Chapter 4: Heuristics for the Vehicle Routing Problem

Chapter · November 2014

DOI: 10.1137/1.9781611973594.ch4

3 authors, including:

Stefan Ropke
Technical University of Denmark
74 PUBLICATIONS **8,354** CITATIONS

SEE PROFILE

Thibaut Vidal
Polytechnique Montréal
115 PUBLICATIONS **5,435** CITATIONS

SEE PROFILE

Chapter 4

Heuristics for the Vehicle Routing Problem

Gilbert Laporte
Stefan Ropke
Thibaut Vidal

4.1 ▪ Introduction

In recent years, several sophisticated mathematical programming decomposition algorithms have been put forward for the solution of the VRP. Yet, despite this effort, only relatively small instances involving around 100 customers can be solved optimally, and the variance of computing times is high. However, instances encountered in real-life settings are sometimes large and must be solved quickly within predictable times, which means that efficient heuristics are required in practice. Also, because the exact problem definition varies from one setting to another, it becomes necessary to develop heuristics that are sufficiently flexible to handle a variety of objectives and side constraints. These concerns are clearly reflected in the algorithms developed over the past few years. This chapter provides an overview of heuristics for the VRP, with an emphasis on recent results.

The history of VRP heuristics is as old as the problem itself. In their seminal paper, Dantzig and Ramser [19] sketched a simple heuristic based on successive matchings of vertices through the solution of linear programs and the elimination of fractional solutions by trial and error. The method was illustrated on an eight-vertex graph. It was not pursued, but may have inspired the developers of matching-based heuristics (see Altinkemer and Gavish [3], Desrochers and Verhoog [20], and Wark and Holt [91]). Since then, a wide variety of *constructive* and *improvement heuristics* have been proposed, culminating in recent years with the development of powerful *metaheuristics* capable of computing within a few seconds solutions whose value lies within less than one percent of the best known values.

The field of VRP heuristics is now so rich that it makes no sense to provide an exhaustive compilation of them in a book chapter such as this. Instead, we have decided to focus on methods and principles that have withstood the test of time or present some interesting distinctive features. For a more complete description of the classical heuristics and of the early metaheuristics, we refer the reader to the two chapters by Laporte and

Semet [46] and by Gendreau, Laporte, and Potvin [25] in the first edition of the Toth and Vigo [83] book.

The evolution of VRP heuristics over the past 10 years has taken place, almost exclusively, within the context of metaheuristics. The concept that best encapsulates this evolution is probably that of hybridization. First, we have witnessed the emergence of new heuristics combining several concepts initially developed independently from each other, often related to search principles such as simulated annealing, tabu search, variable neighborhood search, and genetic algorithms. Second, various other strategies, exotic large neighborhoods, exact mathematical techniques, decomposition, and cooperation schemes have found their way into the most successful methods. Third, there has been a hybridization of scope in the sense that researchers are now developing *flexible* methods (e.g., Cordeau, Laporte, and Mercier [15], Vidal et al. [86], and Subramanian, Uchoa, and Ochi [76]) which can be directly applied to the solution of a wide variety of VRP variants without any major structural change.

We will first summarize in Sections 4.2, 4.3, and 4.4 some of the relevant results pertaining to constructive heuristics, improvement heuristics, and metaheuristics for the classical VRP, i.e., the version of the problem containing capacity and route length restrictions only. We will then review and analyze a number of hybridization strategies in Section 4.5, and we will discuss unified algorithms in Section 4.6. Computational comparisons of some of the heuristics surveyed in this chapter are presented Section 4.7. Conclusions follow in Section 4.8.

4.2 ▪ Constructive Heuristics

Constructive heuristics are usually employed to provide a starting solution to an improvement heuristic. However, most metaheuristics are now so robust that they can be initialized from any random solution, feasible or not. Several such heuristics have been proposed over the years and are fully described in the first edition of this book (Toth and Vigo [83]). However, since most of these heuristics have now fallen into disuse, we will concentrate on two classical methods which still present a particular interest.

4.2.1 ▪ The Clarke and Wright Savings Heuristic

The Clarke and Wright heuristic [12] initially constructs back and forth routes $(0, i, 0)$ for $(i = 1, \ldots, n)$ and gradually merges them by applying a saving criterion. More specifically, merging the two routes $(0, \ldots, i, 0)$ and $(0, j, \ldots, 0)$ into a single route $(0, \ldots, i, j, \ldots, 0)$ generates a saving $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. Since the savings remain the same throughout the algorithm, they can be computed a priori. In the so-called parallel version of the algorithm which appears to be the best (see Laporte and Semet [46]), the feasible route merger yielding the largest saving is implemented at each iteration, until no more merger is feasible. This simple algorithm possesses the advantages of being intuitive, easy to implement, and fast. It is often used to generate an initial solution in more sophisticated algorithms. Several enhancements and acceleration procedures have been proposed for this algorithm (see, e.g., Nelson et al. [59] and Paessens [62]), but given the speed of today's computers and the robustness of the latest metaheuristics, these no longer seem justified.

4.2.2 ▪ Petal Algorithms

Petal algorithms consist of generating a set S of feasible VRP routes and combining them through the solution of a set partitioning problem. More specifically, let d_k be the cost of

route k, let a_{ik} be a binary coefficient equal to 1 if and only if customer i belongs to route k, and let x_k be a binary variable equal to 1 if and only if route k belongs to the solution. Then the problem is to

$$\begin{aligned} \text{minimize} & \sum_{k \in S} d_k x_k \\ \text{s.t.} & \sum_{k \in S} a_{ik} x_k = 1 && \forall\, i = 1, \ldots, n, \\ & x_k \in \{0, 1\} && \forall\, k \in S. \end{aligned}$$

The idea of solving the VRP through a set partitioning formulation was first proposed as an exact algorithm by Balinski and Quandt [5], but proved impractical because the number of feasible routes is typically extremely large and computing the costs d_k usually requires the solution of an NP-hard problem. The problem is normally solved heuristically by generating a set of promising routes, as suggested by Foster and Ryan [23] and by Ryan, Hjorring, and Glover [72]. Renaud, Boctor, and Laporte [67] go one step further and also generate configurations called 2-petals, consisting of two embedded or intersecting routes. On the 14 benchmark instances of Christofides, Mingozzi, and Toth [11], these authors quickly obtained solution with costs lying within 2.38% of those of the best known solutions.

Petal algorithms are also particularly well suited for problems containing constraints, such as time windows, other than capacity and route duration constraints. Column generation then becomes a solution methodology of choice, especially for tightly constrained instances.

4.3 ▪ Classical Improvement Heuristics

Classical improvement heuristics perform intra-route and inter-route moves. In the first case, one can apply any improvement heuristic designed for the *Traveling Salesman Problem* (TSP), such as λ-OPT exchanges (Lin [49]), in which λ edges are removed and replaced by λ other edges. In the implementation of Lin and Kernighan [50], the value of λ is modified dynamically throughout the search. According to Johnson and McGeoch [43], who have conducted a detailed empirical analysis of several TSP heuristics, a careful implementation of the Lin and Kernighan algorithm yields the best results. One such implementation was proposed by Helsgaun [36] and is now incorporated in the Concorde TSP solver of Applegate et al. [4].

In practice, inter-route improvement moves are essential to achieve good results. These include classical operators such as removing k consecutive customers from their current route and reinserting them elsewhere (RELOCATE), swapping consecutive customers between different routes (SWAP), or removing two edges from different routes and reconnecting them differently (2-OPT*). These three types of moves are the most commonly used in recent state-of-the-art methods, along with 2-OPT. The number of exchanged customers usually remains small, no more than two in most cases.

A complete exploration of standard neighborhoods requires $O(n^2k^2)$ operations. For large instances, it becomes necessary to reduce the list of considered moves. One such pruning technique, called granular search (see Johnson and McGeoch [42] and Toth and Vigo [84]), considers geographical restrictions to avoid moves between distant customers. Specific move evaluation orders and restrictions, considering the partial sums of length of exchanged edges as in Irnich, Funke, and Grünert [40], can also lead to important search effort reductions.

