represents a travel cost or travel time). Note that when C is symmetrical, the set A can be conveniently replaced by a set E of undirected edges. We also assume that a fleet of m vehicles is available at the depot to visit the customers, where m is fixed or free. In the latter case, the number of vehicles in a solution can vary between 1 and $n - 1$. All vehicles are identical and have the same finite capacity Q.

Usually, the number of vehicles is free and the objective is to minimize the total distance traveled by the vehicles to visit all customers. Sometimes, a fixed cost is associated with each vehicle and the objective is then a weighted sum of fixed costs and travel costs. The problem then consists in designing a set of optimal routes such that:

- each customer is visited exactly once by exactly one vehicle;
- all vehicle routes start and end at the depot;
- some side constraints are satisfied.

This problem is NP-Hard, as it is a generalization of the TSP. Accordingly, it has given rise to a large number of heuristic problem-solving methodologies, including metaheuristics, to address instances of reasonable size. The most common types of vehicle routing problems reported in the literature are:

- *Capacitated VRP*: In the capacitated VRP, a non negative demand d_i is associated with each customer $i \in V \setminus \{1\}$ and the total demand on a route should not exceed the vehicle capacity Q. In some time- or distance-constrained variants, additional constraints state that the length of a route should not exceed a given limit. Note that VRP will stand for the capacitated VRP, with or without route length limits, in the following.

- *VRP with time windows*: In the VRPTW, a vehicle must arrive at customer i within the time interval $[e_i, l_i]$. In most cases, it is possible for the vehicle to wait at customer i and thus to arrive before the lower bound e_i. Some variants also allow a vehicle to arrive late, thus after l_i, although a penalty is incurred in the objective. Also, a time window at the depot defines the scheduling horizon (i.e., the vehicles depart from the depot, follow their route and return to the depot within that horizon).

- *VRP with time deadlines*: The VRPTD is a special case of the VRPTW, where the time windows are replaced by time deadlines. More precisely, there is no lower bound e_i at each customer i, only an upper bound l_i.

- *Time-dependent VRP*: The TVRP is aimed at modeling more realistic situations by considering travel times that depend both on the distance between two customers and the time of the day. For example, it takes longer to get from one customer to the next during rush hours. The TVRP takes these considerations into account.

- *Periodic VRP*: The PVRP is a variant of the VRP where the horizon extends over a number of periods. Routes are constructed for each period, and each customer is visited once or more over the horizon, depending on the customer requirements.

- *VRP with heterogeneous fleet*: The VRPHF is a variant of the VRP where the vehicles do not share the same characteristics (e.g., different capacities).

- *Multiple Depot VRP*: In the MDVRP, there are many depots and each vehicle can start or end its route from any of these depots.

- *VRP with backhauls*: In the VRPB, some customers require deliveries (linehauls) while others require pickups (backhauls). Each route is thus a mix of linehaul and backhaul customers, where the backhauls are typically visited after the linehauls. It has been

quickly recognized that substantial cost savings can be achieved by allowing empty vehicles that return from deliveries to pick up inbound products. In the grocery industry, for example, supermarkets and grocery suppliers would be the linehaul and backhaul customers, respectively.

- *VRP with simultaneous delivery and pick-up*: In the VRPSDP, the same customer can ask both for goods to be delivered from the depot and for other goods to be picked up and brought to the depot.

- *Pick-up and delivery problem*: Each customer i in a PDP has both a pick-up location i^+ and a delivery location i^-. That is, the demand collected at i^+ must be delivered at i^-. Both the pick-up and delivery points of customer i must be in the same route (pairing constraint) and i^+ must be visited before i^- (precedence constraint). Different real-world applications, usually with time windows (PDPTW), are reported in the literature, like courier and dial-a-ride services.

We examine different adaptations of EAs for these vehicle routing problems. In the sections that follow, genetic algorithms (GA) and evolution strategies (ES) are reviewed. Particle swarm optimization (PSO), which is often related to EAs, is also briefly considered.

3. Genetic Algorithms

Genetic algorithms (GAs) have emerged from the work of Holland at the University of Michigan [67]. Starting from some initial population, the search mechanism of a simple GA is divided into four phases: (1) evaluation of each solution in the population, (2) selection of parent solutions, (3) application of crossover and mutation operators to parent solutions to generate offspring solutions and (4) replacement of the old population by the new population of offspring solutions. This process is repeated for a number of iterations or until the system does not improve anymore. This search mechanism is applied on solutions encoded as bit strings. Through this coding scheme, generic operators have been designed that manipulate bit strings without any knowledge of the corresponding solution (apart from the solution value). A simple GA can be sketched as follows:

1. Create an initial population of P solutions.

2. Evaluate each solution.

3. Repeat for a fixed number of iterations:

   3.1 Repeat until P offspring solutions are created:

      3.1.1 Select two parent solutions in the population (with replacement) using a randomized selection procedure based on the solution values.

      3.1.2 Apply crossover to the two parent solutions (with a certain probability) to create two offspring solutions. If crossover is not applied, the offspring are identical to the parents.

      3.1.3 Apply mutation (with a certain probability) to each offspring.

      3.1.4 Insert the two offspring in the new population.

   3.2 Evaluate each offspring in the new population.

   3.3 Replace the old population by the new one.

4. Return the best solution found.

In Step 1, the initial population can be created randomly, but it is better to use construction heuristics to create at least a fraction of the population. Different techniques exist for selecting the parent solutions in Step 3.1.1. Typically, a randomized roulette-wheel selection assigns a selection probability to each solution which is proportional to its value. More recently, tournament selection procedures have been devised. Here a number of solutions are selected in the population and the best one becomes a parent. The procedure is repeated twice to get two parents. The crossover and mutation operators are applied in Steps 3.1.2 and 3.1.3. The classical one-point crossover creates two offspring by randomly selecting a cut point on the two parent strings and by exchanging the two substrings after the cut point. The mutation operator then processes each offspring position by position and switches the bit value at each position with a small probability. This secondary operator introduces small perturbations into the solutions and is often useful to maintain a diversity of solutions in the population.

When applied to combinatorial optimization problems in general, and to vehicle routing problems in particular, the simple GA described above exhibits a number of shortcomings.

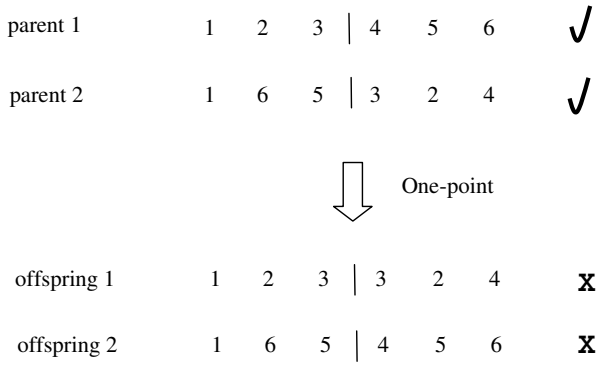

Figure 1: One-point crossover

These shortcomings have been recognized early in the case of the TSP, a canonical combinatorial optimization problem which can be viewed as a simple vehicle routing problem. In this particular case, a single vehicle of infinite capacity, starting from any given vertex, visits all other vertices and returns to its starting point, with the objective of minimizing the total distance. The main difficulties encountered when a classical GA is applied to the TSP can be summarized as follows:

- The binary string representation is quite cumbersome. An integer string representation, where each integer stands for a customer index, is much more natural. In the literature, this is referred to as the path representation, which is simply the sequence of customer indices in a tour. In this representation, the last customer in the sequence is implicitly connected to the first one to form a tour.

- The classical one-point crossover proved inadequate, particularly when applied on the path representation. As illustrated in Figure 1 on a tour with six vertices, this operator leads to invalid tours with missing and duplicated vertices. This is also true for generalized versions of this operator, like the 2-point and M-point crossover operators, where substrings between two consecutive cut points are exchanged.

- Special crossover and mutation operators, called order-based operators, must be designed to allow valid offspring tours to be generated. A well-known crossover operator for the TSP is the order crossover OX [115]. First, two cut points are randomly chosen.

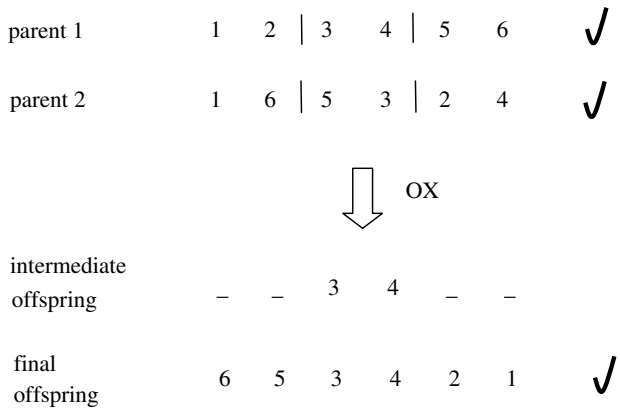

Figure 2: *OX* crossover

The substring between the two cut points on the first parent is copied to the offspring. Then, the remaining positions are filled by following the customer order on the second parent, starting at the position just after the second cross point. When the end of the sequence is reached, the procedure resumes at position 1. Figure 2 provides an example on a tour with six vertices. The substring made of vertices 3 and 4 in parent 1 is first copied to the offspring. Then, the customers are processed in the order 2, 4, 1, 6, 5, 3 on the second parent. Vertex 2 is inserted at position 5, vertex 4 is discarded because it is already present in the offspring, vertex 1 is inserted at position 6, vertex 6 at position 1, vertex 5 at position 2 and vertex 3 is discarded. This operator thus produces a valid offspring tour from two parent tours. Many order-based operators like this one are reported in the literature to handle sequencing problems, where the solutions differ only by the ordering of their elements [127].

- Heuristic information must be incorporated into the GA to obtain competitive results. For example, powerful heuristic-driven local search-based operators are now commonly used within GAs and often play the role of the mutation operator. This kind of GA is also known in the literature as a memetic algorithm [106, 107].

Vehicle routing problems are more complex than the TSP since they comprise both an assignment subproblem, where each customer is assigned to a particular vehicle, and a sequencing subproblem, where the customers assigned to the same vehicle are sequenced to form a route. Clearly, the difficulties observed in the case of the TSP are even worse here. In

particular, the representation, solution evaluation and problem-solving methodologies need to be adapted to solutions made of multiple routes, as explained below.

Representation

With regard to the representation issue, the main formalisms for vehicle routing problems (some of which inherited from the TSP) are:

- *No representation.* Since many GAs for combinatorial optimization have already drifted away from the evolutionary orthodoxy, some authors propose to forget about the representation and to apply the genetic operators directly on the solutions [128].

- *Random keys.* An encoding scheme based on random numerical keys is proposed in [9] for sequencing problems. That is, each element of a solution is tagged with a random key and a sequence is obtained by sorting these keys. By manipulating and modifying the random keys with crossover and mutation operators adapted to this representation, different sequences can be generated. For vehicle routing problems, a solution element is a customer index which is tagged with an integer that represents the assigned vehicle, plus a real number in the interval (0,1). By sorting the real numbers which are coupled with the same vehicle number, a route for this vehicle is obtained [9, 79, 184].

- *Path representation.* Here, a sequence of customer indices is used with separators to indicate where the routes end. When the number of vehicles is a variable, the number of separators also varies, thus leading to a solution string of variable length. To obtain a fixed length representation, some authors use a sufficiently large number of separators and interpret consecutive separators in the sequence as a single separator. Another approach is to add pointers at the end of the representation to indicate where each route ends. In decoder or route first-cluster second approaches (see subections 3.2 and 3.4), a single sequence without any separator is used, thus leading to the classical path representation for the TSP.

- *Assignment representation.* In this representation, each position in the string corresponds to a customer (i.e., position 1 stands for customer 1, position 2 stands for customer 2, ...) and the value at each position is a vehicle number, namely, the vehicle assigned to the corresponding customer. Since this representation is not an ordering,

classical crossover and mutation operators can be used to generate new vehicle assignments. An alternative representation is reported in [76], where the assignment is encoded as a binary matrix. In this matrix, the columns correspond to customers and the rows to vehicles. If the entry (i, j) is 1, it means that customer j is visited by vehicle i. Accordingly, all entries set to 1 in a row correspond to a group of customers visited by the same vehicle. These representations are appropriate to address the assignment subproblem, but not the sequencing subproblem.

- *Cluster representation.* In this representation, each position contains a cluster identifier, that is, a group of customers visited by the same vehicle. This representation is typically used at some upper level, where different partitions of the set of customers are explored before the sequencing issue is addressed.

- *Multi-part representation.* In this more complex representation, each part is a different string that corresponds to a vehicle route (this is called a Genetic Vehicle Representation (GVR) in [124]). A path representation is then used within each part to encode the corresponding route. Note that this is pretty close to approaches where the GA works directly on the solutions (see *No representation*, above).

Problem-solving methodology

Obviously, solutions with multiple routes also impact the problem-solving methodologies. For one thing, the genetic operators need to be adapted to the chosen representation. Also, the GA must account for the side constraints that lead to these multiple routes. This is usually done in the following ways:

- The design of genetic operators that ensure solution feasibility.

- The use of a decoder that constructs a feasible solution from an ordering of customers produced by the GA (see Section 3.2).

- The introduction of penalties in the evaluation of an infeasible solution.

- The use of repair operators to restore feasibility.

Solution evaluation

Multiple routes impact the way a solution is evaluated. A pervasive issue with regard to vehicle routing problems, particularly in the case of the VRPTW, concerns the number

of vehicle routes in a solution versus the total distance or travel time of the vehicles. Most problem-solving approaches for the VRPTW first minimize the number of vehicles and, for the same number of vehicles, minimize the total distance or travel time. This issue appears in the proposed methodologies, either via (1) the use of operators specifically designed for each objective, (2) the application of distinct search phases for each objective, (3) the co-evolution of two populations, one for each objective and (4) the development of true bi-objective approaches.

The reader will note that the considerations mentioned above emerge at different places in this survey. In the following, we first address GAs, which are by far the most widely used EAs for solving vehicle routing problems. The various methodological lines proposed in the literature are described, starting with extensions of methodologies previously developed for the TSP.

3.1. TSP extensions

Since a lot of work has been done on the TSP, it is not surprising that problem-solving approaches developed for the TSP have been adapted to vehicle routing problems. A good example is the edge assembly crossover EAX, a powerful operator for the TSP [111], which was later adapted to the VRP in [109]. In the case of the VRP, this operator can be described as follows:

1. Construct a graph by combining the edges of the two parent solutions.

2. Partition the set of edges in this graph with cycles created by alternately selecting an edge from the first and second parents.

3. Select a subset of cycles.

4. Generate an intermediate solution as follows. Take one parent and remove all edges that are in the subset of cycles. Then, the edges in the subset of cycles from the other parent are added. At this point, the intermediate solution is a collection of routes connected to the depot plus subtours that are not connected to the depot.

5. Create a complete solution. A greedy heuristic is applied where, at each iteration, a subtour is merged at least cost to a route or to another subtour. The procedure is repeated until a set of routes is obtained that cover all customers.

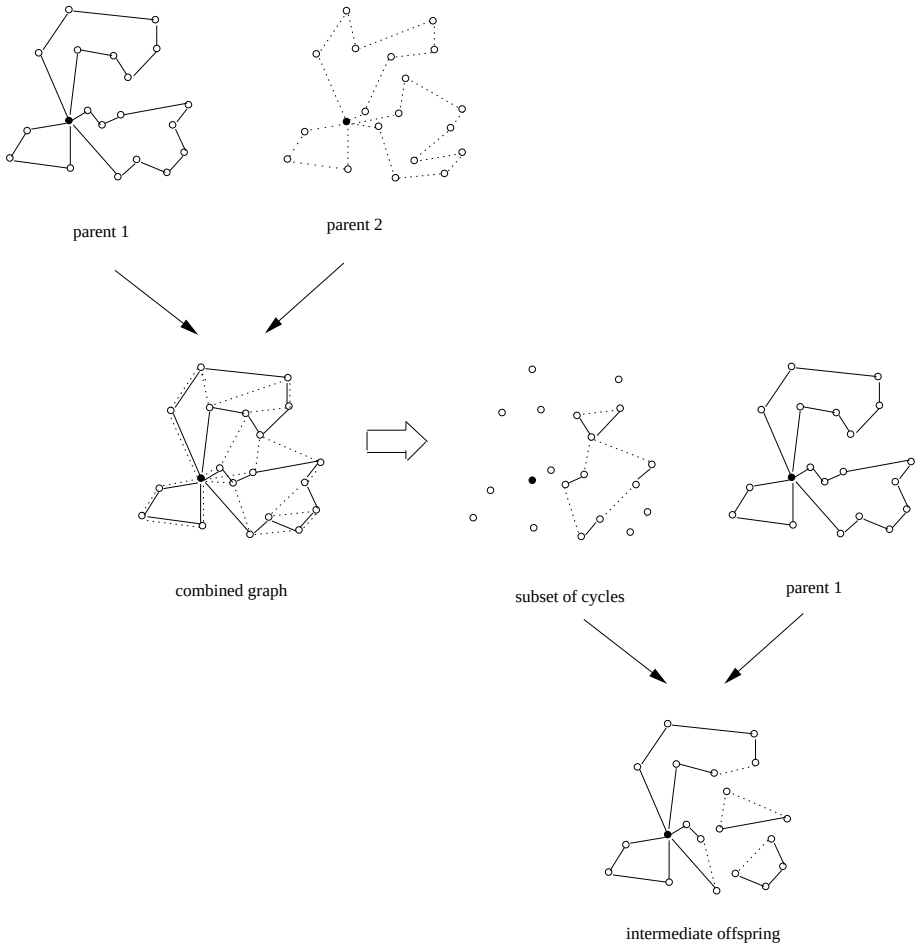

Figure 3: *EAX* crossover

6. Eliminate capacity violations. Using a penalty function for route overcapacity, restricted 2-opt exchanges [96] and customer exchanges are applied until a feasible solution is obtained. These modifications are said to be restricted because an infeasible route is necessarily involved.