Most classical moves are special cases of so-called b-cyclic, k-transfer moves (Thompson and Psaraftis [81]) in which a circular permutation of b routes is considered and k customers from each route are shifted to the next route of the cyclic permutation. Inter-route improvement moves can also be viewed as operators within destroy and repair schemes (see Shaw [75]) which alternate between moves that destroy part of the solution and moves that reconstruct it. In the *Adaptive Large Neighborhood Search* (ALNS) heuristic of Pisinger and Ropke [63], these moves are randomly selected by means of a roulette wheel mechanism, yielding excellent results.

4.4 ▪ Metaheuristics

Current metaheuristics for the VRP can broadly be classified into local search methods and population-based heuristics. Local search methods explore the solution space by moving at each iteration from a solution to another solution in its neighborhood. These include well-known schemes such as simulated annealing (see Kirkpatrick, Gelatt, and Vecchi [44] and Nikolaev and Jacobson [60]), deterministic annealing (see Dueck [21], Dueck and Scheuer [22], and Li, Golden, and Wasil [48]), tabu search (see Glover [29] and Gendreau and Potvin [27]), iterated local search (see Baxter [7] and Lourenço, Martin, and Stützle [51]), and variable neighborhood search (see Mladenović and Hansen [54]). Population-based heuristics evolve a population of solutions which may be combined together in the hope of generating better ones. This category includes ant colony optimization (see Reimann, Doerner, and Hartl [66]), genetic algorithms (see Holland [38] and Prins [64]), scatter search, and path relinking (see Glover [28] and Resende et al. [69]).

At the time of the publication of the first edition of this book (Toth and Vigo [83]), these metaheuristics tended to be rather distinct from each other. However, more than 10 years later, the frontiers between them have become more fuzzy and, more importantly, several powerful hybrids of these algorithms have emerged. Therefore, rather than providing a detailed account of every available implementation, a task which is now next to impossible given their number, we will concentrate on a few principles of general applicability. Readers interested in an overview of metaheuristic principles are referred to the handbook edited by Gendreau and Potvin [26].

4.4.1 ▪ Local Search Algorithms

Local search algorithms start from an initial solution x_1 and move at each iteration t from the current solution x_t to another solution x_{t+1} in its neighborhood $N(x_t)$. If $f(x)$ denotes the cost of x, then $f(x_{t+1})$ is not necessarily less than $f(x_t)$. Care must therefore be taken to avoid cycling.

***Simulated Annealing* (SA).** In SA, cycling is prevented by selecting a solution x randomly in $N(x_t)$. If $f(x) \leq f(x_t)$, then $x_{t+1} = x$. Otherwise,

$$x_{t+1} = \begin{cases} x & \text{with probability } p_t, \\ x_t & \text{with probability } 1 - p_t, \end{cases}$$

where p_t is usually a decreasing function of t and of $f(x) - f(x_t)$. It is common to define p_t as

$$p_t = \exp(-[f(x) - f(x_t)]/\theta_t),$$

where the *temperature* θ_t is a decreasing function of t. A well-known application of SA to the VRP is that of Osman [61].

***Deterministic Annealing* (DA).** In DA, the rule for accepting x is deterministic. One of the best implementations of this scheme is the *Record-to-Record Travel* (RRT) algorithm of Li, Golden, and Wasil [48] which is based on the general idea put forward by Dueck [21]. In this algorithm, a *record* is the best known solution x^*. At iteration t, a solution x is drawn from $N(x_t)$ and $x_{t+1} = x$ if $f(x) \leq \sigma f(x^*)$, where σ is a parameter usually slightly larger than 1 (e.g., $\sigma = 1.05$); if $f(x) > \sigma f(x^*)$, then $x_{t+1} = x_t$. The implementations proposed by Groër, Golden, and Wasil [31] and Li, Golden, and Wasil [48] are rather effective while being easy to reproduce.

***Tabu Search* (TS).** TS moves from a solution t to the best non-tabu solution x_{t+1} in $N(x_t)$. In order to avoid cycling, solutions sharing some attributes with x_t are declared *tabu*, or forbidden, for a number of iterations. The tabu status of a potential solution is revoked whenever it corresponds to a new best known solution. Tens of implementations of TS have been proposed in the past 20 years. These usually contain rules to diversify the search or to intensify it in promising regions. One implementation feature stands out among all those that have been proposed. It consists of considering intermediate infeasible solutions during the search, as in Gendreau, Hertz, and Laporte [24]. This is achieved by minimizing a penalized function $f'(x) = f(x) + \sum_k \alpha_k V_k(x)$, where $V_k(x)$ is the total violation of constraint of type k in solution x (for example, $V_k(x)$ could be the total violation of the capacity constraint over all routes), and α_k is a self-adjusting parameter. Of course, $f'(x)$ and $f(x)$ coincide whenever x is feasible. The weights α_k are initially set at 1 and are adjusted throughout the search. At every iteration in the more recent implementations, if x is feasible with respect to constraint k, then α_k is divided by $1+\delta$, where $\delta > 0$; otherwise, α_k is multiplied by $1+\delta$ (see, e.g., Cordeau, Laporte, and Mercier [15]). A recent tabu search algorithm that provides good results is described in Zachariadis and Kiranoudis [92].

***Iterated Local Search* (ILS).** ILS is a local search heuristic whose origins can be traced back to the 1980s (e.g., Baxter [7]) and which has been designated under a variety of names (see Lourenço, Martin, and Stützle [51] for a historical account). The algorithm is simple and can be applied on top of any local search procedure, be it a simple one like steepest descent based on a single neighborhood or a more complicated one like TS. The idea is to apply an embedded local search mechanism until it reaches a stopping criterion, perturb (or shake) the solution returned by the embedded local search to yield a new starting solution, and then reapply the embedded local search. This continues until a stopping criterion is met (e.g., a number of outer iterations, a time limit, or a number of consecutive iterations without solution improvement). The perturbation operation is application-specific and must be designed with care, so that the perturbation is not easily undone by the embedded local search, but it should not completely destroy the structure of the original solution. Chen, Huang, and Dong [10] have designed an iterated variable neighborhood search descent heuristic for the *Capacitated VRP* (CVRP), which combines several operators and a perturbation strategy based on an exchange of a few consecutive customers. Subramanian, Uchoa, and Ochi [76] have later designed a hybrid metaheuristic combining the design of vehicle routes by means of an ILS heuristic and the solution of a set partitioning problem. Shaking is performed by means of combined exchanges of customers.

***Variable Neighborhood Search* (VNS).** VNS was proposed by Mladenović and Hansen [54] as a general search strategy. It works with several neighborhoods $N_1, \ldots, N_p$ which are often of increasing complexity and can even be embedded, for example, 2-OPT, 3-OPT, etc. Starting with an initial solution the method iteratively applies these neighborhoods in a descent fashion until no further improvement is possible. After the last neighborhood has been applied, a new cycle can be restarted. The algorithm stops after a preset number of cycles or when no further improvement is possible. Several variants of this basic mechanism have been proposed, and the method has been applied to a wide variety of contexts (see Hansen et al. [33] for a survey). VNS was successfully applied to the VRP by Kytöjoki et al. [45]. This search scheme was used to guide the application of several VRP improvement procedures. The method was designed with the solution of large-scale real-life instances in mind. It was capable of quickly identifying high quality solutions on such instances involving up to 20,000 customers.

4.4.2 ▪ Population-Based Algorithms

Whereas the previously described local search algorithms have been inspired by the necessity of escaping from local optima and avoiding cycling, population-based methods take their inspiration from natural concepts, e.g., the evolution of species and the behavior of social insects foraging. These methods implement a high-level guidance strategy based on different memory structures, such as neural networks, pools of solutions represented as chromosomes, or pheromone matrices. Furthermore, all known successful VRP heuristics of this type also rely on local search components to drive the search towards promising solutions. Most of the population-based methods in the VRP literature are thus inherently hybrid.

***Ant Colony Optimization* (ACO).** The ACO algorithm of Reimann, Doerner, and Hartl [66] has been one of the most successful. New solutions are generated by means of a savings-based procedure and local search. Instead of using the standard saving definition $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ of the Clarke and Wright algorithm [12], the authors use an attractiveness value $\chi_{ij} = \tau_{ij}^{\alpha} - s_{ij}^{\beta}$, where the *pheromone value* τ_{ij} measures how good combining i and j turned out to be in previous iterations, and α, β are user-controlled parameters. The combination of vertices i and j takes place with probability $p_{ij} = \chi_{ij}/(\sum_{(h,\ell)\in\Omega_k} \chi_{h\ell})$, where Ω_k is the set of the feasible (i, j) combinations yielding the k best savings.

***Genetic Algorithms* (GA).** The first successful application of GAs (see Holland [38]) to the VRP is due to Prins [64]. The method combines genetic operators, selection, and crossover with an efficient local search which replaces the classic randomized mutation operator. This type of hybrid method is sometimes called memetic algorithm (see Moscato and Cotta [55]). Prins [64] represents a solution as a giant tour without trip delimiters. An optimal shortest path-based procedure (see [8]), called SPLIT, is used for inserting visits to the depots and delimiting the routes. Genetic operators are applied on giant-tour solutions, thus allowing for simple permutation-based crossover operations, while local search procedures are applied on the complete solution representation after SPLIT. Diversity management procedures are exploited to control the population. A new solution s_{NEW} with objective value $z(s_{\text{NEW}})$ is accepted in the population $\mathscr{P}$ if and only if there exists no solution $s \in \mathscr{P}$ such that $|z(s_{\text{NEW}}) - z(s)| \leq \Delta$, where Δ is an objective-spacing coefficient.

Other efficient GAs for the CVRP have later been proposed. The heuristic of Nagata and Bräysy [57] relies on an adaptation of the *Edge-Assembly Crossover* (EAX), which has been successful on the TSP (see Nagata and Kobayashi [58]). This crossover considers the graph associated with the merger of the edges from two parents and selects several cycles in the graph, alternating between edges of parents P1 and P2. A new offspring is then created by removing the selected edges of P1 and replacing them with those of P2. The resulting graph is not always a legit VRP solution, since it may contain cycles that are not connected to the depot or routes exceeding the vehicle capacity. Thus, a repair procedure is applied, including a greedy heuristic to merge disconnected cycles and routes and an efficient local search to reduce capacity infeasibility. Combining this crossover with well-designed local search procedures, including efficient neighborhood pruning methods, proved highly successful.

The hybrid GA of Vidal et al. [85] "with advanced diversity control" builds upon the solution representation of Prins [64], and also integrates a bicriteria evaluation of individuals, driven by solution quality as well as their contribution to the population diversity. The fitness $\phi_{\mathscr{P}}(S)$ of a solution S in population $\mathscr{P}$ is a weighted sum of its rank in the population in terms of contribution to the population's diversity $\phi^{\mathrm{DIV}}(S)$, evaluated as a Hamming distance to the other solutions, and its rank with respect to the solution cost $\phi^{\mathrm{COST}}(S)$. The parameter μ^{ELITE} governs the relative weight of each criterion. This fitness measure is used to select both the parents and the survivors:

$$\phi_{\mathscr{P}}(S) = \phi^{\mathrm{COST}}(S) + \left(1 - \frac{\mu^{\mathrm{ELITE}}}{|\mathscr{P}|}\right)\phi^{\mathrm{DIV}}(S). \tag{4.1}$$

This efficient diversity management system allows the method to rely on an efficient granular local search, applied on all offspring, without risking a premature population convergence. Finally, penalized infeasible solutions are exploited during the solution process to further enhance the method exploration capabilities.

***Scatter Search* (SS) and *Path Relinking* (PR).** Solution recombinations and local improvement are very different and complementary methods. The former allows the algorithm to select and recombine large parts of high-quality solutions, while the latter is used for solution refinement. This may explain the success of recent hybrid GAs. Yet, in many current methods, the crossover operator often appends together sequences of visits from different solutions in a blind and randomized manner. As such, a solution immediately obtained after crossover may be of very low quality. Whereas some randomization and shaking is beneficial at the exploration level, too much of it can be detrimental. Thus, some research has focused on exploiting more purposeful and intelligent recombinations, e.g., in the context of SS or PR (see Glover [28] and Resende et al. [69]). The latter explores in a finer manner the intermediate solutions located on a path between an origin solution and a target solution. The recent PR algorithm of Tarantilis, Anagnostopoulou, and Repoussis [80] has proved very successful on the VRP with time windows.

Learning Mechanisms. Early implementations of learning mechanisms to the VRP were based on neural networks. Some of these methods are described in Gendreau, Laporte, and Potvin [25]. However, their overall success on CVRP benchmark instances remains mitigated, and these methods are now mostly investigated in dynamic contexts (see Créput et al. [18]).

4.5 ▪ Hybridizations

Classifying the current state-of-the-art methods for the CVRP would have been easier a decade ago. However, as research progresses, we witness the emergence of a wider variety of hybrids methods which rely on concepts borrowed from various algorithmic paradigms such as local search, large neighborhoods, collective intelligence and population-based search, perturbations, integer programming, constraint programming, tree search, data mining, and parallel computing. The frontier between methods is thus becoming increasingly blurred. Some important families of hybridizations stand out and are now discussed.

Population-Based Search and Local Search. Population-based heuristics have been long complemented by local search, but the inverse process, i.e., complementing a local search method such as TS by population and recombination concepts, has also been considered and is sometimes called *Adaptive Memory Programming* (AMP) (see Taillard et al. [78]). During the search, recurrent fragments of visit sequences from elite solutions, e.g., local minima, are stored in an *adaptive memory*. Subsequences of visits from these routes are selected to create new solutions used as starting points. The subsequence selection process is driven by the number of occurrences in elite routes, as by some minimum length requirements. This technique has led to several successful heuristics for the CVRP (see Rochat and Taillard [70] and Tarantilis [79]). Extracting and using promising sequences of solutions is a complex and rewarding task, which is sometimes called *guidance* (see Le Bouthillier, Crainic, and Kropf [47]), and may even be performed by means of data mining techniques (see Santos et al. [73]).

Meta-Meta Hybridizations. A very common hybridization scheme is to combine several concepts borrowed from different metaheuristics. In some cases, the concepts and methods may be indissociable from each other and occur in a single phase, while otherwise diverse techniques can be applied, sequentially or concurrently, to the same or different starting points. The heuristic of Kytöjoki et al. [45] combines VNS with a technique known as *Guided Local Search* (GLS), in which some solution features are temporarily penalized in the objective in order to escape from a local optimum (see Voudouris and Tsang [89]). Performing restarts from different initial solutions is a mechanism generally associated with GRASP (see Resende and Ribeiro [68]). Yet, many other metaheuristics also take advantage of this strategy (e.g., Cordeau and Maischberger [16], Prins [65], and Subramanian, Uchoa, and Ochi [76]), and so do implicitly all authors who consider the best solution out of several runs when reporting results. In addition to restarts, [65] performs several randomized local searches on each incumbent solution of an ILS, leading to a population of solutions at each iteration, out of which the best new incumbent is extracted. Finally, [16] combines concepts of ILS with a perturbation performed as a ruin-and-recreate move on a cluster of customers, a tabu memory, and a GLS objective function within a parallel context and restarts from exchanged solutions.

Hybridizations with Large Neighborhoods. The variety of neighborhoods is the essence of VNS. It may arise as a result of a parametrization of classic neighborhoods, e.g., by relocating or exchanging sequences of visits of increasing size as in Hemmelmayr, Doerner, and Hartl [37]. One possible avenue of improvement is to rely on structurally different neighborhoods, such as the SWAP or RELOCATE operators, along with large neighborhoods based on ruin-and-recreate moves or ejection chains. The ALNS algorithm of Pisinger and Ropke [63] is a good example. Structurally different large neighborhoods are used and selected through a roulette wheel mechanism. The selection rate is adapted to the performance of the operator at hand. Mester and Bräysy [53] use a

complex hybridization scheme, involving GLS, large neighborhoods based on ejection chains, decomposition phases, and $1+1$ evolution strategies where a single offspring replaces a single parent in case of improvement. The rationale behind metaheuristic hybridizations stems from the fact that each method leads to a different distribution of final solutions. A strongly attractive local optimum for a method, e.g., TS, may be easier to improve with a different technique, e.g., hybrid GA, and vice versa. Hybrids are therefore inspired by the same strategy as VNS, i.e., alternating between neighborhoods to further improve the solutions, albeit at a higher level.