The *EAX* crossover is illustrated in Figure 3. From two parents, a combined graph is obtained. Then, based on the particular set of cycles shown in this example, an offspring is produced. The latter is obtained from parent 1, by removing the edges of parent 1 and by adding the edges of parent 2 that are in the set of cycles. Clearly, the offspring is not a valid solution since it contains two subtours that are not connected to the depot. These subtours are thus merged with the routes to obtain a valid solution, as mentioned above.

This GA with *EAX* reached the best solution on a subset of instances taken from Christofides, Mingozzi and Toth's VRP set [33] and found ten new best solutions on Golden

et al.'s set [61]. This work has also been extended to the VRPTW in [110]. In this case, an additional term for time window violations is included in the penalty function when feasibility is restored with exchanges.

An adaptation of the natural crossover, previously developed for Euclidean TSPs [81], is reported in [82] for the VRPTW. This operator, works on a two-dimensional graphical representation of the problem. It partitions the set of customers into two different classes by drawing one or more curves or geometric figures, like rectangles and ellipses, over the graphical representation. That is, customers are on one side or the other of the curve or customers are inside or outside of the geometric figure. Then, arcs from the first (respectively, second) parent solution with both endpoints in the first (respectively, second) class are transferred to the offspring solution. At this stage, disconnected segments from both parents are obtained. A repair algorithm is then applied to connect these segments to form feasible routes. Basically, the routes are constructed one by one starting with a segment connected to the depot. This partial route is extended by considering all available segments that produce a feasible route when the last customer on the partial route is connected to the first customer of the segment. Then, the least cost connection is chosen, where the cost takes into account spatial and temporal issues. A local optimization is also applied on the resulting solution, based on intra- and inter-route customer moves and on 2-opt [96]. An example of the natural crossover is shown in Figure 4. In this example, a rectangle is drawn to divide the vertices into two classes, namely those that are inside or outside of the rectangle. The customers inside the rectangle are linked according to the edge pattern of parent 1, while the others are linked according to the edge pattern of parent 2. The intermediate offspring contains four segments that need to be reconnected to produce a valid solution.

In [86, 155], the GENITOR algorithm [191] is adapted to the VRP by modifying its edge recombination crossover ER. This operator, originally designed for the TSP, progressively extends a tour by adding edges found in one of the two parents. These edges are stored in a data structure called an edge table, where all parental edges incident to a given customer node are grouped together. In the case of the VRP, ER is modified in the following ways: (1) the depot is always used as the starting point of a route, (2) the edge table includes the depot for which the number of incident edges is two times the number of vehicles, (3) only edges leading to the closest nodes are considered and (4) capacity constraints are taken into account to stop the construction of the current route and starts a new route. In this work, the mutation operator is a 2-opt [96] which is applied at every G iterations, where G is a

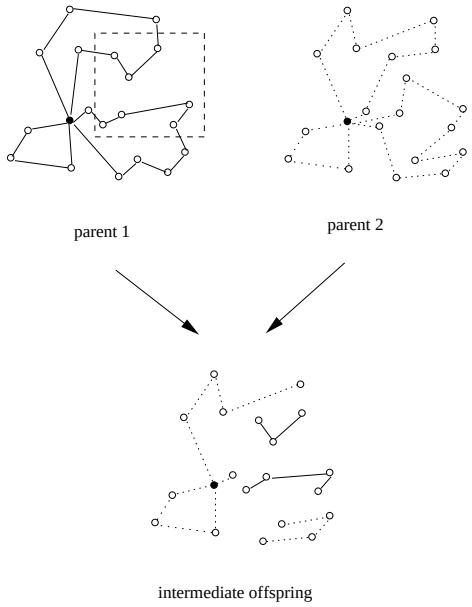

Figure 4: Natural crossover

parameter. In a follow-up paper [42], an open loop power distribution problem is modeled as a VRP and solved using the same approach, except that 2-opt is replaced by 3-opt. In [41, 43], the same modified ER operator is used for a multi-depot vehicle routing problem where each route can start and end at a different depot and where the number of vehicles at each depot is known. After a clustering phase that assigns each customer to the closest depot, and a routing phase at each depot that creates open half routes (by dividing the capacity in two) with a greedy heuristic, the GA with modified ER is applied at each depot to optimize the half routes. When this is done, another GA is applied to find the best way to link pairs of half routes together to form complete routes. Basically, the crossover operator in this GA combines half routes taken from two different solutions. A similar approach to this problem, but where a decoder is used to form the half routes, is discussed in Section 3.2.

The work in [32, 66] addresses the VRPTW through an extension of a matrix-based crossover operator for the TSP [45]. Basically, an entry (i, j) in the matrix is set to a large value if customers i and j are in the same route in both parents and share the same precedence relationship. A small value is used when i and j are not in the same route. Intermediate values are used when customers i and j are in the same route in both parents but do not share the same precedence relationship or when i and j are in the same route in one parent only. Offspring solutions are then generated by applying Solomon's I1 insertion

heuristic [156], using the values stored in the matrix. Solomon's heuristic first selects a seed customer and creates an individual route for this customer. Then, at each iteration, a customer is inserted at least cost over all feasible insertion places in the current route, where the insertion cost takes into account spatial and temporal issues. When no customer can be feasibly inserted in the current route, a new route is created. This is repeated until all customers are visited. In the case of the matrix-based approach, the values in the matrix are used to modify the insertion costs. For example, if customer i precedes customer j in both parent solutions, so that entry (i, j) in the matrix has a large value, the insertion of j just after i is greatly favored by dividing its insertion cost by this large value. The GA is used in conjunction with a tabu search heuristic [57] that further improves a fraction of the solutions in the current population. The authors propose to run the GA without tabu search in the first iterations. Then, the intensity of the tabu search is progressively increased, by applying it to a larger fraction of the population and by running it longer on each solution. With this approach, a graceful convergence of the population is observed. A completely different approach for designing a crossover operator for the VRPTW is also proposed in this work, based on the concept of a dominant and a recessive parent, where the dominant parent is the one with better evaluation. The operator first makes the offspring a duplicate of the dominant parent. Then, the path representation of the recessive parent is swept position by position to get the current customer i and its immediate successor j. These customers are located in the offspring and an attempt is made to put customer j immediately after customer i in the offspring, if this is feasible.

The VRPB is solved in [170] with a number of order-based crossover operators for the TSP that are adapted to take into account backhaul customers. Unfortunately, there is little information about how this adaptation is realized. A problem that exhibits features of both the VRPB and VRPSDP is solved in [46] with an algorithm called CLOVES. In this application, each customer is either a pure delivery (D), a pure pickup (P) or a delivery and pickup (DP). Furthermore, the customers must be visited in the following order: pure delivery, delivery and pickup, pure pickup. In the CLOVES algorithm, only the last stage is based on a GA. The first stages construct a solution for this problem by (1) clustering customers for each of the D, P and DP categories, (2) ordering the customers within each cluster to create segments and (3) generating complete routes for the vehicles by appropriately assigning D, P and DP segments to each vehicle. The GA is then called on a population made of the solution obtained plus additional ones produced by applying Or-opt exchanges

on this solution [118]. Two variants of the OX crossover [115] are proposed. The first variant is the classical operator, where a segment between two crossover points is chosen on one parent and transferred to the offspring. The empty locations in the offspring are then filled by sweeping the second parent from the first to the last position (to preserve, as much as possible, the required D-DP-P ordering). The second variant does the inverse: all customers outside of the crossover points are copied in the offspring and the empty locations between the two crossover points are filled by sweeping the second parent. Due to the unavailability of benchmarks for the problem considered, the algorithm was tested on VRP and VRPB benchmark instances. The average gap with the best-known solutions in these two cases was typically under 1%.

A problem motivated from a real-world application, where a single vehicle has to perform daily tours to visit on-shore oil wells, is addressed in [146]. The tour starts and ends at the oil treatment station where separation of oil from water occurs. As the vehicle cannot visit all wells in a single day, the maximum amount of oil should be collected within a given time horizon. Although this problem is more closely related to the prize collecting TSP [40], it is considered here due to the presence of a physical depot where the vehicle must start and end its route. The solution representation is made of a variable number of wells, as the authors have chosen not to include unvisited wells in it. The crossover operator thus works on a variable-length representation and is also extended to apply to a varying number of parents. It creates an offspring, starting with a randomly chosen node. Then, the next node j is probabilistically chosen based on a measure that takes into account how often node j is found just after node i in the parent solutions. Two different types of local search heuristics are then applied in sequence with a given probability associated with each one. The first one tries to insert a well not currently visited in the solution. The other one exchanges a visited well with an unvisited one. At regular intervals, a so-called data mining module is applied to discover relevant subsequences in the best solutions found thus far. These subsequences are then used to construct new solutions. A crossover mechanism similar to the one mentioned above is applied, except that when the next node is selected, frequent subsequences that involve that node as the starting node are considered for inclusion in the solution.

A similar, but more complex, multi-vehicle problem is reported in [148, 149, 150]. This problem is in fact a PDPTW, motivated from a less-than-truckload application, but where it is still not possible for the fleet of vehicles to visit all customer requests. Accordingly, some requests are subcontracted to a common carrier. The problem is thus to select the customer

requests to be visited, to assign these requests to the available vehicles and to sequence them to form routes. This should be done at least cost, which is the travel cost to visit the selected customers plus the subcontracting cost for the remaining customers. A path representation is used (with unserved requests at the end), along with pointers to indicate the end of each route. The GA starts with an initial population of solutions generated with the construction heuristic reported in [148]. A uniform crossover operator is then applied to create each offspring route. That is, a bit mask is generated to decide if each customer request in the offspring route is taken from parent 1 or parent 2. When the entry in the bit mask is 0, the request is taken from parent 1; when the entry is 1, the request is taken from parent 2. Once the current route is completed, the procedure is repeated with the next route. A mutation operator is also used, where a customer request is reinserted at another location. The request can be taken from the unvisited ones, in which case the operator injects a new request in the solution. Although the crossover and mutation operators are designed to guarantee that the pairing and precedence constraints are satisfied, this is not necessarily the case for the capacity and time window constraints. Consequently, a 2-opt [96] is applied to reduce the travel costs and constraint violations. Afterward, if the solution is still infeasible, requests are removed and added to the unvisited ones to make the solution feasible.

In the work of Alba and Dorronsoro [1], the capacitated VRP with a maximum time- or distance-constraint is solved with a cellular GA. As opposed to classical GAs, where a solution is allowed to recombine with any other solution, the population of a cellular GA is structured according to some underlying topology (e.g., grid, mesh) and a solution can only interact through crossover with its immediate neighbors, according to the topology. As these neighborhoods overlap, good solution features can slowly diffuse in the whole population. Smaller populations can thus be used to solve problems with cellular GAs, as they are less prone to premature convergence. Some early applications of GAs to the TSP were based on this approach (see, for example, [108]). In [1], a 2D toroidal grid defines the connection topology. The neighborhood of a given solution is thus made of the solution itself, plus the so-called North (up), South (down), West (left) and East (right) solutions. The ER crossover operator [191] and reinsertion, exchange and inversion mutation operators (where a substring of customers is inverted) are used within the GA. Local search heuristics are also applied to further improve the solutions. In [2], this cellular GA is used to solve the VRP instances of van Breedam [186], Golden et al. [61] and Taillard [159]. Nine best solutions on these

instances are reported, although eight of them are obtained on van Breedam's instances, which are not as widely known as the two other sets.

Although the adaptation of EAX to the VRP has produced some good results, this is not the case for most methodologies based on TSP extensions, because they (obviously) have been designed for the TSP in the first place and later "twisted" to account for side constraints. Thus, this line of attack is not natural nor easy, in particular for problems with time constraints.

3.2. Decoders

In a decoder, an ordering of customers (here, generated by a GA) is given as input to a route construction heuristic that decodes the ordering into a feasible solution to the problem. Basically, the customers are considered one by one, based on the ordering indicated by the GA, and feasibly inserted into the current solution. The GA is thus free to explore the search space of customer orderings and does not need to account for side constraints, which are handled by the construction heuristic.

Blanton and Wainwright [20] were the first to propose a decoder approach for the VRPTW. In their GA, the Merge crossover operators $MX1$ and $MX2$ exploit a global precedence relationship among customers, based on their time windows, which is independent of any particular solution. Namely, the crossover operators rely on the assumption that it is desirable for customer i to appear before customer j in the ordering if the time window of i is earlier than the time window of j. In the $MX1$ crossover, the two parent orderings are processed position by position and, at each position, the customer with earlier time window (based on the time window's lower bound) is transferred to the offspring. The resulting ordering is thus biased toward the one obtained by sorting the customers in non-decreasing order of their time window. The $MX2$ operator works similarly and differs only in the way the two parent orderings are processed.

Each ordering produced by the GA is then decoded into a solution of the VRPTW and evaluated by adapting an approach reported in [75] for a partitioning problem. Assuming a fixed number of vehicles m, the first m customers in the ordering are used to initialize the vehicle routes, where each route visits a single customer. The remaining customers are then inserted one by one into any of these routes (so that all routes grow in parallel), based on a least-cost insertion measure. A customer who cannot be feasibly inserted in the solution

remains unvisited. Such infeasible solutions are kept in the population but are highly penalized. Tests on problems with 15, 30, 75 and 99 customers were performed by integrating the two operators into the GENITOR package [191]. These operators were shown to outperform general purpose order-based crossover operators like ER [191] and OX [115]. An improvement is proposed in [97] for instances where customers are geographically clustered (as instance classes C1 and C2 in Solomon's data set [156]). Rather then using the first m customers in the sequence to initialize m routes, the authors take care to select customers from different clusters to initialize the routes.

In [132], the authors argue that $MX1$ and $MX2$ are too strongly biased by the customers' time windows. Accordingly, they propose a crossover operator that stands between general purpose order-based crossover operators and the $MX1$ and $MX2$ operators. The $1X$ crossover is similar to the classical one-point crossover and thus generates invalid orderings. These orderings are then repaired by eliminating duplicate customers and by inserting all customers that are not yet part of the ordering. When inserting customers into the ordering, their insertion order is determined by the time occurrence of their time window (i.e., the customer with earliest time window is inserted at the first available position, etc.). As opposed to [20], the construction heuristic for decoding the ordering into a solution is Solomon's I1 insertion heuristic [156], where the routes are built one by one. The new crossover operator was shown to outperform $MX1$ and $MX2$ on Solomon's VRPTW instances. The same GA was then applied to the VRPB with time windows in [131], by slightly modifying Solomon's insertion heuristic to account for the precedence relationship among linehaul and backhaul customers. Similar decoder approaches for the VRPTW are also reported in [105, 164, 193]. In [164, 193], Solomon's I1 sequential insertion heuristic is used, while a time-oriented parallel savings heuristic, also proposed by Solomon in [156], is used to construct routes in parallel in [105].

A messy GA [58] for the VRPTW is described in [163]. Messy GAs can be seen as a relaxation of classical GAs, because they are more flexible with regard to the solution representation. In particular, each element is a couple (position, value) that can appear at any particular location along the string. Messy GAs also allow for variable-length strings that may be under or overspecified with regard to the problem under study. Overspecification occurs when two elements share the same position but have different values. When the string is swept to decode it into a solution, overspecification is handled by selecting the first element encountered and by ignoring the following ones. Underspecification occurs when a particular

position is absent. In that case, the missing value is filled using the current template, which is the best solution from the previous iteration. As in classical GAs, there is a parent selection phase followed by the application of a cut-and-splice operator, which is similar to the one-point crossover, but adapted to variable-length strings. In the case of the VRPTW, an element is a couple (customer index, vehicle). Overspecification occurs when the same customer is served by two or more vehicles. Underspecification occurs when a customer is not served by any vehicle. Decoding the string into a solution of the problem is obtained by considering each couple in turn along the string and by applying Solomon's I1 heuristic [156] to insert the corresponding customer in the vehicle's route. As the authors minimize total distance only, they often obtain better solutions on Solomon's benchmark instances then other approaches, but they use more vehicles. It is worth noting that the GA developed by the authors only approximately follows the original design of messy genetic algorithms, and the theoretical justifications at the core of this design are more or less ignored.