Hybridizations with Mathematical Programming Solvers. Other forms of hybridizations combine metaheuristics with mathematical programming solvers or other exact algorithms. These methods are called *matheuristics* (see Maniezzo, Stützle, and Voss [52]). One successful strategy requires storing high-quality routes in a pool and applying an integer programming solver to a set covering formulation. The resulting method belongs to the family of petal algorithms (see Foster and Ryan [23] and Renaud, Boctor, and Laporte [67]), but it is more sophisticated than the basic scheme since it involves some cooperation between the MIP solver and the route generation heuristic. Some matheuristics of this kind are known to be particularly efficient for VRP instances containing a large number of small routes (see Groër, Golden, and Wasil [32], Muter, Birbil, and Sahin [56], and Subramanian, Uchoa, and Ochi [76]). The synergy between the exact method and the metaheuristic may also be enhanced by promptly communicating new upper bounds and using them as cut-offs, as well as applying a reactive mechanism that interactively controls the size of the set partitioning models (see Groër, Golden, and Wasil [32] and Subramanian, Uchoa, and Ochi [76]). A number of other matheuristics have been proposed for the CVRP. The use of constraint or integer programming algorithms for solution reconstructions, advocated by Shaw [75], is now infrequent for the CVRP. Furthermore, Toth and Tramontani [82] introduced a large-neighborhood improvement procedure derived from the TSP neighborhood of Sarvanov and Doroshko [74]. As illustrated by Figure 4.1, the method relies on a reassignment model to relocate at most $n/2$ groups of customers (white-filled circles) between fixed positions (gray-filled circles). Ahuja et al. [1] survey other very large neighborhoods based on linear or dynamic programming which may be combined with CVRP metaheuristics.

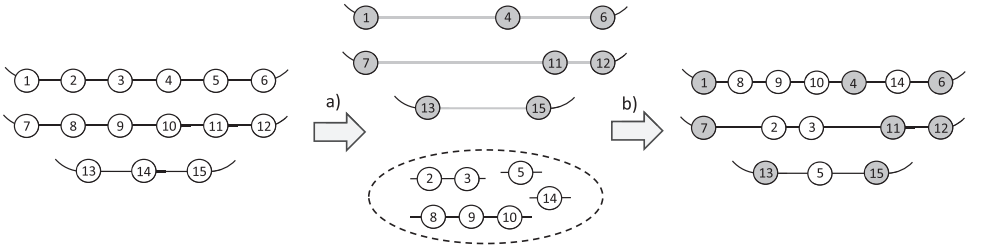

Figure 4.1. *Large Neighborhood of Toth and Tramontani* [82]*:* (a) *some fixed customers are selected, and the edges are short-circuited;* (b) *customer groups located between fixed points are optimally reassigned using integer programming.*

Parallel Algorithms. Parallel and cooperative search mechanisms have also led to well-performing hybrid heuristics. Most recent approaches consider a high-level parallelism (see Crainic and Toulouse [17]) in which different solutions and search trajectories are explored in different threads. At any point in time, the overall algorithm possesses a population of solutions, which can be exchanged, analyzed, and stored. Parallel methods

therefore conveniently combine the exploration capacities of population-based search and the fast improvement mechanisms of local search. Homogeneous solvers are sometimes used. For example, Cordeau and Maischberger [16] combine several ILS with well-designed exchange mechanisms. Yet, most successful methods benefit from heterogeneous collaborating solvers, e.g., GA and TS cooperation in Le Bouthillier, Crainic, and Kropf [47], shaking and set covering solvers in Groër, Golden, and Wasil [32], or multiple TS with different neighborhood structures in Jin, Crainic, and Løkketangen [41].

Decompositions or Coarsening Phases. Several state-of-the-art metaheuristics are complemented by decomposition or coarsening techniques, thus allowing an efficient handling of larger-scale instances such as those of Golden et al. [30] and Li, Golden, and Wasil [48]. A common way to proceed is to rely on the customer-to-route assignments of an existing solution and combine subsets of routes and their associated customers to create subproblems. Different methods can be used to define the groups, e.g., (a) randomly, (b) relatively to a proximity criterion between routes such as the barycentric distance, (c) by increasing polar angle around the depot as in Vidal et al. [86], or (d) by dividing the geometrical space into sections, as in Groër, Golden, and Wasil [32]. These alternatives are depicted in Figure 4.2. Such decompositions, based on one elite solution, can

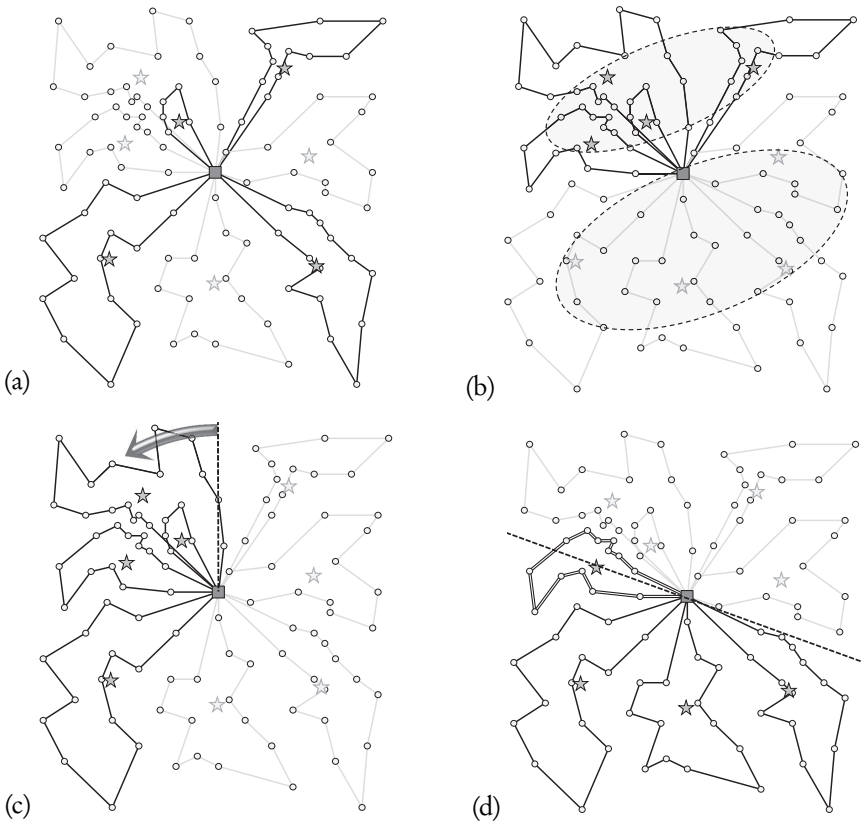

Figure 4.2. *Decomposition strategies on instance P03 of Christofides, Mingozzi, and Toth* [11] *with* 100 *customers. Route barycenters are represented by stars. Strategy* (a) *is a random selection of routes,* (b) *relies on a clustering of barycenters,* (c) *performs a circular sweeping of barycenters, and* (d) *divides the space into two sections (any route being part of two sections, e.g., the route displayed with double lines, being considered in both subproblems).*

be assimilated to projection methods, which fix some assignment decisions to obtain a separable problem. Another instance size reduction technique involves the identification of recurrent edges in good solutions, which can then be temporarily fixed by "merging" customers, as in multi-level heuristics, in order to focus on the remaining decision variables (see Walshaw [90]). It should be noted that, in this case, the reduced problem is asymmetric.

Diversification vs. Intensification. The balance between diversification, i.e., exploring unknown solution features, and intensification, i.e., focusing the search around known good solutions, has been the subject of considerable attention since the emergence of metaheuristics. Most hybrid heuristics for the CVRP seek to achieve a fine balance between these two search strategies. To illustrate, Table 4.1 summarizes some key intensification and diversification techniques. Numerous methodological elements and strategies have thus been suggested over the past years. In addition to finding new promising concepts, one main challenge of current research on metaheuristics is to better assess the scope of the application and the impact of some of these separate elements and combine them effectively.