A VRP with time deadlines is addressed in [99, 116]. One interesting feature of the proposed GA is a route crossover RC based on a bit mask where the number of bits corresponds to the number of routes in the first parent solution. If the bit value is 1, the corresponding route is transferred to the offspring solution. If the bit value is 0, the customers in the corresponding route are added to a temporary list. At the end, the customers in the temporary list are sorted, based on their relative order in the second parent and inserted, one by one, into the routes of the current offspring solution. Using the final ordering obtained, including the addition of infeasible customers at the end, a decoder then produces the final solution with a sequential construction heuristic.

A multi-depot VRP, where each route can originate and terminate at a different depot and where the number of vehicles at each depot is known, is addressed in [153, 154]. In this particular application, the number of vehicles that start their route from a given depot and end their route at another depot is specified for each pair of depots. The problem-solving methodology first assigns a number of closest customers to each depot. This number corresponds to the number of vehicles that goes in and out of the corresponding depot. Each assigned customer is used to initialize a route that goes in (first customer of the route) or out (last customer of the route) of the depot. Afterward, a GA is applied to handle the remaining customers and enlarge the routes until half of the vehicle capacity is reached. A decoder is used for this purpose, that is, an ordering of unassigned customers is provided as input to a construction heuristic that links each customer in turn to the closest end node of a

half route. When all customers are routed, the route halves are linked together in a heuristic way by sequentially and randomly selecting a half route and connecting it in the cheapest possible way to another half route to form a complete route (while taking care to satisfy the specifications about the number of vehicles that travels from one depot to another).

Finally, a GA for solving a dynamic vehicle routing problem is described in [65]. Here, new customer requests are received over the day and must be integrated in real-time into the current vehicle routes. In this work, the horizon is divided into time slices. A new vehicle routing problem is then defined and solved by the GA at the end of each time slice by incorporating any new request that has been received during that time. A variable-length string representation is proposed that encodes the new customer requests that have not been assigned yet, plus pointers to existing routes. A decoder then inserts these new requests one by one into existing routes, if this is feasible, or creates new routes for them, based on their ordering in the string.

Decoder approaches are attractive for real-life vehicle routing problems with many specific side constraints, due to the simplicity and generic nature of the insertion heuristics that are used to construct a solution. On well-studied problems like the VRP and VRPTW, though, these general-purpose insertion heuristics did not prove to be powerful enough to lead to results that are competitive with other state-of-the-art methods.

3.3. Cluster First-Route Second

The well known cluster first-route second problem-solving methodology is based on the natural partition of a vehicle routing problem into two subproblems, namely (1) an assignment subproblem where groups or clusters of customers that are visited by the same vehicle are identified and (2) a sequencing subproblem where the customers within each cluster are sequenced to form routes.

Sam R. Thangiah was one of the first researchers to apply GAs to vehicle routing problems with this framework. Basically, clusters of customers are first identified by a GA, while the routing is done afterward with standard operations research techniques. The GIDEON system [175, 177, 179] was first developed in the context of the VRPTW. Its clustering and routing phases work as follows. In the clustering phase, a genetic sectoring algorithm called GENSECT partitions the set of customers into m clusters, one for each vehicle, by identifying $m-1$ rays originating from the central depot. All customers lying between two consecutive rays are then grouped together to form a cluster. As each ray is defined by its polar angle,

$m-1$ polar angles are encoded on a bit string (to be more precise, each angle is encoded as an offset from a fixed increment angle). To evaluate the clusters produced by GENSECT, a route is formed within each cluster. This is done through a least-cost insertion heuristic, where the cost accounts for the total travel distance, as well as penalties for time window and capacity violations. At the end, a local search heuristic is applied to the best solution produced by GENSECT for further improvement. The neighborhood structure is based on the λ-interchanges of Osman [119] with $\lambda = 2$. In this case, one or two customers are moved to another route or exchanged with one or two customers in another route. This final phase is particularly important to reduce or eliminate violations of the capacity or time window constraints. GIDEON was used to solve a school bus routing problem in [178]. It was also applied to VRPTDs [180, 183], where the time window at each customer is replaced by a time deadline.

In [174], an alternative approach is proposed to cluster customers. The GA, called GENCLUST, adapts geometric shapes (more specifically, circles) to cluster customers. Each bit string encodes the origin and radius of a set of circles. The number of circles corresponds to the number of vehicles, and is estimated by first solving the VRPTW with Solomon's I1 insertion heuristic [156]. All customers found in a given circle are grouped into the same cluster. Customers that are in-between two or more circles are associated with the closest circle, while customers that are found inside two or more circles (assuming that circles can overlap) are associated with the circle that appears first along the bit string. Thus, when a small circle is included within another larger circle, and customers are found only within the small one, the circle that appears first takes all customers. The other circle becomes empty and is removed, thus saving one vehicle. As in GENSECT, the circles or clusters produced by the GA are evaluated by sequencing the customers within each cluster with a least-cost insertion heuristic to form routes.

GENCLUST was also adapted to solve the multi-depot vehicle routing problem, where each vehicle starts and ends its route at the same depot [145, 182]. Basically, when a set of circles is produced by the GA, a solution to the multi-depot problem is obtained by associating each circle or cluster of customers with the nearest depot. The interested reader will find a review of Thangiah's work with GAs for these different types of vehicle routing problems in [181].

In [126], the cluster first-route second approach is implemented using a different GA in each phase. For the clustering phase, an assignment representation is exploited by the GA

to assign customers to vehicles, that is, to form clusters. Another GA then form routes within each cluster by applying the PMX order-based crossover [60] on a classical path representation. This problem-solving approach is applied on a PDP, more precisely a dial-a-ride problem with no time windows. In this particular application, a repair operator is needed to swap the pick-up and delivery associated with a given customer, when the precedence constraint is not satisfied.

A dial-a-ride problem with time windows is addressed in [76]. Here, the assignment representation is based on a binary matrix, where entry (i, j) is 1 if customer j is visited by vehicle i, and 0 otherwise. Thus, each row contains the group of customers assigned to a given vehicle. A crossover operator then explores different ways to perform this assignment. First, one row is selected in the matrix of each parent. A binary mask is then used to create a new row for the offspring. Each row element is inherited from parent 1 or parent 2, depending if the bit value in the mask is 0 or 1, respectively. The other rows are inherited from parent 1. A repair operator is then applied to make sure that each customer is assigned to exactly one vehicle (i.e., each column has exactly one entry equal to 1). Routes are then created from the obtained assignment using a nearest neighbor heuristic that accounts for both spatial and temporal proximity.

In [121], each element in the representation of a PDPTW corresponds to a group of requests that are assigned to the same vehicle, rather than a single request or a single service point (pickup or delivery). A variant of the 2-point crossover operator is then applied to combine clusters from two parents on a single offspring. The substring of clusters between the two cut points on the second parent is inserted just after the first cut point on the first parent to create an offspring. Then, a repair operator removes duplicate customers (which can lead to the removal of whole clusters). Finally, unassigned requests are inserted, allocating a new vehicle if necessary. The mutation operator randomly removes a cluster and reinserts its customers into other clusters, also allocating a new vehicle if necessary. While clusters are worked on, the corresponding routes are created and updated. For example, when a customer request is reinserted in another cluster, the insertion is also performed in the corresponding route to ensure feasibility. An adaptation of this GA for a dynamic version of the PDPTW, where new requests are incorporated in real-time into the current routes, is reported in [122]. Basically, a series of static subproblems over known requests are solved under a rolling time horizon. A static subproblem is defined each time a new request occurs. At this point, each solution in the current population is updated (synchronized) to reflect

the current state of the system. The population obtained becomes the initial population for the next execution of the GA.

In [8], the VRP is addressed with a GA using an assignment representation. Although this representation is not meant to specify the sequence of customers visited by each vehicle, the routes are implicitly inferred. That is, the customers are indexed in such a way that it makes sense for two customers with consecutive indices to be visited one after the other by the same vehicle. These indices are determined by sorting the customers according to a nearest neighbor TSP solution, when the customers are clustered, or based on their polar angle, when the customers are randomly located. The route associated with a given vehicle is then obtained by taking all customers with the same vehicle number, and by sequencing these customers based on their relative order in the representation. Local optimization heuristics, like 2-opt and 3-opt are then applied on each route. Since the representation encodes groups of customers that are assigned to the same vehicle as well as their sequence, it is not a true cluster-first, route-second approach. However, these routes are only the starting point for the local optimization heuristics. They do not correspond to the real routes, which are those obtained after the local optimization.

In the case of the work in [37, 188], the cluster first-route second paradigm is extended to a periodic VRP. Here, *cluster first* corresponds to the selection of customers that are visited during the same period over a given multi-period horizon, while *route second* corresponds to the construction of routes for each period. A binary representation is used to encode the periods where each customer is visited. For example, if an horizon of three periods is considered, there are seven possibilities: 001, 010, 100, 110, 011, 101, 111. That is, if 101 is associated with a given customer, this customer is visited in periods 1 and 3. Based on a particular selection of visits for each customer, a VRP is then solved in each period using a cost saving heuristic. The proposed crossover operator creates two offspring by exchanging the selection of visits of a given customer on the two parents (if feasible), while the mutation operator randomly changes the selection of visits of a number of customers (if feasible). A parallel implementation of this GA is reported, where subpopulations evolve in parallel and periodically exchange solutions.

Although dividing the problem into an assignment and a routing subproblem is a natural approach for vehicle routing problems, difficulties occur when both spatial and temporal issues are present. In the methodologies reported in this section, spatial considerations are the primary concern in the clustering phase, while feasibility is addressed in the following route

construction phase (either through repair operators or penalties for constraint violations).

3.4. Route First-Cluster Second

Route First-Cluster Second is an alternative paradigm for tackling vehicle routing problems. Basically, the path representation encodes a (likely infeasible) unique giant tour that covers all customers. Using such a representation means, in particular, that classical order-based crossover and mutation operators previously developed for the TSP can be used, given that a single sequence of customers is worked on. The giant tour then needs to be partitioned into individual feasible routes. By working on a single sequence of customers and by coupling this representation with a procedure that transforms this sequence into a solution of the problem, an approach very similar to the decoders of subection 3.2 is obtained. However, the final routes are usually more closely tied to the original sequence through the use of decoders that can only break the sequence into subsequences. This approach can thus be viewed as a restricted type of decoder.

A VRP with route time constraints is addressed in this way in [147]. A true solution to the problem is obtained by sweeping the customers along the giant tour. A route ends when either the capacity or the maximum route time constraint would be exceeded by including the next customer. This customer is then used to start a new route. In [137], a polynomial-time algorithm is developed to partition the giant tour into individual routes in an optimal way. This algorithm, called SPLIT, works on an auxiliary directed acyclic graph with vertices {1,2,...,n}, where vertex 1 is the depot and the other vertices are customers. An arc (i, j) is added to the graph when a route from vertex $i + 1$ to vertex j, based on the ordering in the giant tour, is feasible. The length of the arc is the total distance of the corresponding route. A solution is then obtained by solving a shortest-path problem from node 1 to node n. This can be done in polynomial time due to the acyclic nature of the graph. An example taken from [137] is illustrated in Figure 5, assuming a vehicle capacity of 10. In this figure, the demand at each vertex is shown within brackets and each link is labeled with its length. The giant tour in Figure 5(a) starts at the depot 1, visits the vertices in the order 2, 3, 4, 5, 6, and returns to the depot. The corresponding shortest path problem is illustrated in Figure 5(b). In this graph, the arc from vertex 4 to vertex 6, for example, corresponds to the route 1, 5, 6, 1 with a total distance of 90. The optimum, shown with bold arcs, is to split the tour into three feasible routes: the first route visits vertices 2 and 3, the second route visits vertex 4 only and the third route visits nodes 5 and 6, for a total distance of

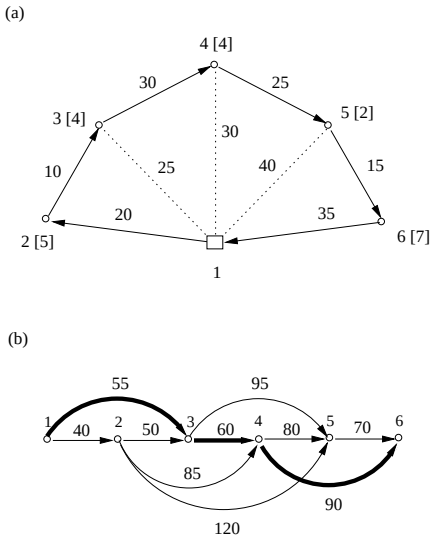

Figure 5: Exact split procedure

55+60+90=205. It is worth noting that a similar optimal procedure, that accounts for time windows, is reported in [194].

Before the discovery of this exact, polynomial-time tour splitting procedure, different heuristic approaches were proposed in the literature to address this problem. In [161], a local search heuristic explores different configurations of cut points along the giant tour. In [98], a cooperative co-evolutionary model, where two subpopulations evolve concurrently, is proposed. In the first subpopulation, the integer strings contain a single sequence of customers, while the strings in the second subpopulation contain a number of route sizes (in terms of number of customers). The giant tour encoded in the first string is then split according to the route sizes encoded in the second string. The two subpopulations are thus required to get a complete solution.

3.5. Ruin and Recreate and Large Neighborhoods

In this section, the genetic operators are inspired from the ruin and recreate paradigm [151], where a fraction of a solution is destroyed and then reconstructed in some alternative way. When integrated within a local search heuristic, this approach leads to large neighborhoods. For example, we can choose to remove q customers from a solution to ruin it. But each selection of q customers is likely to lead to a different neighboring solution after reconstruction. Clearly, the size of the neighborhood grows quickly with q, thus producing a large neighborhood search.

A good example of this approach is found in the work of Berger and his colleagues for the VRPTW. As the first objective of the VRPTW is usually to minimize the number of vehicles and, for the same number of vehicles, to minimize the total distance, researchers in the field have soon recognized the need to pay a special attention to each objective. The line of research of Berger and his colleagues makes no exception, as explained in the following.

In [17], an insertion-based crossover IB_X is proposed. First, route 1 is probabilistically selected from parent 1, with a bias toward routes with large waiting times. A number of customers are then removed from this route (ruin), based on criteria indicating that a more suitable relocation of these customers into alternate routes is likely, such as a large distance to the immediate successor or a large waiting time. A subset of routes close to route 1 are then selected in parent 2, and the best candidate customers in these routes, plus any unvisited customers, are considered for insertion in route 1. This procedure is repeated for a number of routes in parent 1. At the end, the offspring solution is completed with the remaining routes of parent 1. If there are still unvisited customers, they are handled by constructing additional routes with a nearest neighbor heuristic. A second child is obtained by interchanging the role of the two parents. Three mutation operators are also used to either reduce the number of routes or to reorder customers within routes. It is worth noting that solution feasibility is always enforced. A variant of the VRPTW, where a maximum route time constraint is introduced, is also addressed with the same approach in [18].

In a follow-up work [16], the authors relax the time constraints and evolve two populations with different objectives. Population 1, which contains at least one feasible solution, minimizes the total distance, while population 2 minimizes time constraint violations, both subject to a fixed number of routes (i.e., m and $m-1$ routes, respectively). When a feasible solution with $m-1$ routes emerges in population 2, population 1 is replaced by population 2, the number of routes m is reduced by one, and the mutation operator RSR_M (see below) is applied to population 2 to obtain a new population with solutions with one fewer route. This is repeated until no feasible solution emerges in population 2. The crossover operators are IB_X [17] and a variant called insert-within-route neighborhood crossover IRN_X. In this case, customers are removed from a subset of routes in parent 1. Then, the customers in the routes of parent 2 that are close enough to the subset of routes in parent 1, become candidate for insertion in this subset. A suite of five mutation operators is also proposed, among which the large neighborhood search-based mutation $LNSB\_M$ follows the guidelines of Shaw in [152]. Here, related customers (due to proximity or a common route assignment)

are removed and reinserted with a variant of Solomon's I1 heuristic [156], within a search tree framework. The insertion order of the customers is determined through the summation of two ranks: one is based on the insertion cost and the other on the number of feasible insertion places. The smaller the insertion cost and the number of insertion places, the better the customer's overall ranking. Note that the second term is new and is aimed at alleviating the myopic behavior of the original approach. The edge exchange mutation EE_M is inspired from the λ-interchanges [119] and examines the insertion or exchange of a customer with the two closest routes (as evaluated by the customer's distance to the route centroid). The repair solution mutation RS_M is similar to EE_M, but focuses on solution feasibility by exploring exchanges involving infeasible customer visits. The reinsert shortest route mutation RSR_M is used to eliminate the route with the smallest number of customers from a solution. It first tries to insert each customer at a feasible place in the two closest routes by maximizing a so-called regret cost function. Basically, the procedure computes the best insertion place for the customer in the two routes and considers the difference between the two values. This measure indicates how much is lost by not inserting the customer at its best insertion place. This a special case of the generalized regret measure proposed in [133], where the regret is computed over all routes instead of the two closest routes only. Finally, the reorder-customers mutation RC_M is applied when a new best feasible solution is found. It reorders the customers within each route by repeatedly applying Solomon's I1 heuristic [156] using different sets of randomly generated parameter values.

A parallel master-slave implementation of this GA is reported in [15]. In this implementation, the master supervises and controls the execution of the GA, while the slaves concurrently apply the genetic operators on different solutions in the two populations. In [24], a post-optimization phase is added to the GA. In this final phase, a population of routes (rather than complete solutions) is evolved, starting with the routes in the best solution produced by the original GA. The routes in the population are randomly ordered and pairs of routes are recombined based on this ordering. After a number of iterations, the best possible solution is constructed with the routes found in the final population. Four crossover operators are applied, depending on a priori probabilities. The CE_X operator applies CROSS exchanges [160] where two segments of consecutive customers are selected (one from each route) and exchanged. The insertion crossover I_X inserts one customer at a time in other routes. The rebuilding crossover R_X frees all customers in two parent routes and reinserts them using a least-cost insertion heuristic. The random rebuilding crossover RR_X is a

variant of R_X where the customers are considered for insertion in random order. The two mutation operators considered are first-improvement local search heuristics, where a new route is accepted as soon as it provides an improvement. The permute mutation operator P_M is based on Or-opt exchanges [118], where a string of one, two or three consecutive customers is removed from a route and inserted elsewhere. In the reorder mutation operator R_M, customers are removed from a route. Then, those with a tight time window are considered first during the reinsertion.

This parallel GA was later adapted to the VRP [13, 14]. In this case, the two populations are used for diversity purposes only, as they both optimize the total distance. Since the VRP is less constrained than the VRPTW, the IB_X crossover and the suite of five mutation operators in [16] are also used here. An approach inspired from the granular tabu search [185] is also exploited to reduce the number of possible insertion places when applying the genetic operators. Basically, insertion places for a given customer are restricted to surrounding routed customers located within a certain distance limit.

In the work of Potvin and Bengio [128], a genetic algorithm called GENEROUS is proposed to solve the VRPTW. One interesting feature of this work is that the encoding issue is avoided by applying the genetic operators directly on the solutions. A route-based crossover RBX is proposed where two routes from two different parent solutions are exchanged to form offspring. Another crossover operator, called SBX, is based on 2-opt* exchanges [134], where the end parts of two routes, one in each parent solution, are exchanged. In both cases, a repair operator is applied to produce feasible solutions, that is, duplicate customers are eliminated and missing customers are reinserted using a least-cost insertion heuristic. A mutation operator is specifically designed to empty small routes, by inserting customers one by one into other routes. Another mutation operator is a primitive ejection chain [56], where a customer can eject another customer from another route. The ejected customer is then feasibly inserted, if possible, at some other location. Apart from these true mutation operators, a local search heuristic based on Or-opt exchanges [118] is also applied with a certain probability to further improve the solution. The same SBX crossover operator is used in [120]. The authors also generalize this idea by designing a new crossover operator, inspired from the CROSS exchanges [160], where intra-route segments from two parent solutions are exchanged to create offspring. Similar operators for the VRP and VRPTW are described in [124, 144, 171, 172, 173]. Using a multi-part solution representation called GVR, where each part represents an individual route, the authors first design a crossover operator that

removes an intra-route segment from one parent and inserts it after the closest customer in a route of the second parent to create an offspring. Duplicates are then removed. A crossover operator similar to RBX is also proposed where a full route is taken from one parent and transferred to the other parent to create an offspring. Duplicates are then eliminated.

The GA for the VRPTW of Bräysy and Dullaert [25] emphasizes fast computation times. The algorithm is divided into two different stages that are applied a number of times within a multi-start framework. The first stage corresponds to the creation of an initial solution. The second stage corresponds to the improvement of the initial solution and is divided into two different phases. The first phase is deterministic and minimizes the number of routes using ejection chains [56]. The second phase exploits ideas found in [24] by evolving a population of routes with the aim of minimizing the total distance. The initial population is made with the routes obtained at the end of the first phase. Parent routes are paired randomly and the offspring routes are required to contain all customers found in the parent routes. The proposed $ICROSS$ operator is based on CROSS exchanges [160], although exchanges that are not likely to lead to feasible routes are filtered out to limit the computational burden. The reinsert related customers crossover RRC removes a set of customers from two parent routes, and then tries to reinsert them, as in [152]. After reinsertion, the customers in both routes are reordered with Or-opt exchanges [118]. When no further improvement can be obtained via $ICROSS$ and RRC, a mechanism that allows the search to explore other regions in the solution space is applied. Basically, the total waiting time along the routes is introduced in the objective and its weighting coefficient is gradually reduced until it vanishes, in a manner reminiscent of the cooling schedule of simulated annealing [84]. At the end, each offspring route is improved with a local search heuristic based on intra-route CROSS exchanges (i.e., the two segments to be exchanged are taken in the same route).

The ruin-and-recreate principle and large neighborhood search have led to very good results on vehicle routing problems, either within the GA framework or within other metaheuristic schemes (see Section 7). In fact, by controlling the magnitude of the ruin component, it is possible to jump to other regions of the solution space and escape from local optima, while retaining a significant part of the current solution (which should be reasonably good). This is much better than restarting from scratch.

3.6. Column Generation

In the work of Alvarenga and his colleagues, a set partitioning formulation of the vehicle routing problem is exploited to partition the set of customers into a number of routes. These routes correspond to columns in the integer programming model (IP). As it is not possible to enumerate all feasible routes, only a subset of routes is generated in a heuristic way with a GA. The scaled-down set partitioning problem is then solved exactly with an IP solver. It is worth noting that the overall approach is heuristic, because there is no guarantee that the GA will generate all routes needed to get an optimal solution.

In [4, 6], the VRPTW is addressed in this way with the unusual objective of minimizing the total distance only. Many executions of the GA are performed to produce different routes (columns). Within the GA, the crossover operator creates an offspring by alternately selecting a route from the first and second parents, while removing duplicate customers if necessary. The remaining customers are then inserted in the existing offspring routes, if possible. Otherwise, a new route is created. Thus, there is no incentive to reduce the number of routes. Many different mutation operators are proposed to reinsert a customer or exchange two customers, merge a pair of routes or split a route, with or without consideration for an improvement in the objective. The solution obtained after solving the set partitioning problem, based on the routes produced by the GA, is then divided into a number of subsets of routes. The GA is then applied again on each subset of routes to generate additional routes. By working on smaller subproblems, an intensification of the search is obtained. At the end, the final set partitioning problem produced with all columns generated from the start, is solved to get the final solution. This algorithm is shown to improve the best distances reported in the literature on many Solomon's benchmark instances [156], but at the expense of an increase in the number of vehicles. The same approach is proposed in [22]. In this work, however, the focus is on the benefits of a memory where solutions from previous iterations are preserved and used to fill a part of the current population.

In a follow-up work by Alvarenga and his colleagues [3, 5], the algorithm described above is modified to focus first on route minimization and then, on total distance. To this end, an independent GA run is performed at the start to reduce the number of vehicles. When selecting parent solutions, the hierarchical or lexicographic order of criteria proposed in [68] is applied (see Section 4), where the focus is on the route with the smallest number of customers. However, the authors note that additional criteria can be useful to distinguish

between two solutions. They propose criteria of their own based on the so-called taker route, which is the route with minimum time window violations when a customer from the smallest route is inserted in it. The mutation operators from their previous work in [4] are also modified because some of them are not appropriate when the number of vehicles is considered (e.g., the mutation operator that splits a route). The new operators are aimed at emptying the smallest route, in particular through primitive forms of ejection chains [56]. The final population produced by this GA, then becomes the initial population for the GA in [4] which minimizes the total distance.

3.7. Landscape Analysis

The work reported in this section is based on an analysis of the solution space topology. As observed in [87], most instances of the VRP exhibit global convexity, which means that global optima are typically found at the bottom of big valleys (see Figure 6). Although solutions that are close to one another do not have to preserve strict convexity (and thus, are local optima), there is a general trend toward convexity. This observation implies that good solutions, like local optima, should be close to one another and should have similar features or characteristics. In [87], a GA coupled with local search is proposed to exploit global convexity. Crossover operators are designed to generate offspring that are close to their parents, using similarity measures based on common edges and customers assigned to the same vehicle. The Cluster Preserving Crossover CPX preserves groups or clusters of customers that are common to both parents. Clusters in the offspring are formed by intersecting pairs of routes that share the largest number of customers. Routes are then created from these clusters by sequencing the customers. The Common Edges Preserving Crossover $CEPX$ preserves edges. It identifies paths of maximum cardinality that are common to both parents and uses them to create routes in the offspring. Finally, a mixed crossover operator $CECPX$ is designed to preserve both characteristics. It creates routes through paths of maximum cardinality within groups or clusters of customers that are common to both parents.

In a follow up work [88], similar ideas are explored to accelerate the local search in the GA. First, a cache is used to store the change in the objective value when a move from the current solution to a neighboring solution is performed. Since a move leaves most of the solution unchanged, many values stored in the cache are still valid for the new current solution, and can be directly fetched when evaluating the neighborhood of this solution. As opposed to the TSP, however, the capacity constraints in the VRP ask for an extension of the cache to store

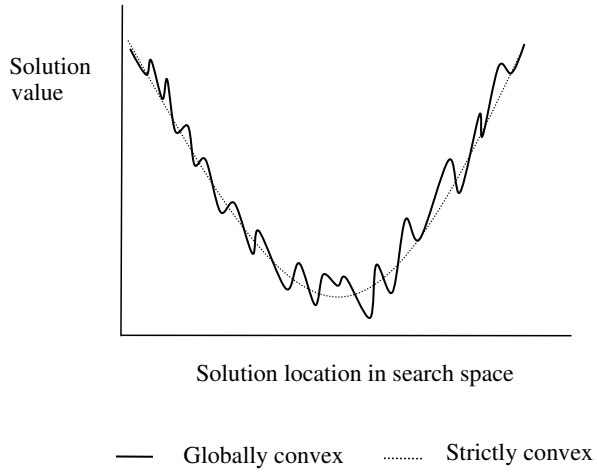

Figure 6: Global convexity

the feasibility status of a move. Unfortunately, due to the non negligible overhead associated with the management of the cache, the authors did not find any significant speed-up with the cache. The second technique is based on the global convexity property and emphasizes closeness when the local search is applied. In this work, after applying the edge-preserving operator $CEPX$, the local search explores a restricted neighborhood, based on 2-opt [96] and customer exchanges, where any move that would remove a common edge is forbidden.

In [73], the same ideas are exploited for a vehicle routing application in a waste management company. Starting from a central operating base, each vehicle visits a number of sectors that contains waste containers. When the vehicle is full, it goes to a dumping site to unload, before visiting the next sector. A route for an operations day is thus made of a number of route segments between two dumping sites (the dumping sites at the start and end of the day correspond to the operating base). A GA, coupled with a local search that moves sectors from one vehicle to another, is proposed to solve this problem. A crossover operator is developed to generate offspring solutions that are not too far from the parents. The features that are preserved in the parents are (1) the percentage of common arcs, (2) the percentage of common assignments of sectors to vehicles, (3) the percentage of common assignment of arcs to vehicles and (4) the percentage of common pairs of sectors assigned to a single route segment. First, each vehicle route in parent 1 is transferred to the offspring with probability 0.5. Then, the route segments in parent 2 are inserted, whenever possible, in a route segment associated with the same vehicle in the offspring.

3.8. Composite and Hierarchical Approaches

The term composite refers here to approaches where different, loosely integrated, metaheuristics are exploited to solve a problem. These approaches benefit from the strengths (while alleviating the weaknesses) of the various metaheuristics involved, leading to more robustness. One example is the GENSAT system which uses simulated annealing, tabu search and a GA to solve the VRPTW [176]. First, customers are clustered with the genetic sectoring heuristic GENSECT (see Section 3.3) and routes are obtained by sequencing the customers within each cluster with a least-cost insertion heuristic. Then, the solution is improved with either a local search, tabu search, simulated annealing or a hybrid of tabu search and simulated annealing. In the latter case, a probabilistic simulated annealing acceptance criterion is embedded within the tabu search. The neighborhoods of the improvement heuristics are all based on 2-interchanges [119]. A number of variants are obtained and regrouped into the GENSAT system by generating the initial solution either with GENSECT or Solomon's I1 heuristic [156] and by applying the various improvement procedures. At the time, GENSAT produced 40 new best solutions over 60 test instances, namely the 56 Solomon's benchmark instances [156] plus four instances in [143]. It also matched the best known solution on 11 instances. To be fair, GENSAT can hardly be seen as a single algorithm, but rather a collection of algorithms that are applied on a given instance, with the best solution returned at the end. In fact, the results showed that the best solutions were produced with different algorithms, depending on the instance.

A parallel algorithmic scheme is proposed by LeBouthillier and Crainic in [91], where different tabu searches and GAs, explore the search space concurrently. The interaction between these various metaheuristics is realized through a central warehouse, basically an adaptive memory [140], made of elite solutions. When a metaheuristic finishes its current execution and returns its best solution, this solution is included in the warehouse if it is better than the worst solution. The diversity of solutions in the warehouse is exploited to provide new starting solutions for the metaheuristics. The proposed GAs in this system are quite standard and are based on the ER [191] and OX [115] crossovers. Through a parallel exploration of the search space, the authors have found solutions that are competitive with the best approaches for the VRPTW on Solomon's [156] and Gehring and Homberger's [47] benchmark instances. A first refinement of this approach is proposed in [92], where a guidance mechanism is added through the identification of common patterns in over-average,

average or under-average solutions in the warehouse. Frequent patterns associated with over-average solutions are fixed during a number of iterations to provide intensification. Conversely, frequent patterns associated with under-average solutions are prohibited to provide diversification. A second refinement in [90] dynamically determines when to apply the pattern-based intensification and diversification phases, based on the entropy of the solution warehouse. This entropy or uncertainty is higher when the number of different solution elements, like arcs, is larger, thus triggering intensification. Conversely, when the number of different solution elements is smaller, diversification is triggered.

In [74], a single vehicle PDPTW is addressed with a dynamic programming algorithm and a GA. If the dynamic programming algorithm cannot find an optimal solution within the allocated time, the partial solution is used to seed the initial population of the GA. This approach greatly speeds up the solution process, thus allowing applications in dynamic settings (although no such application is studied). It also allows the GA to find much better solutions when compared with a randomly generated initial population.

In [123], a hierarchical GA is used to solve a variant of the VRP where the objective is to minimize the length of the longest route based on a Manhattan distance metric. At the lower level, solutions evolve within subpopulations called niches. At the upper level, an evolutionary mechanism is applied to the niches themselves. At each upper level iteration, a number of new niches are created by merging a bad niche with a good one (where the quality of a niche is defined by its best solution). Then, the best niches obtained replace the worst ones in the original population of niches. At the lower level, solutions evolve within each niche through the application of crossover, mutation and local optimization operators. The crossover operator removes a segment of route from the first parent, remembers the immediate predecessor of the first node of the segment, finds the predecessor node in the routes of the second parent and inserts the segment just after it. Then, duplicates are removed. This is repeated as long as a feasible offspring is not obtained. The mutation operator removes a route segment and inserts it into another route. Local optimization is performed by iterating over a number of classical neighborhood structures, while always maintaining feasibility. As observed in this work, the best solutions in different niches are often the same. Accordingly, an elitist population replacement strategy, where the best solution is transferred to the next population, is applied to only one of these niches. This best solution is thus likely to disappear from the other niches. A newspaper distribution application illustrates how this two-level GA behaves.

3.9. Multi-Objective Approach

As mentioned before, vehicle routing problems often involve conflicting objectives, like minimizing the number of vehicles versus minimizing the total distance. This issue is often addressed though a weighted linear sum that aggregates the objectives or through a hierarchical approach, where the first objective is optimized followed by the second objective, while taking care not to degrade the value of the first objective. In this section, we review Multi-Objective Genetic Algorithms (MOGAs) which address vehicle routing problems using a true multi-objective approach. As opposed to the methodologies presented in the previous sections, MOGAs do not end up with one "best" solution, but rather a collection of (non-dominated) solutions. In a practical context, it would thus be possible to generate different solutions for a decision maker who would then choose a particular solution based on other, more subjective, criteria.

MOGAs are often inspired from the non-dominated sorting GA reported in [35, 157]. This GA is characterized by a selection procedure based on Pareto ranks. Basically, non-dominated solutions in the current population are ranked first and get the same evaluation (which can be the rank itself). After removing these solutions, the non-dominated solutions in the reduced population are ranked second and get a lower evaluation, etc. Parent selection is then biased toward solutions with smaller Pareto ranks, usually via a tournament selection procedure. To obtain a diversity of solutions along the Pareto frontier, the value of each solution is also divided by a quantity proportional to the number of solutions that are close to it (this is called *sharing* [59]). Hence, if many solutions are in the same region along the Pareto frontier, these solutions get a relatively low evaluation and are less likely to be selected as parents. Their number will thus tend to diminish. Finally, while the GA is executed, an archive that contains all non-dominated solutions found during the search is maintained. This archive can also be used by the genetic operators to intensify the search around these solutions.