4.6 ▪ Unified Algorithms

Recently, we have witnessed the emergence of a large number of new vehicle routing variants with additional constraints, objectives, and decision variables, called *attributes* in Vidal et al. [87]. Examples of recent attributes include fuel consumption optimization, turn penalties, time-dependent or flexible travel times, multiple compartments, and consistency. A very large number of metaheuristics dedicated to some attribute combinations have been proposed in the past years, for example for the solution of multi-period and multi-depot VRPs, open heterogeneous VRPs, or vehicle fleet mix problems with time windows. However, given the almost unlimited number of hybrid problems that can be formulated, it is becoming increasingly important to move from problem-specific methods to more flexible approaches which have the potential of being applied to a wider range of settings. This also leads to the issue of assessing the scope and the degree of generality of algorithm components.

A number of algorithms are sufficiently flexible to produce high-quality solutions on several multi-attribute VRPs through a single implementation. Thus, the *Unified Tabu Search* (UTS) heuristic of Cordeau, Laporte, and Mercier [15] can solve classical VRPs as well as *Pickup-and-Delivery Problems* (PDPs) with any combination of multiple depots, multiple planning periods, duration constraints, and time windows. The method was extended by Cordeau and Maischberger [16] into a parallel iterated TS heuristic. Similarly, the ALNS algorithm of Pisinger and Ropke [63] can efficiently solve VRPs and PDPs with multiple depots, vehicle-customer compatibility constraints, backhauls, and time windows. As a matter of fact, ALNS can be viewed as a metaheuristic framework in itself. It has been applied to a host of other multi-attribute VRPs, but often with separate implementations. A rich family of problems with time attributes was solved in Hashimoto et al. [34], Ibaraki et al. [39], and Hashimoto, Yagiura, and Ibaraki [35], who applied dynamic programming techniques for route evaluations. The list of covered attributes includes soft and multiple time windows, as well as time-dependent and flexible travel times. The hybrid ILS and set covering heuristic of Subramanian, Uchoa, and Ochi [76] efficiently solves a variety of problems with multiple depots, heterogeneous fleet, and simultaneous or mixed pickup and deliveries. Finally, the *Unified Hybrid Genetic Search* (UHGS) heuristic of Vidal et al. [88] was specifically designed to be highly flexible. Problem-specific

Table 4.1. *Key metaheuristic strategies.*

Diversification		**Intensification**	
Noise in edge costs	ALNS [63]		
Penalization of known solution features	GLS [89]	Incentives on elite solution features	[79]
Incentive on under-explored solution features		Pheromones	ACO
Penalized infeasible solutions: load, distance, or number of routes	[24], others		
Crossover and recombinations	GA, ES, SS	Enumerative neighborhoods: RELOCATE, SWAP, 2-OPT, among others	[50]
Variable neighborhoods	VNS [54]	Variable neighborhoods	VNS [54]
Large neighborhoods: ruin-and-recreate, ejection chains, set covering, among others	[75], [1], others	Large neighborhoods: ruin-and-recreate, ejection chains, set covering, among others	[75], [1], others
Randomized choice of neighbors: random first improvement, top X% savings	Various papers	Greedy choice of neighbors: best improvement, best saving	Various papers
Deteriorating moves	SA [61]	Deterioration thresholds	RRT [21]
Tabu memories	TS [15, 24]	Elite guiding solutions	PR [28]
Restarts and jumps on unexplored solutions	Various papers	Restart and jumps on elite solutions	Various papers
		False starts	[24], others
		Multiple randomized LS on the same solution	[65]
Populations of solutions	GA, ES	Exchange of elite solutions	Parallel [16, 41]
Parents and survivors selection with respect to diversity	[64, 85]	Parents and survivors selection with respect to solution cost	GA, ES
		Adaptation of method components relatively to performance measures	ALNS [63]
Fixing under-explored solution elements, uncoarsening	[32, 90]	Fixing variables from elite solutions, coarsening	[32, 90]
		Decomposition phases based on routes from an elite solution	Various papers

Acronyms : ACO – Ant Colony Optimization, ALNS – Adaptive Large Neighborhood Search, ES – Evolutionary Strategy, GA – Genetic Algorithm, GLS – Guided Local Search, PR – Path Relinking, RRT – Record-to-Record Travel, SA – Simulated Annealing, SS – Scatter Search, TS – Tabu Search, VNS – Variable Neighborhood Search.

features are relegated to small modular components, thus allowing the algorithm to be highly competitive on about 30 different problems.

Designing more unified solvers poses significant new challenges. As long as only a few attributes are considered, producing a unified method for a rich formulation merging all attributes is a feasible alternative. However, some attributes, such as soft time windows, hours of service regulations, or loading constraints, lead to complex and time-consuming solution evaluations and local search procedures. Attribute-related parameters may be set to default values when dealing with subproblems, e.g., time windows of $[-\infty, \infty]$, but heavy computations with dummy values may still be performed, with adverse consequences on computational performance. Finally, tailored strategies known to be efficient for specific attributes, e.g., efficient move evaluation techniques, may not apply for the general problem. For these reasons, it is sometimes necessary to design efficient methods that focus on the specific attributes of the problem at hand. This may require some form of adaptation relatively to the problem characteristics, as was attempted in Vidal et al. [88].

Finally, significant care should be paid to the calibration of parameters which may be highly correlated to the specific structure of the problems and instances, such as the distribution of customer locations, route size, and time window width. Thus, when moving towards more flexible methods, it is necessary to study not only the number of parameters but also their sensitivity to several problem features. This consideration applies in general to any choice of algorithm hybridization where some components are useful in some settings but detrimental in others. For this reason, adaptive heuristic frameworks such as ALNS (see Pisinger and Ropke [63]) and other hyper-heuristics (heuristics that select or build heuristics; see Burke et al. [9]) may prove very helpful.

4.7 ▪ Computational Comparison of Selected Metaheuristics

In this section we report on computational experiments from a selected set of successful metaheuristics. We compare the metaheuristics in terms of solution quality and speed, and we also touch upon issues such as simplicity, flexibility, and parameter sensitivity in the comparison. Table 4.2 lists the metaheuristics included in the computational

Table 4.2. *Metaheuristics included in the computational comparison.*

Name	Reference
CLM01	Cordeau, Laporte, and Mercier [15]
TV03	Toth and Vigo [84]
RDH04	Reimann, Doerner, and Hartl [66]
T05	Tarantilis [79]
MB07	Mester and Bräysy [53]
PR07	Pisinger and Ropke [63]
NB09	Nagata and Bräysy [57]
P09	Prins [65]
GGW10	Groër, Golden, and Wasil [31]
ZK10	Zachariadis and Kiranoudis [92]
GGW11	Groër, Golden, and Wasil [32]
CM12	Cordeau and Maischberger [16]
JCL12	Jin, Crainic, and Løkketangen [41]
VCGLR12	Vidal et al. [85]
SUO13	Subramanian, Uchoa, and Ochi [76]

comparison, along with the abbreviations we use to identify them. We have selected metaheuristics proposed after the year 2000 and that perform well with respect to at least one of the criteria listed above.

4.7.1 ▪ Benchmark Data Sets

The two most widely used benchmark data sets for the VRP are those proposed by Christofides, Mingozzi, and Toth [11] and by Golden et al. [30]. We will refer to these as the *CMT* and the *GWKC* data sets, respectively. The first set contains 14 instances ranging from 50 to 200 customers, while the second one contains 20 instances involving from 240 to 483 customers. In this section we compare the selected metaheuristics using these data sets. Instances from Taillard [77] and Rochat and Taillard [70] (denoted *RT* instances in the following) and from Li, Golden, and Wasil [48] (denoted *LGW* instances) are also used but are not as widely accepted as the CMT and GWKC data sets. The RT set contains 13 instances ranging from 75 to 385 customers, and the LGW set contains 12 instances ranging from 560 to 1200 customers. The instances in the GWKC and the LGW data sets are all distributed spatially in symmetric patterns, while those from the CMT and RT sets have a more realistic appearance.

4.7.2 ▪ Comparing Computation Time and Solution Quality

We compare the metaheuristics in terms of solution quality and computation time. Performing such a comparison is difficult since we do not have access to all the metaheuristics and we must rely on the results reported in the respective papers. Typically, several sets of results are reported in the papers for each instance. A stochastic metaheuristic can, for example, have been run 10 times on the same instance, and the average solution quality along with the best solution value obtained are typically reported. Sometimes the best result "found during testing" is also reported. However, often little information is provided about how many test runs were performed. In order to make our comparison as fair as possible, we use the best result reported for which we know the computational effort. In the example above we would report the best result found in the 10 runs, as well as the time needed to perform all runs. We would not report the best result found during testing since we do not know how much effort was put into obtaining this result. Typically, we report results from several configurations of each metaheuristic in order to show what results can be obtained with different running times. Even with this approach, it can be difficult to compare results since authors often have quite different measures for what constitutes an acceptable running time. Also, the fact that computational tests are performed on different computers makes it difficult to compare running times. To partly remedy this problem we normalize the running times to match an Intel Core i7 CPU running at 2.93 GHz. We use the CINT2006, CINT2000, and CINT95 benchmarks (see the Standard Performance Evaluation Corporation web page, http://www.spec.org) to perform the normalization. However, the result of the normalization should only be taken as a rough estimate. Comparing sequential and parallel metaheuristics poses another problem. Should one simply compare the "wall clock" time spent by each metaheuristic, thereby giving a huge advantage to the parallel metaheuristics, or should the wall clock time be multiplied by the number of threads that were run in parallel? We have chosen the latter approach, but we realize that it places parallel metaheuristics at a disadvantage because of the communication overhead. The parallel metaheuristics included in the comparison are CGW11 and JCL12. The CM12 metaheuristic is also parallel, but the results we report are from a sequential run.

Tables 4.3 and 4.4 contain aggregated results for the CMT and GWKC instances, respectively. The tables contain five columns each. The first column identifies the paper using the abbreviations of Table 4.2, the second column shows the configuration used (we refer to the original papers for further details), and the third column shows the average gap to the best known solution. For each algorithm and each instance in the data set we calculate the percentage gap as $100(z - z^b)/z^b$, where z is the solution value obtained by the metaheuristic and z^b is the best known solution value for the instance. The gaps are averaged over all instances in the data set. Columns 4 and 5 show the normalized and original computation time per instance, respectively. We refer to the original papers for information about the computers used in the experiments. Both tables are sorted according to solution quality. We note that not all metaheuristics are present in Table 4.3, either because results were omitted from the original paper or because the paper did not provide enough information.

Table 4.3. *Computational results for the CMT instances. Note that the result for the CLM*01 *metaheuristic is from the survey by Cordeau et al.* [13].

Heuristic	Configuration	Gap (%)	Time (s)	Time* (s)
NB09	Best of 10 runs	0.00	506	831
SUO13	Best of 10 runs	0.00	1008	1008
GGW11	129 threads, best of 5 runs	0.00	138899	193500
VCGLR12	Average of 10 runs – 50,000 it.	0.02	356	585
NB09	Average of 10 runs	0.03	51	83
MB07	Best configuration	0.03	87	163
GGW11	8 threads, best of 5 runs	0.03	4102	5714
VCGLR12	Average of 10 runs – 10,000 it.	0.05	81	133
GGW11	4 threads, best of 5 runs	0.05	2051	2857
P09	-	0.07	9	16
MB07	Fast configuration	0.08	2	3
SUO13	Average of 10 runs	0.08	101	101
PR07	Best of 10 runs – 50,000 it.	0.11	593	1046
T05	Standard configuration	0.18	21	338
GGW10	Set partitioning	0.28	7	9
GGW10	Ejection - random	0.31	17	22
PR07	Average of 10 runs – 50,000 it.	0.31	59	105
CLM01	-	0.56	529	1477
TV03	-	0.64	5	230

One sees that all metaheuristics in the comparison achieve good results on the CMT instances, but their running times vary significantly. From a practical point of view, the solution quality obtained is more than adequate for typical applications. For the larger instances of the GWKC data set, one observes a larger spread in solution quality. The increased difficulty of these instances is also illustrated by the fact that we have witnessed a steady stream of new best solutions for the GWKC data set in the last decade, while only one improved solution has been found for the CMT data set. The difference in computational effort spent is striking, ranging from seven seconds to more than 200,000 seconds per instance (using normalized times). The trade-off between time and solution quality for the GWKC data set is depicted in Figure 4.3. The normalized computation time is shown on the x-axis using a logarithmic scale, while solution quality, measured as the average gap, is displayed along the y-axis. Most configurations from Table 4.4 are shown

Table 4.4. *Computational results for the GWKC data set. We note that the time spent for the CM12 metaheuristic probably is underestimated since the paper only reports time averaged over both the GWKC and the CMT data sets and the CMT instances typically are solved faster than the GWKC instances. We also note that the results for the CLM01 and RDH04 metaheuristics are from the survey by Cordeau et al.* [13].

Heuristic	Configuration	Gap (%)	Time (s)	Time* (s)
GGW11	129 threads, best of 5 runs	0.12	138899	193500
VCGLR12	Average of 10 runs, 50,000 iterations	0.16	3387	5563
NB09	Best of 10 runs	0.17	13006	21359
ZK10	Best of 10 runs	0.22	14128	24300
VCGLR12	Average of 10 runs, 10,000 iterations	0.27	1042	1712
NB09	Average of 10 runs	0.27	1301	2136
GGW11	8 threads, best of 5 runs	0.30	8470	11800
MB07	Best configuration	0.33	782	1461
JCL12	8 threads, best of 10 runs	0.35	200978	200978
SUO13	Best of 10 runs	0.40	39382	39382
ZK10	Average of 10 runs	0.43	1413	2430
GGW11	4 threads, best of 5 runs	0.44	4235	5900
CM12	Best of 10, 10^6 iterations	0.56	18770	18770
SUO13	Average of 10 runs	0.56	3938	3938
P09	-	0.63	233	436
JCL12	8 threads, average of 10 runs	0.60	20098	20098
PR07	Best of 10 runs, 50,000 iterations	0.82	3662	6457
T05	Standard	0.93	169	2729
RDH04	-	0.93	599	2960
CM12	Average of 10 runs, 10^6 iterations	0.94	1877	1877
GGW10	Ejection - random	1.19	43	55
MB07	Fast configuration	1.23	7	13
GGW10	Set partitioning	1.27	10	13
PR07	Average of 10 runs, 50,000 iterations	1.35	366	646
CM12	Average of 10 runs, 10^5 iterations	1.46	188	188
CLM01	-	1.79	1207	3366.6
TV03	-	3.21	21	1053

as data points. However, we have left out the configurations with gaps of more than 2% to make the figure less cluttered. One can see that configurations MB07-B, CGW10-A, T05, P09, MB07-A, VCGLR12-A, VCGLR12-B, and CGW11-A define the Pareto front (meaning that any other configuration is dominated by one of these configurations). We acknowledge that a different benchmark for normalizing time may yield a slightly different Pareto front. Configurations CGW10-B and NB09-B are, for example, very close to being part of the Pareto front. We therefore highlight the metaheuristics MB07, CGW10, T05, P09, VCGLR12, NB09, and CGW11 as outstanding when it comes to either solution quality, computing time, or a combination of these two statistics.

For completeness we include detailed results for the CMT and GWKC instances in Tables 4.5–4.6 and Tables 4.7–4.9, respectively. These tables only include results for one