The authors in [117] address the VRPTW as a true bi-ojective problem, where the two objectives are the number of vehicles and total distance. They used a MOGA with Pareto-ranking to this end. The crossover operator, called $BCRC$ for Best Cost Route Crossover, works as follows. Routes 1 and 2 are first selected in parents 1 and 2, respectively. Then, customers selected in route 1 are removed from parent 2 and customers selected in route 2 are removed from parent 1. The removed customers are finally reinserted in each parent, in

random order, with a least-cost insertion heuristic. Due to the time windows, a constrained inversion mutation operator is designed, where the length of the inverted segment is limited to three consecutive customers. The VRPTW is also addressed by Tan et al. [165, 167] with a similar approach, except that the crossover operator is RBX [128]. Also, three different route-based mutation operators and local search heuristics are proposed to further improve the solutions. The same authors have adapted their bi-objective GA to a more complex real-world problem [166, 168]. Here, a company must visit customers with time windows using different combinations of trucks and trailers. A truck-trailer pair moves between a port, customer warehouses, trailer exchange points (where different types of trailers are available) and container depots (where empty containers are available). Each job is made of a number of tasks, and each task is made of an origin and destination point. The RBX crossover operator in [167] is applied, as well as a route merge mutation. A local search heuristic that tries to merge small routes into larger routes is also used to further improve the solutions. In all cases, infeasible tasks are outsourced to other companies.

In [77, 78], a bi-objective VRP is also addressed, but with the objectives of minimizing the total distance and balancing the route lengths (more precisely, minimizing the difference between the longest and shortest routes). To obtain solutions that are dispersed along the Pareto frontier, a diversification mechanism based on elitism is proposed. As is usually the case, an archive contains all non-dominated solutions found during the search. Some solutions in this archive are included into the population at each iteration to provide a form of intensification. Diversification is obtained by maintaining additional archives of non-dominated solutions where one of the objectives is inverted (i.e., one objective is maximized rather than minimized). This approach is used in a context where different subpopulations evolve in parallel and are connected through a toroidal grid topology. Each subpopulation maintains the standard archive plus an additional archive where one of the original objectives is inverted. When the best solutions migrate from one subpopulation to another to refresh the genetic pool, each subpopulation also sends its standard archive to all neighboring subpopulations. However, the additional archive is sent only to neighbors with the same inverted objective. The non-dominated sorting GA [157] applied within each subpopulation exploits the RBX and SBX crossover operators [128]. This MOGA is hybridized with a tabu search, which is applied only at the end. The tabu search evaluates each solution in the neighborhood of the current solution based on its dominance relation with the current solution. Overall, the additional archives are shown to improve diversity along the approximate Pareto frontier

generated by this GA. A related approach for a bi-objective VRPTW based on the number of vehicles and total distance, but without parallel subpopulations and diversification archives, is reported in [139].

A multi-objective pick-up and delivery problem with time windows, motivated by a courier service application, is addressed in [93]. Here, transportation requests are received over the day and must be assigned to vehicles in real-time. Multi-criteria utility functions, aimed at evaluating the quality of a vehicle for visiting a transportation request, are evolved with a GA to approximate the (unknown) utility function used by a human dispatcher to take decisions. Basically, the GA evolves the weights of a weighted linear utility function, using real-valued strings. In a follow-up work [12], Genetic Programming (GP) [85] is used for the same purpose. GP is an evolutionary algorithm closely related to GAs, but where tree structures, rather than linear structures, are evolved. These trees encode computer programs that are made of a predefined set of functions and terminals. In the case of vehicle dispatching, the terminal set contains attributes like the detour or the delay introduced in the current vehicle route by a new request, while the functions are basic mathematical operators. This is an interesting application of GP because vehicle utility functions (programs) that are much more complex than weighted linear functions can be generated.

3.10. Parameter Optimization

Genetic algorithms have also been used to search for good parameter values for optimization algorithms. As these parameters are often real-valued, both bit strings and real-valued strings have been proposed for this purpose [103]. With a classical bit string representation, a lower and an upper bound on the parameter values are defined, and the bit string is then interpreted as an integer which is mapped within the parameter interval. The GA thus searches in a discrete space that corresponds to a sampling of the real interval. The sampling is more refined when the bit string is longer, but the search space then increases accordingly.

In [11], the parameter values of Solomon's I1 heuristic [156] are optimized with a GA on Solomon's VRPTW benchmark instances. Each parameter is encoded as a bit string and these strings are appended to encode groupings of parameter values. A GA based on classical one-point crossover and mutation operators is used to evolve these parameter groupings. The same approach is proposed in [129, 130] to optimize the parameter values of a parallel insertion heuristic for the VRPTW [133].

In [187], the author analyzes how the various parameters of a GA impact its performance when solving the VRP. This work thus addresses parameter optimization for the GA itself.

4. Evolution Strategies

The Evolutions Strategies (ES) paradigm was developed in the 60's and 70's at the Technical University of Berlin by Rechenberg and Schwefel (for an introductory text on the subject, see [19]). Although mostly applied to real-valued function optimization, discrete variants have emerged for solving combinatorial optimization problems. The algorithmic framework of ES is similar to genetic algorithms as they evolve a population of solutions through mutation and recombination (crossover) operators. One distinctive feature is the concurrent evolution, with the same genetic operators, of the so-called strategy parameters which are usually properties of the mutation operator, like the mutation step size to be applied to each element of the solution. In ES, the recombination operator produces a single offspring and either involve two parents or is extended to a variable number of parents, known as a family of parents. Also, the population replacement mechanism is completely deterministic. That is, for a population of size μ, either the μ best solutions are selected out of λ offspring solutions ($\mu < \lambda$), which is known as a (μ, λ)-selection, or out of the union of the parent and offspring solutions, which is a $(\mu + \lambda)$-selection. The latter scheme is elitist and guarantees a monotonically improving performance. A special case is the $(1 + 1)$-ES where there is no recombination operator, only a mutation operator. The ES search framework can be summarized as follows (note the similarities and differences with the pseudo-code of GAs):

1. Create an initial population of μ individuals, where each individual contains a solution and its associated strategy parameters.

2. Evaluate each solution.

3. Repeat for a fixed number of iterations:

    3.1 Repeat until λ offspring are created:

        3.1.1 Randomly select ρ parents in the population (with replacement).

        3.1.2 Apply recombination to the parent solutions to create a single offspring solution.

3.1.3 Apply recombination to the parent strategy parameters to create the offspring strategy parameters.

3.1.4 Append the offspring solution and strategy parameters to create a complete individual.

3.1.5 Apply mutation to the offspring solution.

3.1.6 Apply mutation to the offspring strategy parameters.

3.2 Evaluate each offspring solution.

3.3 If (μ, λ)-ES then select the μ best individuals to create the new population.

3.3 If $(\mu + \lambda)$-ES then select the μ best individuals among the union of the μ parents and λ offspring to create the new population.

4. Return the best solution found.

Evolution strategies, as depicted above, have led to some very good results on difficult problems, in particular because they offer a mechanism to self-adapt the parameter values within an evolutionary framework. The first application of ES to a vehicle routing problem is found in the work of Homberger and Gehring [68], where the VRPTW is addressed with a (μ, λ)-ES. The mutation operator is a local search heuristic and the mutation step size then corresponds to the number of moves or modifications to the current solution. There is an additional strategy parameter associated with the mutation operator which is the objective to be favored (either minimization of the total distance or minimization of the number of vehicles). Two evolution strategies, called *ES1* and *ES2* are proposed. In the simple variant *ES1*, there is no recombination, only a mutation operator that modifies the solution based on the following: a random choice between reinsertion of a customer, exchange of two customers or 2-opt* [134]. This is followed by a specialized Or-opt [118] aimed at reducing the number of routes (when this objective is favored) by repeatedly inserting customers from the smallest route into alternate routes. It is worth noting that the two mutation strategy parameters are directly inherited by the offspring without any modification. This is opposed to the second variant *ES2*, where a uniform crossover operator [158] is applied to the mutation step size of the parents to generate new values.

The authors note that there is a bias toward the distance, when one tries to simultaneously minimize the distance and the number of vehicles. To put more emphasis on the number of vehicles, the authors propose a new approach to select the solutions that will be part of the

new population. First, the λ offspring solutions are evaluated using the following hierarchical ordering of four criteria: number of vehicles, number of customers in the smallest route, minimal time delay introduced in the solution when all customers in the smallest route are inserted into other routes, total distance. The second and third criteria measure how easily a route can be eliminated from the solution, considering the time window constraints. Based on this ordering, the best κ ($\kappa < \mu$) offspring solutions are put in the new population. Then, the λ offspring are sorted again based on the original objective which is to minimize the number of vehicles first and total distance second. The best $\mu - \kappa$ offspring are then selected to complete the new population. Depending on the ratio κ/μ, the minimization of the number of vehicles is more or less emphasized with regard to the distance.

In a follow-up paper [69], the authors describe a two-stage approach where $ES1$ is first applied with the objective of minimizing the number of vehicles, followed by a tabu search heuristic to minimize the total distance. This work is an extension of a two-phase algorithm from the same authors, where a $(1, \lambda)$-ES was used [47]. A parallel implementation of this algorithm is reported in [48, 49]. Using a coarse-grained architecture made of a number of workstations, the two-stage algorithm is executed in parallel on each machine, but with different starting points and parameter configurations. These independent search threads then communicate periodically by exchanging their best solution (to allow jumps in the search space).

In [100], an algorithm called AGES combines guided local search (GLS) [189, 190] and ES in an iterative two-stage procedure to address both the VRP and VRPTW. GLS introduces modifications into the objective function through penalties when the search gets trapped in a local optimum. Here, the penalty counter of one arc in the solution is incremented by one, each time a local optimum is reached. The arc is selected on the basis of its length (a long arc is more likely to be penalized) and current penalty counter (an arc with a high penalty counter is less likely to be penalized again). In the first stage, GLS controls the objective function of a composite local search, based on reinsertion of a customer, exchange of two customers and 2-opt* [134]. The neighborhoods considered are restricted to routes which are close to the currently penalized arc. This is called a penalty variable neighborhood. If no improvement is found for a number of iterations, the first stage is stopped and the second stage starts. In the second stage, GLS controls the objective function of a $(1+1)$-ES. In this ES, mutation is performed on a parent solution, where the number and type of modifications considered each time are randomly chosen. When the offspring is better than the parent, it

replaces the parent. The ruin and recreate principle is at the core of the proposed mutation. That is, a number of customers are removed from the current solution and are reinserted at least cost. Once again, a penalty variable neighborhood is used to restrict the search.

A refinement of this algorithm for the VRP in reported in [101]. In this implementation, the composite local search is based on some additional intra- and inter-route moves, like Or-opt [118]. The authors also explore different parameter configurations and neighborhood combinations in the composite local search, and identifies what they call a *best* AGES and a *fast* AGES. Another variant of this algorithm is proposed in [102] where the objective function is not modified with GLS. Since there are no penalized arcs, the penalty variable neighborhood cannot be used to restrict the search. Instead, an adaptive variable neighborhood search [104] is proposed based on a partition of the service area into smaller rectangular areas. The original problem is thus divided into smaller subproblems, based on the geographic proximity among customers. The rectangular areas are then processed in a given order. Afterward, they are gradually merged back together until the original service area is restored. The idea is to work on smaller problems when a lot of (time-consuming) improvements are still possible, and to gradually enlarge the neighborhood as the number of potential improving moves diminishes.

5. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based metaheuristic where each solution, called a particle, is associated with an objective value and a velocity [83]. The latter defines a direction in the search space, as well as an amplitude, for modifying the solution and is determined through interaction with other solutions. The velocity basically replaces operators like crossover and mutation in classical GAs. The velocity of particle i at iteration t is obtained through a randomized weighted sum of three components: the velocity of particle i at iteration $t-1$, the velocity of the best solution achieved by particle i and the velocity of the best solution achieved over all particles (or over particles that are sufficiently close to particle i, depending on the variant). Although PSO has mostly been used for solving continuous optimization problems, discrete variants also exist.

In [31], a binary version of PSO is used, where a solution or particle is a binary vector. The velocity vector is made of values between 0 and 1 and the corresponding solution is obtained by interpreting each velocity value as the probability that the corresponding element

in the solution is 0 (and conversely, as 1 minus the probability that the corresponding element is 1). This approach is exploited to solve the VRP. Basically, a particle is a vector of size $n \times m$, where n is the number of customers and m the number of vehicles. That is, the vector of customers is duplicated m times, one time for each vehicle. An entry of 1 means that the customer is served by the corresponding vehicle. The PSO thus solves the assignment subproblem. As the solution obtained is not necessarily valid (i.e., exactly one position associated with a customer should be equal to 1), a repair operator is used. If the solution is valid but infeasible, the solution is discarded and the velocity is recalculated until a feasible solution is obtained. A simulated annealing heuristic is then used to sequence the customers in each route.

In [195], an alternative particle representation with only $n + m - 1$ elements is used for solving the VRPTW. Each entry corresponds to the position of one of the n customers or one of the $m - 1$ copies of the depot in the solution vector. The PSO thus solves the assignment and sequencing subproblems concurrently. The objective value is the total distance plus penalties for violations of the capacity and time window constraints. The velocity is a vector of values between -$(n+m-2)$ and $n+m-2$ which is added to the solution vector to update it. Normalization is then applied to obtain new integer positions.

6. Classification by Problem Types

The previous review was divided along methodological lines. The aim of this section is to group all references according to vehicle routing problem types addressed with EAs. These types have already been identified and briefly described in Section 2: the capacitated vehicle routing problem (VRP), VRP with time windows (VRPTW), VRP with time deadlines (VRPTD), Periodic VRP (PVRP), Time-dependent VRP (TVRP), VRP with Heterogeneous Fleet (VRPHF), Multi-Depot VRP (MDVRP), VRP with backhauls (VRPB) and Pickup and Delivery Problem (PDP). The catch-all type *Others* is used to refer to real-world vehicle routing applications that do not fall into any of these categories due to problem-specific features. The row headings in Table 1 correspond to the various problem types, while the column headings correspond to EAs.

Clearly, a high concentration of papers report applications of GAs. But, the few implementations of ES have been very successful (see Section 7). EAs have mostly been applied to the VRPTW, followed by the VRP, which are also the most studied problems in the ve-

Table 1: EAs for different problem types

Problem \ EA	GA	ES	PSO	GP
VRP	[1][2][8][9][13][14][42][63] [64][65][77][78][86][87][88] [98][109][123][124][137][144][147] [155][172][173][187]	[101][102]	[31]	
VRPTW	[3][4][5][6][11][15][16][17][18] [20][22][24][25][30][32][66][70] [71][82][91][92] [97][99][105] [110][116][117][120][128][129][130] [132][139][161][162][163][164] [165][167][171][172][173] [174][175][176][177][178] [179][181][184][193][194]	[47][48][49] [68][69][100]	[195]	
VRPTD	[180][183]			
PVRP	[37][188]			
TVRP	[62][80]			
VRPHF	[113][114][170]			
MDVRP	[41][43][145][153][154][181] [182]			
VRPB	[46][131][170]			
PDP	[74][76][79][93][121][122][126] [148][149][150]			[12]
Others	[73][146][166][168]			

hicle routing literature in general. Overall, the number of papers devoted to the VRPTW is substantially larger than those devoted to the VRP. This can be explained as follows. When the first applications of EAs for vehicle routing appeared in the literature, the VRPTW was much less studied than the classical VRP. Thus, there were more opportunities for researchers to improve the best results reported in the literature on VRPTW benchmark instances. Accordingly, the first EAs were often developed for the VRPTW and only later adapted to the VRP, even if the VRP can be considered as an easier problem, due to the absence of time constraints.

7. Computational Performance

The aim of this section is to evaluate how evolutionary algorithms compare with other metaheuristic approaches for solving the VRP, VRPTW and PDPTW. These three problem

classes have been chosen because of their importance in the vehicle routing literature and the availability of benchmark instances that allow a fair comparison between competing methods.

7.1. VRPTW

This section focuses on the most effective problem-solving approaches for solving the VRPTW on Solomon's [156] and Homberger and Gehring's [47] benchmark instances. It just happens that EAs, either alone or combined with other metaheuristics, are currently among the best approaches for solving the VRPTW. Another powerful approach is the large neighborhood search [152], based on the ruin-and-recreate principle [151], where customer removal heuristics are applied to destroy a part of the current solution before reconstructing it. All work reported here minimizes the number of vehicles first, followed by the total distance.

7.1.1. Solomon's instances

Solomon's 100-customer instances [156] are divided into six problem classes depending on the geographical distribution of the customers and the width of the time horizon. The latter is set by a time window at the depot which defines an earliest start time and latest end time for each vehicle route. The customers are randomly distributed in instances of type R, while they are clustered in instances of type C. Instances of type RC are a mix of types R and C. With regard to the time horizon, instances of type 1 have a short horizon and each vehicle can only visit a small number of customers. Conversely, instances of type 2 have a long horizon and each vehicle can visit a large number of customers. Overall, these characteristics lead to classes $R1$, $R2$, $C1$, $C2$, $RC1$ and $RC2$, with 8 to 12 instances in each class. All instances are Euclidean and the time units correspond to the distance units (i.e., the travel time between two customers is the same as the Euclidean distance).

Table 2 contains the best results reported in the literature, at the time of writing, on Solomon's benchmark instances. The two rows in each entry correspond to the average number of vehicles and total distance, respectively, for the corresponding problem class and problem-solving method. At the end, the cumulative number of vehicles (CNV) and cumulative total distance (CTD) over the 56 instances are shown for each method. The computer type (CPU), the average time in seconds T(s) for solving each instance once and the number of runs (Runs) to obtain the reported results are also indicated. In the case of CPU, O stands for Opteron, P stands for Pentium, X for Xeon, SU for Sun Ultra, M for

MHz and G for GHz. Note also that $u \times v$ in row T(s) means that the algorithm was run for v seconds on a parallel architecture with u processors.

The following algorithms are reported in Table 2:

- B is the reactive variable neighborhood search (VNS) of Bräysy [23]. In this algorithm, the number of vehicles is first minimized with an ejection chain-based approach [56]. Then, the distance is minimized with VNS.

- *BBB* is the GA of Berger et al. [16] where two populations are evolved (see Section 3.5). In the first population the distance is minimized based on a fixed number of routes; in the second population, time constraint violations are minimized on solutions with one fewer route.

- *BVH* is the two-stage hybrid local search of Bent and van Hentenryck [10] where simulated annealing is used in the first stage to minimize the number of vehicles and a large neighborhood search is used in the second stage to minimize the distance.

- *HG* is the algorithm of Homberger and Gehring [69] where ES is used in the first stage to minimize the number of routes and a tabu search is used in the second stage to minimize the distance (see Section 4).

- *IAL* is the problem-solving approach of Ibaraki et al. [72], where local search heuristics are embedded within different iterated and multi-start schemes.

- *LC* is the parallel search framework of Le Bouthillier and Crainic [90] where GA and tabu search threads interact through a common solution warehouse (see Section 3.8).

- *PDR* is the recent large neighborhood search of Prescott-Gagnon, Desaulniers and Rousseau [136] which uses a heuristic branch-and-price method to explore the neighborhoods.

- *PR* is the adaptive large neighborhood search of Pisinger and Ropke [125] which adaptively chooses among different removal and insertion heuristics to intensify and diversify the search.

With regard to solution values, all methods reached the same number of vehicles, both globally and for each instance class. They only differ in total distance. The results are the

Table 2: Solomon's instances

	B	BBB	BVH	HG	IAL	LC	PDR	PR
R1	11.92	11.92	11.92	11.92	11.92	11.92	**11.92**	11.92
	1222.12	1221.10	1211.10	1212.73	1217.40	1213.06	**1210.34**	1212.39
R2	2.73	2.73	2.73	2.73	2.73	**2.73**	2.73	2.73
	975.12	975.43	954.27	955.03	959.11	**952.73**	955.74	957.72
C1	**10.00**	10.00	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	828.38	828.48	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**
C2	**3.00**	3.00	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	589.86	589.93	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**
RC1	11.50	11.50	11.50	11.50	11.50	11.50	**11.50**	11.50
	1389.58	1389.89	1384.17	1386.44	1391.03	1385.14	**1384.16**	1385.78
RC2	3.25	3.25	3.25	**3.25**	3.25	3.25	3.25	3.25
	1128.38	1159.37	1124.46	**1108.52**	1122.79	1129.16	1119.44	1123.49
CNV	405	405	405	**405**	405	405	405	405
CTD	57710	57952	57273	**57192**	57444	57325	57240	57332
CPU	P-200M	P-400M	SU 10	P-400M	P3 1G	X 3.6G	O 2.3G	P4 3G
T(s)	4950	1800	7 200	n/a	15 000	5×720	1800	146
Runs	1	3	n/a	n/a	1	1	5	10

same, for all practical purposes, on classes $C1$ and $C2$ which contain the easiest instances. The recent large neighborhood search PDR of Prescott-Gagnon, Desaulniers and Rousseau is the best on two of the four remaining classes, namely $R1$ and $RC1$. The parallel algorithm of LeBouthillier and Crainic LC and the ES of Homberger and Gehring HG are the best on $R2$ and $RC2$, respectively. Overall, the best cumulative total distance is obtained by HG. However, the perspective changes when the computation times are considered. The results of HG have been obtained over the course of many experiments involving different parameter settings, and no computation times are reported. On the other hand, the results of PDR, PR and LC have all been obtained over a fixed number of runs (best of five runs, best of ten runs and a single run, respectively) using fixed parameter settings. When considering the whole picture, the adaptive large neighborhood search PR of Pisinger and Ropke deserves a special mention, as it is very fast and still reaches a CTD which is at only .2% of the one obtained with HG.

7.1.2. Gehring and Homberger's instances

Gehring and Homberger [47] have designed larger VRPTW instances with 200, 400, 600 and 1 000 customers. For each size, six different classes, with 10 instances in each class, have been generated using the procedure of Solomon described in [156]. Thus, there are 60 instances for each problem size and a total of 300 instances. In Table 3, only the CNV and CTD values are reported for each problem size. They are shown in the first two rows of each entry. The computer type, the average computation time in minutes for solving an instance once and the number of runs to obtain the reported results are indicated in the next three rows. In the case of the computer type, A stands for AMD Athlon.

At the time of writing, the best methods for these larger instances are (apart from BVH, LC, PDR and PR which have already been introduced in the previous section):

- *BHD* is the multi-start local search of Bräysy, Hasle and Duallert [28]. This algorithm is divided into two stages. The first stage minimizes the number of vehicles using an ejection chain-based mechanism, while the second stage minimizes the distance using CROSS-exchanges. At the end, a deterministic threshold accepting variant of simulated annealing [39] is applied to further reduce the distance.

- *GH99* and *GH01* stand for the parallel algorithms of Gehring and Homberger where a $(1, \lambda)$- and (μ, λ)-ES, respectively, are applied in the first phase to minimize the

number of routes [47, 48]. A tabu search is used in the second phase to minimize the distance (see Section 4).

- *MB* is the AGES algorithm of Mester and Bräysy [100] that combines ES, GLS and local search (see Section 4).

On these larger instances, the large neighborhood search *PDR* of Prescott-Gagnon, Desaulniers and Rousseau produce the best solution values for each problem size. The adaptive large neighborhood search of Pisinger and Ropke, however, is still noteworthy with excellent solution values obtained in very fast computation times. Admittedly, the EA-based methods trail a little bit on these instances, but the ES-based algorithm *MB* of Mester and Bräysy and the parallel algorithm *LC* of LeBouthilllier and Crainic still produce competitive results in reasonable computation times.

7.2. VRP

The previous section has shown that EAs are at the core of competitive approaches for the VRPTW. This is also true for the capacitated VRP. There are many VRP benchmark instances in the literature, including those of Christofides, Mingozzi and Toth (CMT) [33], Golden et al. (GAL) [61], van Breedam [186], Augerat et al. [7] and Li, Golden and Wasil [94]. Some real-world VRP instances are also found in [44, 159]. In this section, results are reported for methods that have been applied to both CMT and GAL, the two most widely used data sets in the literature. There are 14 benchmark instances in CMT, ranging from 50 to 199 customers, and 20 larger instances with up to 483 customers in GAL.

The best methods for minimizing the total distance on these benchmark instances are:

- *DK* is the solution attribute-based tabu search of Derigs and Kaiser [36]. Results are shown for the variant *DKP* where the initial routes are constructed in parallel with the savings heuristic of Clarke and Wright [34] and the variant *DKS* where the routes are constructed one by one with the sequential savings heuristic.

- *LGW* is the record-to-record travel heuristic of Li, Golden and Wasil [94]. This heuristic is a deterministic variant of simulated annealing, where a new solution is accepted if its value is within an acceptable margin of the best known solution value (the record) [38].

Table 3: Gehring and Homberger's instances

		BHD	BVH	GH99	GH01	LC	MB	PDR	PR
200									
	CNV	695	697	694	696	694	694	**694**	694
	CTD	172 406	171 715	176 180	179 328	169 903	168 573	**168 556**	169 042
	CPU	A-700M	SU 10	P-200M	P-400M	X 3.6G	P4 2G	O 2.3G	P4 3G
	T(m)	2.4	n/a	4×10	4×2.1	5×10	8	53	7.7
	Runs	3	n/a	1	3	1	1	5	10
400									
	CNV	1391	1393	1390	1392	1388	1389	**1385**	1385
	CTD	399 132	410 112	412 270	428 489	396 159	390 386	**389 011**	393 210
	CPU	A-700M	SU 10	P-200M	P-400M	X 3.6G	P4 2G	O 2.3G	P4 3G
	T(m)	7.9	n/a	4×20	4×7.1	5×20	17	89	15.8
	Runs	3	n/a	1	3	1	1	5	5
600									
	CNV	2084	2091	2082	2079	2084	2082	**2071**	2071
	CTD	820 372	858 040	867 010	890 121	809 882	796 172	**800 797**	807 470
	CPU	A-700M	SU 10	P-200M	P-400M	X 3.6G	P4 2G	O 2.3G	P4 3G
	T(m)	16.2	n/a	4×30	4×12.9	5×30	40	105	18.3
	Runs	3	n/a	1	3	1	1	5	5
800									
	CNV	2776	2778	2770	2760	2759	2765	**2745**	2758
	CTD	1 384 306	1 469 790	1 515 120	1 535 849	1 444 525	1 361 586	**1 391 344**	1 358 291
	CPU	A-700M	SU 10	P-200M	P-400M	X 3.6G	P4 2G	O 2.3G	P4 3G
	T(m)	26.2	n/a	4×40	4×23.2	5×40	145	129	22.7
	Runs	3	n/a	1	3	1	1	5	5
1000									
	CNV	3465	3468	3461	3446	3439	3446	**3432**	3438
	CTD	2 133 376	2 266 959	2 276 390	2 290 367	2 133 673	2 078 110	**2 096 823**	2 110 925
	CPU	A-700M	SU 10	P-200M	P-400M	X 3.6G	P4 2G	O 2.3G	P4 3G
	T(m)	39.6	n/a	4×50	4×30.1	5×50	600	162	26.6
	Runs	3	n/a	1	3	1	1	5	5

- *MB* is the AGES algorithm of Mester and Bräysy [101] (see Section 4). Both the best results *MB Best* and those obtained with a fast version of this algorithm *MB Fast* are shown.
- *N* is the GA of Nagata [109], based on the *EAX* crossover operator (see Section 3.1).
- *P* is the GA of Prins [137], where a giant tour is split into feasible routes (see Section 3.4).
- *PR* is the adaptive large neighborhood search of Pisinger and Ropke [125].
- *T* is the tabu search with adaptive memory of Tarantilis [169].

Table 4 reports the best results obtained with these methods on the CMT and GAL instances. The row *%* refers to the gap in percentage with the best known solutions at the time of writing (this way of reporting results is the norm in the VRP literature). As in the previous tables, row *T(m)* refers to the average computation time in minutes, *CPU* is the type of processor used (where *C* stands for Celeron) and *Runs* is the number of runs.

When considering Table 4, it should be noted that *LGW* has been tested with three different settings in [94] and the results reported here correspond to the setting associated with the best average solution value on each data set. Also, the results of *P* on the CMT instances have been obtained over multiple runs with different parameter settings, while those of *T* on the same CMT instances are based on a single run, but using one of two different parameter settings, depending on the instance. Table 4 shows that the GA of Nagata reaches the optimum on the CMT instances, but was applied on only 7 instances out of 14. Also, its performance degrades on the largest GAL instances. The ES-based algorithm *MB* of Mester and Bräysy looks particularly attractive here. The *Best* variant produces the best solution values on both data sets, if we consider only methods that have been tested on all benchmark instances. Furthermore, the *Fast* variant still produces good solutions, but in a fraction of the computation time required by the other methods (although *LGW* is also quite fast).

7.3. PDPTW

The PDPTW has been included in this study, because this is a problem where EAs have not yet convincingly demonstrated their effectiveness. In fact, not so much work is reported

Table 4: CMT and GAL instances

		DKP	DKS	LGW	MB Best	MB Fast	N	P	PR	T
CMT										
	%	0.28	0.21	0.41[a]	0.03	0.08	0.00[a]	0.24	0.11	0.18
	CPU	C 2.4G	C 2.4G	A 1G	P4 2.8G	P4 2.8G	X 3.2G	P3 1G	P4 3G	P2 400M
	T(m)	4.54	5.84	0.32	2.80	0.05	2.37	5.19	17.50	6.60
	Runs	1	1	1	1	1	10	n/a	10	1
GAL										
	%	0.79	0.87	1.05	0.11	1.05	2.83[b]	1.03	0.60	0.71
	CPU	C 2.4G	C 2.4G	A 1G	P4 2.8G	P4 2.8G	X 3.2G	P3 1G	P4 3G	P2 400M
	T(m)	113.34	112.14	0.97	24.40	0.20	41.37	66.60	107.67	45.48
	Runs	1	1	1	1	1	10	1	10	1

[a] Average on 7 out of the 14 instances
[b] Average on 12 out of the 20 instances

on the PDPTW in general. Benchmark instances for the PDPTW are typically derived from VRPTW instances by pairing two customers (to get a pickup and a delivery node for each customer request). For example, Nanry and Barnes [112] and Lau and Liang [89] have generated PDPTW instances from Solomon's instances [156] in this way. An extended data set with 100, 200, 400, 600, 800 and 1 000 nodes is also reported by Li and Lim [95]. Recently, a new set of instances with randomly generated pickup and delivery nodes has been produced by Ropke, Cordeau and Laporte [141].

The data set of Li and Lim has emerged as a standard in the last few years. Unfortunately, a single paper reports results on a subset of these instances, thus precluding a true assessment of EAs with the best known methods. In a number of cases, home-made instances have been used to experiment with EAs. In other cases, dynamic variants where new customer requests occur over time have been addressed. Furthermore, some authors minimize the total distance only, while others minimize the number of vehicles first (as opposed to the VRPTW, where most researchers have adopted the same objective, there is no consensus yet). The only application of a GA on benchmark instances comes from the work of Pankratz in [121] (see Section 3.3), where results are reported on the nine 100-node instances of Nanry and Barnes and the 56 100-node instances of Li and Lim, with the objective of minimizing the total distance.

On the instances of Li and Lim, the CNV and CTD of the GA of Pankratz are 410 and 58 191, respectively. On the same instances, Ropke and Pisinger [142] report 402 and 58 060, respectively, with an adaptive large neighborhood search with fixed parameter settings, and with the objective of minimizing the number of vehicles first. It means that they improved upon the CTD of the GA, even if it was not their primary objective. Furthermore, the best results reported by Ropke and Pisinger in [142] over a number of experiments with different parameter settings, is much lower at 56 060.

8. Conclusions

This survey has shown that EAs have been applied in many different ways to address difficult vehicle routing problems. Among EAs, genetic algorithms have been the most widely used, although the few implementations of evolution strategies reported in the literature proved to be very successful. The VRP and VRPTW have been the topic of most of these studies, but recent developments are now observed for less known vehicle routing variants, like the

PDPTW. Also, a few real world applications have been reported lately. Due to the flexibility offered by EAs in the handling of side constraints (e.g., penalties, repair operators, decoders in the case of GAs), we expect to see more and more of these real world applications which are characterized by a multiplicity of very specific constraints. Also, dynamic problems requiring fast response times are now addressed with EAs, in contradiction with some old thoughts about the high computational burden of these methods and their lack of scalability.

Although this survey has focused on pure vehicle routing problems, EAs are also used to address these problems in relation with other logistics problems. For example, the work in [138] describes a GA for a location-routing problem, where depots for the vehicles are located in a transportation network in addition to the optimization of their routes. EAs are thus alive and well, and used to address an ever widening range of vehicle routing applications.

Acknowledgments

Financial support for this work was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC). This support is gratefully acknowledged.

References

[1] Alba, E., B. Dorronsoro. 2004. Solving the vehicle routing problem by using cellular genetic algorithms. J. Gottlieb, G.R. Raidl, eds. *Lecture Notes in Computer Science 3004*. Springer, Berlin. 11–20.

[2] Alba, E., B. Dorronsoro. 2006. Computing nine new best-so-far solutions for capacitated VRP with a cellular genetic algorithm. *Information Processing Letters* **98** 225–230.

[3] Alvarenga, G.B., R.M. de Abreu Silva, R.M. Sampaio. 2005. A hybrid algorithm for the vehicle routing problem with time window. *INFOCOMP Journal of Computer Science* **4** 9–16.

[4] Alvarenga, G.B., G.R. Mateus. 2004. A two-phase genetic and set partitioning approach for the vehicle routing problem with time windows. M. Ishikawa et al., eds. *Pro-*

ceedings of the Fourth International Conference on Hybrid Intelligent Systems. IEEE Computer Society Press, Los Alamitos, CA. 428–433.

[5] Alvarenga, G.B., G.R. Mateus. 2004. Hierarchical tournament selection genetic algorithm for the vehicle routing problem with time windows. M. Ishikawa et al., eds. *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems*. IEEE Computer Society Press, Los Alamitos, CA. 410–415.

[6] Alvarenga, G.B., G.R. Mateus, G. de Tomi. 2007. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research* **34** 1561–1584.

[7] Augerat, P., J.M. Belenguer, E. Benavent, A. Corberán, D. Naddef, G. Rinaldi. 1995. Computational results with a branch and cut code for the capacitated vehicle routing problem. Research Report No. 949-M, Université Joseph Fourier, Grenoble, France.

[8] Baker, B.M., M.A. Ayechew. 2003. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research* **30** 787–800.