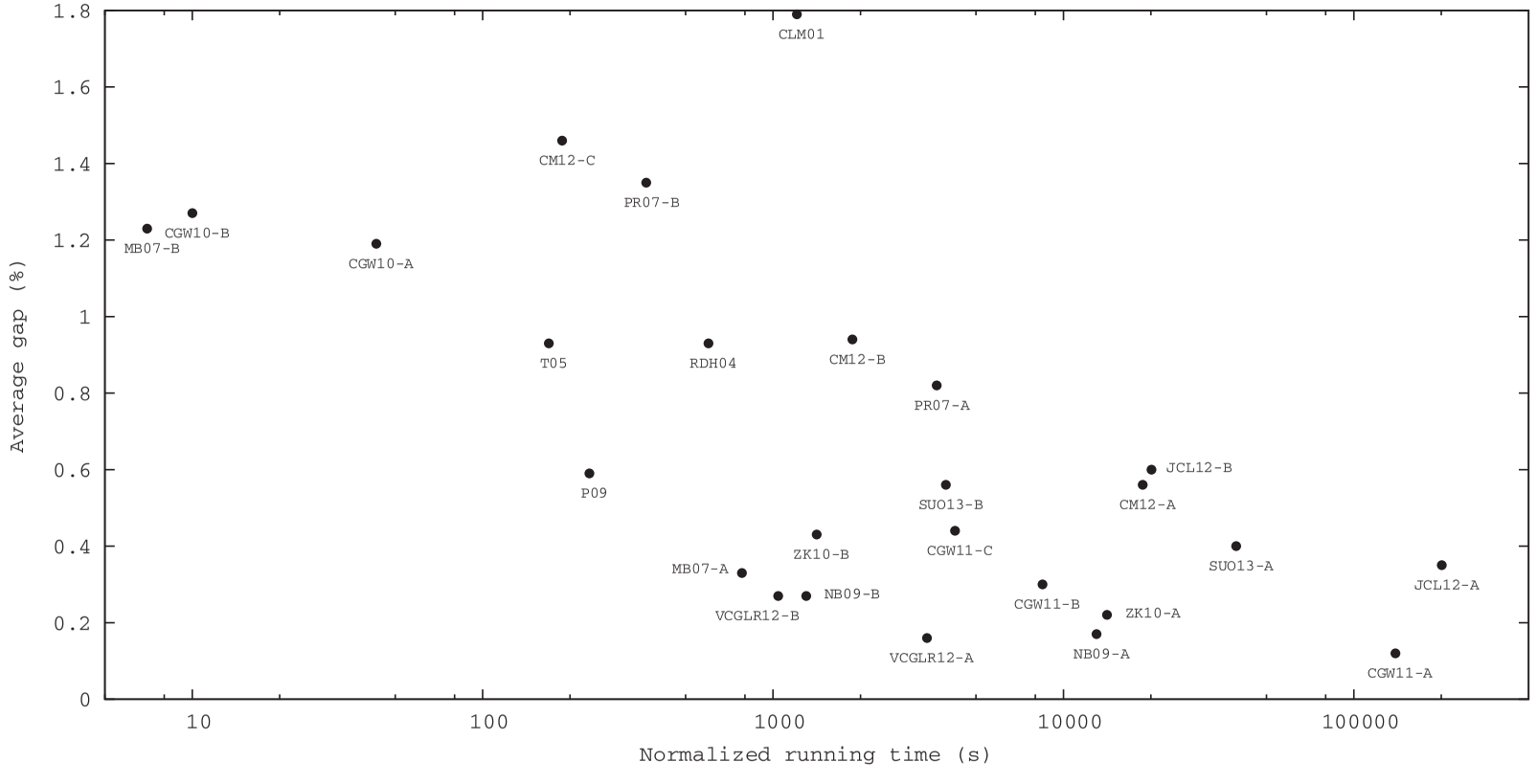

Figure 4.3. *Solution quality vs. running time for the GWKC instances. The x-axis shows the average computation time per instance in seconds (logarithmic scale), while the y-axis shows the solution quality measured as the average percent deviation from best known solution. Each label identifies a heuristic using the identifiers defined in Table* 4.2. *If several configurations of a heuristic are used, then each configuration is indicated by a letter "A", "B", or "C" after the heuristic identifier, with A being the most powerful configuration. The data for the table can be found in Table* 4.4.

Table 4.5. *CMT data set, detailed computational results part* I. *Results for the CLM*01 *metaheuristic are from the survey by Cordeau et al.* [13].

		Best	CLM01		TV03		T05		MB07	
#	n	known	Value	%	Value	%	Value	%	Value	%
1	50C	**524.61**	**524.61**	0.00	**524.61**	0.00	**524.61**	0.00	**524.61**	0.00
2	75C	**835.26**	**835.28**	0.00	838.6	0.40	**835.26**	0.00	**835.26**	0.00
3	100C	**826.14**	**826.14**	0.00	828.56	0.29	**826.14**	0.00	**826.14**	0.00
4	150C	**1028.42**	1032.68	0.41	1033.21	0.47	1029.64	0.12	**1028.42**	0.00
5	199C	**1291.29**	1315.76	1.90	1318.25	2.09	1311.48	1.56	**1291.29**	0.00
6	50CD	**555.43**	**555.43**	0.00	**555.43**	0.00	**555.43**	0.00	**555.43**	0.00
7	75CD	**909.68**	**909.68**	0.00	920.72	1.21	**909.68**	0.00	**909.68**	0.00
8	100CD	**865.94**	**865.95**	0.00	869.48	0.41	**865.94**	0.00	**865.94**	0.00
9	150CD	**1162.55**	1167.85	0.46	1173.12	0.91	1163.19	0.06	**1162.55**	0.00
10	199CD	**1395.85**	1416.84	1.50	1435.74	2.86	1407.21	0.81	1401.12	0.38
11	120C	**1042.11**	1073.47	3.01	1042.87	0.07	**1042.11**	0.00	**1042.11**	0.00
12	100C	**819.56**	**819.56**	0.00	**819.56**	0.00	**819.56**	0.00	**819.56**	0.00
13	120CD	**1541.14**	1549.25	0.53	1545.51	0.28	1544.01	0.19	**1541.14**	0.00
14	100CD	**866.37**	**866.37**	0.00	**866.37**	0.00	**866.37**	0.00	**866.37**	0.00
Average				0.56		0.64		0.20		0.03
Normalized time (s)			529.5		4.5		20.8		87.0	
		Best	PR07		P09		NB09		GGW10	
#	n	known	Value	%	Value	%	Value	%	Value	%
1	50C	**524.61**	**524.61**	0.00	**524.61**	0.00	**524.61**	0.00	**524.61**	0.00
2	75C	**835.26**	**835.26**	0.00	**835.26**	0.00	**835.26**	0.00	842.37	0.85
3	100C	**826.14**	**826.14**	0.00	**826.14**	0.00	**826.14**	0.00	827.39	0.15
4	150C	**1028.42**	1029.56	0.11	1029.48	0.10	**1028.42**	0.00	1032.12	0.36
5	199C	**1291.29**	1297.12	0.45	1294.09	0.22	1291.45	0.01	1307.36	1.24
6	50CD	**555.43**	**555.43**	0.00	**555.43**	0.00	**555.43**	0.00	**555.43**	0.00
7	75CD	**909.68**	**909.68**	0.00	**909.68**	0.00	**909.68**	0.00	**909.68**	0.00
8	100CD	**865.94**	**865.94**	0.00	**865.94**	0.00	**865.94**	0.00	**865.94**	0.00
9	150CD	**1162.55**	1163.68	0.10	**1162.55**	0.00	**1162.55**	0.00	1164.12	0.14
10	199CD	**1395.85**	1405.88	0.72	1401.46	0.40	**1395.85**	0.00	1410.97	1.08
11	120C	**1042.11**	**1042.12**	0.00	**1042.11**	0.00	**1042.11**	0.00	**1042.11**	0.00
12	100C	**819.56**	**819.56**	0.00	**819.56**	0.00	**819.56**	0.00	**819.56**	0.00
13	120CD	**1541.14**	1542.86	0.11	1545.43	0.28	**1541.14**	0.00	1542.86	0.11
14	100CD	**866.37**	**866.37**	0.00	**866.37**	0.00	**866.37**	0.00	**866.37**	0.00
Average				0.11		0.07		0.00		0.28
Normalized time (s)			593.1		8.5		506.2		7.3	

configuration per paper. We include the configuration that obtains the best solution quality within a set time limit. Each table starts with three columns describing the instances. The first column contains the instance name, and the next column shows the number of customers in the instance and indicates whether the instance is constrained by capacity only (C) or by both capacity and distance (or duration) (CD). The third column provides the best known solution value for the instance. For each metaheuristic we provide the solution value obtained by the chosen configuration and the percentage gap. When a metaheuristic obtains the best known solution value, it is highlighted in bold. For each metaheuristic the table also displays the average gap and the average time spent per instance (using normalized times) after reporting results on the individual instances.

Table 4.6. *CMT data set, detailed computational results part* II.