[9] Bean, J.C. 1994. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing* **6** 154–160.

[10] Bent, R., P. Van Hentenryck. 2004. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science* **38** 515–530.

[11] Benyahia, I., J.-Y. Potvin. 1995. Generalization and refinement of route construction heuristics using genetic algorithms. *IEEE International Conference on Evolutionary Computation, Perth, Australia*, IEEE Press, Piscataway, NJ. I–39–43.

[12] Benyahia, I., J.-Y. Potvin. 1998. Decision support for vehicle dispatching using genetic programming. *IEEE Transactions on Systems, Man and Cybernetics* **28** 306–314.

[13] Berger, J., M. Barkaoui. 2003. A hybrid genetic algorithm for the capacitated vehicle routing problem. E. Cantú-Paz et al., eds. *Proceedings of the Genetic and Evolutionary Computation Conference. Lecture Notes in Computer Science 2723*. Springer, Berlin. 646–656.

[14] Berger, J., M. Barkaoui. 2003. A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society* **54** 1254–1262.

[15] Berger, J., M. Barkaoui. 2004. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research* **31** 2037–2053.

[16] Berger J., M. Barkaoui, O. Bräysy. 2003. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *INFOR* **41** 179–194.

[17] Berger, J., M. Salois, R. Bégin. 1998. A hybrid genetic algorithm for the vehicle routing problem with time windows. R.E. Mercer, E. Neufeld, eds. *Lecture Notes in Computer Science 1418*. Springer, London, UK. 114–127.

[18] Berger, J., M. Sassi, M. Salois. 1999. A hybrid genetic algorithm for the vehicle routing problem with time windows and itinerary constraints. W. Banhaf et al., eds. *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, San Francisco, CA. 44–51.

[19] Beyer, H.-G., H.-P. Schwefel. 2002. Evolution strategies: a comprehensive introduction. *Natural Computing* **1** 3–52.

[20] Blanton, J.L., R.L. Wainwright. 1993. Multiple vehicle routing with time and capacity constraints using genetic algorithms. S. Forrest, ed. *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA. 451–459.

[21] Blum, C., A. Roli. 2003. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys* **35** 268–308.

[22] Brandao de Oliveira, H.C., J.L. Alexandrino, M. Moreira de Souza. 2006. Memetic and genetic algorithms: a comparison among different approaches to solve vehicle routing problem with time windows. *International Conference on Hybrid Intelligent Systems, Auckland, New Zealand*. IEEE Computer Society Press, Los Alamitos, CA. 55.

[23] Bräysy, O. 2003. A reactive variable neighborhood search for the vehicle routing problem with time windows. *INFORMS Journal on Computing* **15** 347–368.

[24] Bräysy, O., J. Berger, M. Barkaoui. 2000. A new hybrid evolutionary algorithm for the vehicle routing problem with time windows. *Route 2000 Workshop, Skodsborg, Denmark*.

[25] Bräysy, O., W. Dullaert. 2003. A fast evolutionary metaheuristic for the vehicle routing problem with time windows. *International Journal on Artificial Intelligence Tools* **12** 153–172.

[26] Bräysy, O., W. Dullaert, M. Gendreau. 2004. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics* **10** 587–611.

[27] Bräysy, O., M. Gendreau. 2005. Vehicle routing problem with time windows, part II: metaheuristics. *Transportation Science* **39** 119–139.

[28] Bräysy, O., G. Hasle, W. Dullaert. 2004. A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research* **159** 586–605.

[29] Calégari, P., G. Coray, A. Hertz, D. Kobler, P. Kluonen. 1999. A taxonomy of evolutionary algorithms in combinatorial optimization. *Journal of Heuristics* **5** 145–158.

[30] Chang, Y., L. Chen. 2007. Solve the vehicle routing problem with time windows via a genetic algorithm. *Discrete and Continuous Dynamical Systems, Supplement 2007*. 240–249.

[31] Chen, A.-L., G.K. Yang, Z.M. Wu. 2006. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University SCIENCE A* **7** 607–614.

[32] Chin, A.J., H.W. Kit, A. Lim. 1999. A new GA approach for the vehicle routing problem. *IEEE International Conference on Tools with Artificial Intelligence, Chicago, IL*. IEEE Press, Piscataway, NJ. 307–310.

[33] Christofides, N., A. Mingozzi, P. Toth. 1979. The vehicle routing problem. N. Christofides, A. Mingozzi, P. Toth, C. Sandi, eds. *Combinatorial Optimization*. Wiley, Chichester, UK. 315–338.

[34] Clarke, G., J. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* **12** 568–581.

[35] Deb, K., S. Agrawa, A. Pratap, T. Meyarivan. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. M. Schoenauer et al., eds. *Lecture Notes in Computer Science 1917*. Springer, Berlin. 849–858.

[36] Derigs, U., R. Kaiser. 2007. Applying the attribute based hill climber heuristic to the vehicle routing problem. *European Journal of Operational Research* **177** 719-732.

[37] Drummond, L.M.A., L.S. Ochi, D.S. Vianna. 2001. An asynchronous parallel metaheuristic for the period vehicle routing problem. *Future Generation Computer Systems* **17** 379–386.

[38] Dueck, G. 1993. New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics* **104** 86-92.

[39] Dueck, G., T. Scheurer. 1990. Threshold accepting: a general purpose optimization algorithm. *Journal of Computational Physics* **90** 161-175.

[40] Feillet, D., P. Dejax, M. Gendreau. 2005. Traveling salesman problem with profits. *Transportation Science* **39** 188-205.

[41] Filipec, M., D. Skrlec, S. Krajcar. 1997. Darwin meets computers: new approach to multiple depot capacitated vehicle routing problem. *IEEE International Conference on Systems, Man and Cybernetics, Orlando, FL*. IEEE Press, Piscataway, NJ. I–421–426.

[42] Filipec, M., D. Skrlec, S. Krajcar. 1998. An efficient implementation of genetic algorithms for constrained vehicle routing problem. *IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA*. IEEE Press, Piscataway, NJ. III–2231–2236.

[43] Filipec, M., D. Skrlec, S. Krajcar. 2000. Genetic algorithm approach for multiple depot capacitated vehicle routing problem solving with heuristic improvements. *International Journal of Modelling & Simulation* **20** 320–328.

[44] Fisher M.L. 1994. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research* **42** 626-642.

[45] Fox B.R., M.B. McMahon. 1991. Genetic operators for sequencing problems. G.J.E. Rawlins, ed. *Foundations of Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA. 284–300.

[46] Ganesh, K., T.T. Narendran. 2007. CLOVES: a cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. *European Journal of Operational Research* **178** 699–717.

[47] Gehring, H., J. Homberger. 1999. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. K. Miettinen, M.M. Mäkelä, J. Toivanen, eds. *Proceedings of EUROGEN99 - Short Course on Evolutionary Algorithms in Engineering and Computer Science*. Reports of the Department of Mathematical Information Technology, No. A 2/1999, University of Jyväskylä, Finland. 57–64.

[48] Gehring, H., J. Homberger. 2001. A parallel two-phase metaheuristic for routing problems with time windows. *Asia-Pacific Journal of Operational Research* **18** 35–47.

[49] Gehring, H., J. Homberger. 2002. Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of Heuristics* **8** 251–276.

[50] Gendreau, M., G. Laporte, J.-Y. Potvin. 1997. Vehicle routing: modern heuristics. J.K. Lenstra, E.H.L. Aarts, eds. *Local Search in Combinatorial Optimization*. Wiley, Chichester, UK. 311–336.

[51] Gendreau, M., G. Laporte, J.-Y. Potvin. 2002. Metaheuristics for the capacitated VRP. P. Toth, D. Vigo, eds. *The Vehicle Routing Problem*. SIAM Series on Discrete Mathematics and Applications, Philadelphia, PA. 129–154.

[52] Gendreau, M., G. Laporte, J.-Y. Potvin, F. Semet. 2000. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research* **7** 285–300.

[53] Gendreau, M., G. Laporte, J.-Y. Potvin, F. Semet. 2003. Métaheuristiques pour le problème des tournées de véhicules. M. Pirlot, J. Teghem, eds. *Résolution de Problèmes de RO par les Métaheuristiques*. Hermès, Paris, France. 49–70.

[54] Gendreau, M., J.-Y. Potvin. 2005. Metaheuristics in combinatorial optimization. *Annals of Operations Research* **140** 189–213.

[55] Gendreau, M., J.-Y. Potvin, O. Bräysy, G. Hasle, A. Løkketangen. 2008. Metaheuristics for the vehicle routing problem and extensions: a categorized bibliography. B.L. Golden, S. Raghavan, E. Wasil, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges*, Springer, New York, NY. 143–170.

[56] Glover, F. 1996. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics* **65** 223–253.

[57] Glover, F., M. Laguna. 1997. *Tabu Search*. Kluwer, Boston, MA.

[58] Goldberg, D.E., B. Korb, K. Deb. 1989. Messy genetic algorithms: motivation, analysis and first results. *Complex Systems* **3** 493–530.

[59] Goldberg, D.E., J. Richardson. 1987. Genetic algorithms with sharing for multimodal function optimization. J.J. Grefenstette, ed. *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum Associates, Hillsdale, NJ. 41–49.

[60] Goldberg, D.E., J. Richardson. 1988. Alleles, loci, and the traveling salesman problem. J.J. Grefenstette, ed. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum Associates, Hillsdale, NJ. 154–159.

[61] Golden, B.L, E.A. Wasil, J.P. Kelly, I.-M. Chao. 1998. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets and computational results. T.G. Crainic, G. Laporte, eds. *Fleet Management and Logistics*. Kluwer, Norwell, MA. 33–56.

[62] Haghani, A., S. Jung. 2005. A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research* **32** 2959–2986.

[63] Han, S. 2004. A hybrid meta-heuristic algorithm for vehicle scheduling problem: genetic algorithm and tabu search. *NUCB Journal of Economics and Information Science* **48** 343–358.

[64] Han, S., Y. Tabata. 2002. A hybrid genetic algorithm for the vehicle routing problem with controlling lethal gene. *International Journal of Asian Pacific Management Review* **7** 287–298.

[65] Hanshar, F.T., B.M. Ombuki-Berman. 2007. Dynamic vehicle routing using genetic algorithms. *Applied Intelligence* **27** 89–99.

[66] Ho, W.-K., J. Chin, A. Lim. 2001. A hybrid search algorithm for the vehicle routing problem with time windows. *International Journal on Artificial Intelligence Tools* **10** 431–449.

[67] Holland, J.H. 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI (reprinted in 1992, The MIT Press, Cambridge, MA).

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

[69] Homberger, J., H. Gehring. 2005. A two-phase hybrid metaheuristics for the vehicle routing problem with time windows. *European Journal of Operational Research* **162** 220–238.

[70] Housroum, H., T. Hsu, R. Dupas, G. Goncalves. 2006. A hybrid GA approach for solving the dynamic vehicle routing problem with time windows. *IEEE International Conference on Information & Communication Technologies, Damascus, Syria*. IEEE Press, Piscataway, NJ. I–787–792.

[71] Hwang, H.-S. 2002. An improved model for vehicle routing with time constraint based on a genetic algorithm. *Computers & Industrial Engineering* **42** 361–369.

[72] Ibaraki, T., S. Imahori, M. Kubo, T. Masuda, T. Uno, M. Yaigura. 2005. Effective local search algorithms for routing and scheduling problems with general time window constraints. *Transportation Science* **39** 206–232.

[73] Jaszkiewicz, A., P. Kominek. 2003. Genetic local search with distance preserving recombination operator for a vehicle routing problem. *European Journal of Operational Research* **151** 352–364.

[74] Jih, W.-R., J. Yung-Jen Hsu. 1999. Dynamic vehicle routing using hybrid genetic algorithms. *IEEE International Conference on Robotics and Automation, Detroit, MI*. IEEE Press, Piscataway, NJ. I–453–458.

[75] Jones, D.R., M.A. Beltramo. 1991. Solving partitioning problems with genetic algorithms. R.K. Belew, L.B. Booker, eds. *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA. 442–449.

[76] Jorgensen, R.M., J. Larsen, K.B. Bergvinsdottir. 2007. Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society* **58** 1321–1331.

[77] Jozefowiez, N., F. Semet, E.G. Talbi. 2002. Parallel and hybrid models for multi-objective optimization: application to the vehicle routing problem. J. Guervós et al., eds. *Lecture Notes in Computer Science 2439*. Springer, Berlin. 271–280.

[78] Jozefowiez, N., F. Semet, E.G. Talbi. 2006. Enhancements of NSGA-II and its application to the vehicle routing problem with route balancing. E.G. Talbi et al., eds. *Lecture Notes in Computer Science 3871*. Springer, Berlin. 131–142.

[79] Jung, S., A. Haghani. 2000. Genetic algorithm for a pickup and delivery problem with time windows. *Transportation Research Record* **1733** 1–7.

[80] Jung, S., A. Haghani. 2001. Genetic algorithm for the time-dependent vehicle routing problem. *Transportation Research Record* **1771** 164–171.

[81] Jung, S., B.-R. Moon. 2000. The natural crossover for the 2D Euclidean TSP. D. Whitley et al., eds. *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, San Francisco, CA. 1003–1010.

[82] Jung, S., B.-R. Moon. 2002. A hybrid genetic algorithm for the vehicle routing problem with time windows. W.B. Langdon et al., eds. *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, San Francisco, CA. 1309–1316.

[83] Kennedy, J., R.C. Eberhart. 2001. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA.

[84] Kirkpatrick, S., C.D. Gelatt Jr., M.P. Vecchi. 1983. Optimization by simulated annealing. *Science* **220** 671–680.

[85] Koza, J.R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, MA.

[86] Krajcar, S., D. Skrlec, B. Pribicevic, S. Blagajac. 1995. GA approach to solving multiple vehicle routing problem. C.A. Pinto-Ferreira, N.J. Mamede, eds. *Lecture Notes in Computer Science 990*. Springer, Berlin. 473–481.

[87] Kubiak, M. 2004. Systematic construction of recombination operators for the vehicle routing problem. *Foundations of Computing and Decision Sciences* **29** 205–226.

[88] Kubiak, M., P. Wesolek. 2007. Accelerating local search in a memetic algorithm for the capacitated vehicle routing problem. C. Cotta, J. van Hemert, eds. *Lecture Notes in Computer Science 4446*. Springer, Berlin. 96–107.

[89] Lau, H.C., Z. Liang. 2001. Pickup and delivery with time windows: algorithms and test cases. *IEEE International Conference on Tools with Artificial Intelligence, Dallas, TX*. IEEE Computer Society, Los Alamitos, CA. 333–340.

[90] Le Bouthillier, A. 2007. Recherches coopératives pour la résolution de problèmes d'optimisation combinatoire. Thèse de Doctorat, Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada.

[91] Le Bouthillier, A., T.G. Crainic. 2005. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers & Operations Research* **32** 1685–1708.

[92] Le Bouthillier, A., T.G. Crainic, P. Kropf. 2005. A guided cooperative search for the vehicle routing problem with time windows. *IEEE Intelligent Systems* **20** 36–42.

[93] Leclerc, F., J.-Y. Potvin. 1997. Genetic algorithms for vehicle dispatching. *International Transactions in Operational Research* **4** 391–400.

[94] Li, F., B.L. Golden, E.A. Wasil. 2005. Very large-scale vehicle routing: new test problems, algorithms and results. *Computers & Operations Research* **32** 1165–1179.

[95] Li, H., A. Lim. 2001. A metaheuristic for solving the pickup and delivery problem with time windows. *IEEE International Conference on Tools with Artificial Intelligence, Dallas, TX*. IEEE Computer Society, Los Alamitos, CA. 160–167.

[96] Lin, S. 1965. Computer solutions of the traveling salesman problem. *Bell Systems Technical Journal* **44** 2245–2269.

[97] Louis, S.J., X. Yin, Z.Y. Yuan. 1999. Multiple vehicle routing with time windows using genetic algorithms. *IEEE Congress on Evolutionary Computation, Washington, DC*. IEEE Press, Piscataway, NJ. 1804–1808.

[98] Machado, P., J. Tavares, F.B. Pereira, E. Costa. 2002. Vehicle routing problem: doing it the evolutionary way. W.B. Langdon et al., eds. *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, San Francisco, CA. 690.

[99] Maeda, O., M. Nakamura, B. Ombuki, K. Onaga. 1999. A genetic algorithm approach to vehicle routing problem with time deadlines in geographical information systems. *IEEE International Conference on Systems, Man and Cybernetics, Tokyo, Japan*. IEEE Press, Piscataway, NJ. 595–600.

[100] Mester, D., O. Bräysy. 2005. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research* **32** 1593–1614.

[101] Mester, D., O. Bräysy. 2007. Active guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research* **34** 2964–2975.

[102] Mester, D., O. Bräysy, W. Dullaert. 2007. A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications* **32** 508–517.

[103] Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs, Third Edition*. Springer, Berlin.

[104] Mladenović, N., P. Hansen. 1997. Variable neighborhood search. *Computers & Operations Research* **24** 1097–1100.

[105] Moin, N.H. 2002. Hybrid genetic algorithms for vehicle routing problems with time windows. *International Journal of the Computer, the Internet and Management* **10** 51–67.

[106] Moscato, P. 1989. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical Report Caltech Concurrent Computation Program 826, California Institute of Technology, Pasadena, CA.

[107] Moscato, P., C. Cotta. 2003. A gentle introduction to memetic algorithms. F. Glover, G.A. Kochenberger, eds. *Handbook of Metaheuristics*. Kluwer, Boston, MA. 105–144.

[108] Mühlenbein, H., M. Gorges-Schleuter, O. Krämer. 1988. Evolution algorithms in combinatorial optimization. *Parallel Computing* **7** 65–85.

[109] Nagata, Y. 2007. Edge assembly crossover for the capacitated vehicle routing problem. C. Cotta, J. van Hemert, eds. *Lecture Notes in Computer Science 4446*. Springer, Berlin. 142–153.

[110] Nagata, Y. 2007. Effective memetic algorithm for the vehicle routing problem with time windows: edge assembly crossover for the VRPTW. *Metaheuristics International Conference, Montreal, Canada* (on CD-ROM).

[111] Nagata Y., S. Kobayashi. 1997. Edge assembly crossover: a high-power genetic algorithm for the traveling salesman problem. T. Bäck, ed. *Proceedings of the Seventh International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, CA. 450–457.

[112] Nanry, W.P., J.W. Barnes. 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research B* **34** 107–121.

[113] Ochi, L.S., D.S. Vianna, L.M.A. Drummond, A.O. Victor. 1998. An evolutionary hybrid metaheuristic for solving the vehicle routing problem with heterogeneous fleet. W. Banzhaf, T.C. Fogarty, R. Poli, M. Schoenauer, eds. *Lecture Notes in Computer Science 1391*. Springer, Berlin. 187–195.

[114] Ochi, L.S., D.S. Vianna, L.M.A. Drummond, A.O. Victor. 1998. A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet. *Future Generation Computer Systems* **14** 285–292.

[115] Oliver, I.M., D.J. Smith, J.R.C. Holland. 1987. A study of permutation crossover operators on the traveling salesman problem. J.J. Grefenstette, ed. *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum Associates, Hillsdale, NJ. 224–230.

[116] Ombuki, B., M. Nakamura, O. Maeda. 2002. A hybrid search based on genetic algorithms and tabu search for vehicle routing. *IASTED International Conference on Artificial Intelligence and Soft Computing, Banff, Canada*, ACTA Press, Calgary, Canada, 176–181.

[117] Ombuki, B., B.J. Ross, F. Hanshar. 2006. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence* **24** 17–30.

[118] Or, I. 1976. Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking. Ph.D. Thesis, Dept. of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

[119] Osman, I.H. 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research* **41** 421–451.

[120] Ozdemir, H.T., C.K. Mohan. 2000. Evolving schedule graphs for the vehicle routing problem with time windows. *IEEE Congress on Evolutionary Computation, La Jolla, CA*. IEEE Press, Piscataway, NJ. II–888–895.

[121] Pankratz, G. 2005. A grouping genetic algorithm for the pick up and delivery problem with time windows. *OR Spectrum* **27** 21–41.

[122] Pankratz, G. 2005. Dynamic vehicle routing by means of a genetic algorithm. *International Journal of Physical Distribution & Logistics* **35** 362–383.

[123] Pedroso, J.P. 1998. Niche search: an application in vehicle routing. *IEEE International Conference on Evolutionary Computation, Anchorage, AK*. IEEE Press, Piscataway, NJ. 177–182.

[124] Pereira, F.B., J. Tavares, P. Machado, E. Costa. 2002. GVR: a new genetic representation for the vehicle routing problem. M. O'Neill, R.F.E. Sutcliffe, C. Ryan, M. Eaton eds. *Lecture Notes in Computer Science 2464*. Springer, Berlin. 95–102.

[125] Pisinger, D., S. Ropke. 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research* **34** 2403–2435.

[126] Potter, T., T. Bossomaier. 1995. Solving vehicle routing problems with genetic algorithms. *IEEE International Conference on Evolutionary Computation, Perth, Australia*. IEEE Press, Piscataway, NJ. II–788–793.

[127] Potvin, J.-Y. 1996. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research* **63** 339–370.

[128] Potvin, J.-Y., S. Bengio. 1996. The vehicle routing problem with time windows, part II: genetic search. *INFORMS Journal on Computing* **8** 165–172.

[129] Potvin, J.-Y., D. Dubé. 1994. Improving a vehicle routing heuristic through genetic search. *IEEE International Conference on Evolutionary Computation, Orlando, FL.* IEEE Press, Piscataway, NJ. 194–199.

[130] Potvin, J.-Y., D. Dubé, C. Robillard. 1996. A hybrid approach to vehicle routing using neural networks and genetic algorithms. *Applied Intelligence* **6** 241–252.

[131] Potvin, J.-Y., C. Duhamel, F. Guertin. 1996. A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence* **6** 345–355.

[132] Potvin, J.-Y., F. Guertin. 1997. Coupling a greedy route construction heuristic with a genetic algorithm for the vehicle routing problem with time windows. R.S. Barr, R.V. Helgason, J.L. Kennington, eds. *Advances in Metaheuristics, Optimization and Stochastic Modeling Technologies*. Kluwer, Boston, MA. 423–442.

[133] Potvin, J.-Y., 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.

[134] Potvin, J.-Y., J.-M. Rousseau. 1995. An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society* **46** 1433–1446.

[135] Potvin, J.-Y., S.R. Thangiah. 1999. Vehicle routing through simulation of natural processes. L.C. Jain, N.M. Martin, eds. *Fusion of Neural Networks, Fuzzy Sets and Genetic Algorithms: Industrial Applications*. CRC Press, Boca Raton, FL. 141–168.

[136] Prescott-Gagnon, E., G. Desaulniers, L.-M. Rousseau. 2007. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Technical Report G-2007-67, GERAD, Montreal, Canada.

[137] Prins, C. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research* **31** 1985–2002.

[138] Prins, C., C. Prodhon, R.W. Calvo. 2006. A memetic algorithm with population management (MA/PM) for the capacitated location-routing problem. J. Gottlieb and G.R. Raidl, eds. *Lecture Notes in Computer Science 3906*. Springer, Berlin. 183–194.

[139] Rahoual, M., B. Kitoun, M.-H. Mabed, V. Bachelet, F. Benameur. 2001. Multicriteria genetic algorithms for the vehicle routing problem with time windows. *Metaheuristics International Conference, Porto, Portugal*. 527–532.

[140] Rochat, Y., É.D. Taillard. 1995. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics* **1** 147–167.

[141] Ropke, S., J.-F. Cordeau, G. Laporte. 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks* **49** 258–272.

[142] Ropke, S., D. Pisinger. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* **40** 455–472.

[143] Russell, R.A. 1995. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation Science* **29** 156–166.

[144] Russell, M.A., G.B. Lamont. 2005. A genetic algorithm for unmanned aerial vehicle routing. H.-G. Beyer et al., eds. *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM Press, New York, NY. 1523–1530.

[145] Salhi, S., S.R. Thangiah, F. Rahman. 1998. A genetic clustering method for the multi-depot vehicle routing problem. G.D. Smith, N.C. Steele, R.F. Albrecht, eds. *Artificial Neural Networks and Genetic Algorithms*. Springer Verlag, Austria. 234–237.

[146] Santos, H.G., L.S. Ochi, E.H. Marinho, L.M.A. Drummond. 2006. Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem. *Neurocomputing* **70** 70–77.

[147] Schmitt, L.J. 1994. An empirical computational study of genetic algorithms to solve order based problems: an emphasis on the TSP and VRPTC. Ph.D. dissertation, Fogelman College of Business and Economics, University of Memphis, TN.

[148] Schönberger, J., H. Kopfer. 2003. A memetic algorithm for a pickup and delivery selection problem. M. Mohammadian, ed. *Proceedings of the International Conference on Computational Intelligence for Modeling, Control and Automation* (on CD-ROM).

[149] Schönberger, J., H. Kopfer. 2003. A combined approach to solve the pickup and delivery selection problem. U. Leopold-Wildburger, F. Rendl, G. Wäscher, eds. *Operations Research Proceedings 2002*. Springer, Berlin. 150–155.

[150] Schönberger, J., H. Kopfer. 2004. Planning the incorporation of logistics service providers to fulfill precedence and time window-constrained transport requests in a most profitable way. B. Fleischmann, A. Klose, eds. *Distribution Logistics: Advanced Solutions to Practical Problems*. Springer, Berlin. 141–158.

[151] Schrimpf, G., J. Schneider, H. Stamm-Wilmbrandt, G. Duek. 2000. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics* **159** 139–171.

[152] Shaw, P. 1998. Using constraint programming and local search methods to solve vehicle routing problems. M. Maher, J.-F. Puget, eds. *Lecture Notes in Computer Science 1520*. Springer, New-York, NY. 417–431.

[153] Skok, M., D. Skrlec, S. Krajcar. 2000. The non-fixed destination multiple depot capacitated vehicle routing problem and genetic algorithms. *International Conference on Information Technology Interfaces, Zagreb, Croatia*. 403–408.

[154] Skok, M., D. Skrlec, S. Krajcar. 2000. The genetic algorithm method for multiple depot capacitated vehicle routing problem solving. *International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies, Brighton, UK*. IEEE Press, Piscataway, NJ. 520–526.

[155] Skrlec, D., M. Filipec, S. Krajcar. 1997. A heuristic modification of genetic algorithm used for solving the single depot capacitated vehicle routing problem. *Intelligent Information Systems, Grand Bahama Island, Bahamas*. IEEE Computer Society, Los Alamitos, CA. 184–188.

[156] Solomon, M.M. 1987. Time window constrained routing and scheduling problems. *Operations Research* **35** 254–265.

[157] Srinivas, N., K. Deb. 1994. Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation Journal* **2** 221–248.

[158] Syswerda, G. 1989. Uniform crossover in genetic algorithms. J.D. Schaffer, ed. *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA. 2–9.

[159] Taillard, É.D. 1993. Parallel iterative search methods for vehicle routing problems. *Networks* **23** 661–673.

[160] Taillard, É.D., P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin. 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science* **31** 170–186.

[161] Tan, K.C., L.H. Lee, K. Ou. 2001. Hybrid genetic algorithms in solving vehicle routing problems with time window constraints. *Asia-Pacific Journal of Operational Research* **18** 121–130.

[162] Tan, K.C., L.H. Lee, K. Ou. 2001. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. *Engineering Applications of Artificial Intelligence* **14** 825–837.

[163] Tan, K.C., T.H. Lee, K. Ou, L.H. Lee. 2001. A messy genetic algorithm for the vehicle routing problem with time window constraints. *IEEE Congress on Evolutionary Computation, Seoul, South Korea*. IEEE Press, Piscataway, NJ. I–679–686.

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

[165] Tan, K.C., T.H. Lee, Y.H. Chew, L.H. Lee. 2003. A multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *IEEE International Conference on Systems, Man and Cybernetics, Washington, DC*. IEEE Press, Piscataway, NJ. I–361–366.

[166] Tan, K.C., T.H. Lee, Y.H. Chew, L.H. Lee. 2003. A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle routing problems. *IEEE Congress on Evolutionary Computation, Canberra, Australia*. IEEE Press, Piscataway, NJ. III–2134–2141.

[167] Tan, K.C., Y.H. Chew, L.H. Lee. 2006. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications* **34** 115–151.

[168] Tan, K.C., Y.H. Chew, L.H. Lee. 2006. A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research* **172** 855–885.

[169] Tarantilis, C.D. 2005. Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research* **32** 2309–2327.

[170] Tavakkoli-Moghaddam, R., A.R. Saremi, M.S. Ziaee. 2006. A memetic algorithm for the vehicle routing problem with backhauls. *Applied Mathematics and Computation* **181** 1049–1060.

[171] Tavares, J., F.B. Pereira, P. Machado, E. Costa. 2002. GVR delivers it on time. *Asia-Pacific Conference on Simulated Evolution and Learning, Singapore*. 745–749.

[172] Tavares, J., F.B. Pereira, P. Machado, E. Costa. 2003. On the influence of GVR in vehicle routing. *ACM Symposium on Applied Computing, Melbourne, FL*. 753–758.

[173] Tavares, J., F.B. Pereira, P. Machado, E. Costa. 2003. Crossover and diversity: a study about GVR. *Analysis and Design of Representations and Operators: A Bird-of-a-Feather Workshop at the 2003 Genetic and Evolutionary Computation Conference, Chicago, IL*. 27–33.

[174] Thangiah, S.R. 1995. An adaptive clustering method using a geometric shape for vehicle routing problems with time windows. L.J. Eshelman, ed. *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, CA. 536–543.

[175] Thangiah, S.R. 1995. Vehicle routing with time windows using genetic algorithms. L. Chambers, ed. *Practical Handbook of Genetic Algorithms: New Frontiers*. CRC Press, Boca Raton, FL. II–253–277.

[176] Thangiah, S.R. 1999. A Hybrid genetic algorithm, simulated annealing and tabu search heuristic for vehicle routing problems with time windows. L. Chambers, ed. *Practical Handbook of Genetic Algorithms: Complex Coding Systems*. CRC Press, Boca Raton, FL. III–347–384.

[177] Thangiah, S.R., A.V. Gubbi. 1993. Effect of genetic sectoring on vehicle routing problems with time windows. *IEEE International Conference on Developing and Managing Intelligent System Projects, Washington, DC*. IEEE Computer Society Press, Los Alamitos, CA. 146–153.

[178] Thangiah, S.R., K.E. Nygard. 1992. School bus routing using genetic algorithms. G. Biswas, ed. *Applications of Artificial Intelligence X: Knowledge-Based Systems*, SPIE, Bellingham, WA. 387–398.

[179] Thangiah, S.R., K.E. Nygard, P.L. Juell. 1991. GIDEON: A genetic algorithm system for vehicle routing with time windows. *IEEE Conference on Artificial Intelligence Applications, Miami, FL*. IEEE Computer Society Press, Los Alamitos, CA. 322–328.

[180] Thangiah, S.R., I.H. Osman, R. Vinayagamoorthy, T. Sun. 1995. Algorithms for the vehicle routing problems with time deadlines. *American Journal of Mathematical and Management Sciences* **13** 323–355.

[181] Thangiah, S.R., P. Petrovic. 1998. Introduction to genetic heuristics and vehicle routing problems with complex constraints. D.L. Woodruff, ed. *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search*. Kluwer, Boston, MA. 253–286.

[182] Thangiah, S.R., S. Salhi. 2001. Genetic clustering: an adaptive heuristic for the multidepot vehicle routing problem. *Applied Artificial Intelligence* **15** 361–383.

[183] Thangiah, S.R., R. Vinayagamoorthy, A.V. Gubbi. 1993. Vehicle routing with time deadlines using genetic and local algorithms. S. Forrest, ed. *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA. 506–513.

[184] Tong, Z., L. Ning, S. Debao. 2004. Genetic algorithm for vehicle routing problem with time window with uncertain vehicle number. *World Congress on Intelligent Control and Automation, Hangzhou, China*. 2846–2849.

[185] Toth, P., D. Vigo. 2003. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing* **15** 333–346.

[186] Van Breedam, A. 1994. An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customer-related and time-related constraints. Ph.D. Dissertation, Faculty of Applied Economics, University of Antwerp, Belgium.

[187] Van Breedam, A. 1996. An analysis of the effect of local improvement operators in genetic algorithm and simulated annealing for the vehicle routing problem. RUCA Working Paper 96/14, University of Antwerp, Belgium.

[188] Vianna, D.S., L.S. Ochi, L.M.A. Drummond. 1999. A parallel evolutionary metaheuristic for the period vehicle routing problem. J.P.D. Rolim et al., eds. *Lecture Notes in Computer Science 1586*. Springer, Berlin. 183–191.

[189] Voudouris, C., E.P.K. Tsang. 1995. Guided local search. Technical Report CSM-247, Department of Computer Science, University of Essex, Colchester, UK.

[190] Voudouris, C., E.P.K. Tsang. 2003. Guided local search. F. Glover, G.A. Kochenberger, eds. *Handbook of Metaheuristics*. Kluwer, Boston, MA. 185–218.

[191] Whitley, D. 1989. The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. J.D. Schaffer, ed. *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA. 116–121.

[192] Yagiura, M., T. Ibaraki. 2001. On metaheuristic algorithms for combinatorial optimization problems. *Systems and Computers in Japan* **32** 33–55.

[193] Zhu, K.Q. 2000. A new genetic algorithm for VRPTW. H.R. Arabnia, ed. *Proceedings of the International Conference on Artificial Intelligence*. CSREA Press, USA. 1–10.

[194] Zhu, K.Q. 2003. A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows. *International Conference on Tools with Artificial Intelligence, Sacramento, CA*. IEEE Computer Society, Los Alamitos, CA. 176–183.

[195] Zhu, Q., L. Qian, Y. Li, S. Zhu. 2006. An improved particle swarm optimization algorithm for vehicle routing problem with time windows. *IEEE Congress on Evolutionary Computation, Vancouver, Canada*. IEEE Press, Piscataway, NJ. 1386–1390.