		Best	GGW11		VCGLR12		SUO13	
#	n	known	Value	%	Value	%	Value	%
1	50C	**524.61**	**524.61**	0.00	**524.61**	0.00	**524.61**	0.00
2	75C	**835.26**	**835.26**	0.00	**835.26**	0.00	**835.26**	0.00
3	100C	**826.14**	**826.14**	0.00	**826.14**	0.00	**826.14**	0.00
4	150C	**1028.42**	**1028.42**	0.00	**1028.42**	0.00	**1028.42**	0.00
5	199C	**1291.29**	1291.45	0.01	1291.74	0.03	1291.45	0.01
6	50CD	**555.43**	**555.43**	0.00	**555.43**	0.00	**555.43**	0.00
7	75CD	**909.68**	**909.68**	0.00	**909.68**	0.00	**909.68**	0.00
8	100CD	**865.94**	**865.94**	0.00	**865.94**	0.00	**865.94**	0.00
9	150CD	**1162.55**	**1162.55**	0.00	**1162.55**	0.00	**1162.55**	0.00
10	199CD	**1395.85**	**1395.85**	0.00	1397.7	0.13	**1395.85**	0.00
11	120C	**1042.11**	**1042.11**	0.00	**1042.11**	0.00	**1042.11**	0.00
12	100C	**819.56**	**819.56**	0.00	**819.56**	0.00	**819.56**	0.00
13	120CD	**1541.14**	**1541.14**	0.00	1542.86	0.11	**1541.14**	0.00
14	100CD	**866.37**	**866.37**	0.00	**866.37**	0.00	**866.37**	0.00
Average				0.00		0.02		0.00
Normalized time (s)			138898.5		356.4		1008.2	

4.7.3 ▪ Other Measures of Quality

The preceding section focused on solution quality and running time. These may be the two most obvious and the easiest criteria used to assess the quality of an algorithm, but other criteria are also relevant when comparing metaheuristics (see, e.g., Cordeau et al. [14]). Here we discuss such criteria and point out a few metaheuristics that perform especially well for one or more of these. Desirable features of a VRP metaheuristics are the following.

Simplicity. Ideally, a metaheuristic should be simple in the sense that it should be easy to understand, implement, and fine-tune. Focusing on improving solution quality may to some extent be detrimental to the objective of producing simple metaheuristics since there exists a tendency to take successful components from many past metaheuristics and combine them into new metaheuristics that perform slightly better than the previous generation but are more complex. Simplicity may come at the expense of speed or accuracy, but a successful metaheuristic should strike a reasonable balance between the criteria. Examples of relatively simple metaheuristics in recent years are those of Cordeau, Laporte, and Mercier [15], Toth and Vigo [84], Prins [64], and Groër, Golden, and Wasil [31].

Number of Parameters and Sensitivity to Parameters. This feature is clearly related to the previous one. We prefer metaheuristics that are controlled by few parameters, and ideally the metaheuristic should not be too sensitive to the parameters in the sense that the same set of parameters should be used for a wide range of instances. The metaheuristics of Groër, Golden, and Wasil [31] are controlled by relatively few parameters, and so are the TS algorithms of Cordeau, Laporte, and Mercier [15] and Toth and Vigo [84]. Another way of avoiding the curse of parameters is to let the metaheuristic select some of the parameters itself. Examples of such metaheuristics are the reactive TS of Battiti and Tecchiolli [6] and the ALNS metaheuristic of Ropke and Pisinger [63, 71].

Table 4.7. *GWKC data set, detailed computational results part* I. *Results for the CLM*01 *and RDH*04 *metaheuristics are from the survey by Cordeau et al.* [13].

		Best	CLM01		TV03		RDH04		T05 standard		MB07	
#	n	known	Value	%	Value	%	Value	%	value	%	value	%
1	240C	**5623.47**	5681.97	1.04	5736.15	2.00	5644.02	0.37	5676.97	0.95	5627.54	0.07
2	320C	**8404.61**	8657.36	3.01	8553.03	1.77	8449.12	0.53	8459.91	0.66	8447.92	0.52
3	400C	**11030.80**	11037.40	0.06	11402.75	3.37	11036.22	0.05	11036.22	0.05	11036.22	0.05
4	480C	**13592.88**	13740.60	1.09	14910.62	9.69	13699.11	0.78	13637.53	0.33	13624.52	0.23
5	200C	**6460.98**	6756.44	4.57	6697.53	3.66	**6460.98**	0.00	**6460.98**	0.00	**6460.98**	0.00
6	280C	**8403.25**	8537.17	1.59	8963.32	6.66	8412.90	0.11	8414.28	0.13	8412.88	0.11
7	360C	**10102.68**	10267.40	1.63	10547.44	4.40	10195.59	0.92	10216.50	1.13	10195.56	0.92
8	440C	**11635.34**	11869.50	2.01	12036.24	3.45	11828.78	1.66	11936.16	2.59	11663.55	0.24
9	255CD	**579.71**	587.39	1.32	593.35	2.35	586.87	1.24	585.43	0.99	583.39	0.63
10	323CD	**736.26**	752.76	2.24	751.66	2.09	750.77	1.97	746.56	1.40	741.56	0.72
11	399CD	**912.84**	929.07	1.78	936.04	2.54	927.27	1.58	923.17	1.13	918.45	0.61
12	483CD	**1102.69**	1119.52	1.53	1147.14	4.03	1140.87	3.46	1130.40	2.51	1107.19	0.41
13	252CD	**857.19**	875.88	2.18	868.80	1.35	865.07	0.92	865.01	0.91	859.11	0.22
14	320CD	**1080.55**	1102.03	1.99	1096.18	1.45	1093.77	1.22	1086.07	0.51	1081.31	0.07
15	396CD	**1337.92**	1363.76	1.93	1369.44	2.36	1358.21	1.52	1353.91	1.20	1345.23	0.55
16	480CD	**1612.50**	1647.06	2.14	1652.32	2.47	1635.16	1.41	1634.74	1.38	1622.69	0.63
17	240CD	**707.76**	710.93	0.45	711.07	0.47	708.76	0.14	708.74	0.14	707.79	0.00
18	300CD	**995.13**	1014.62	1.96	1016.83	2.18	998.83	0.37	1006.90	1.18	998.73	0.36
19	360CD	**1365.60**	1383.79	1.33	1400.96	2.59	1367.20	0.12	1371.01	0.40	1366.86	0.09
20	420CD	**1818.25**	1854.24	1.98	1915.83	5.37	1822.94	0.26	1837.67	1.07	1820.09	0.10
Average				1.79		3.21		0.93		0.93		0.33
Normalized time (s)			1206.7		20.6		598.9		168.5		782.0	

Table 4.8. *GWKC data set, detailed computational results part* II.

#	n	Best known	PR07 Value	PR07 %	P09 Value	P09 %	NB09 Value	NB09 %	GGW10 Value	GGW10 %	ZK10 Value	ZK10 %
1	240C	**5623.47**	5650.91	0.49	5644.52	0.37	5626.81	0.06	5662.23	0.69	5626.81	0.06
2	320C	**8404.61**	8469.32	0.77	8447.92	0.52	8431.66	0.32	8466.92	0.74	8447.92	0.52
3	400C	**11030.80**	11047.01	0.15	11036.22	0.05	11036.22	0.05	11078.85	0.44	11036.22	0.05
4	480C	**13592.88**	13635.31	0.31	13624.52	0.23	**13592.88**	0.00	13634.11	0.30	13624.53	0.23
5	200C	**6460.98**	6466.68	0.09	**6460.98**	0.00	**6460.98**	0.00	6472.38	0.18	**6460.98**	0.00
6	280C	**8403.25**	8416.13	0.15	8412.90	0.11	8404.26	0.01	8415.21	0.14	8412.90	0.11
7	360C	**10102.68**	10181.75	0.78	10195.59	0.92	10156.58	0.53	10195.59	0.92	10169.26	0.66
8	440C	**11635.34**	11713.62	0.67	11643.90	0.07	11691.06	0.48	11869.61	2.01	11651.67	0.14
9	255CD	**579.71**	585.14	0.94	586.23	1.12	580.42	0.12	588.31	1.48	581.28	0.27
10	323CD	**736.26**	748.89	1.72	744.36	1.10	738.49	0.30	747.76	1.56	738.57	0.31
11	399CD	**912.84**	922.70	1.08	922.40	1.05	914.72	0.21	934.85	2.41	916.99	0.45
12	483CD	**1102.69**	1119.06	1.48	1116.12	1.22	1106.76	0.37	1133.01	2.75	1105.93	0.29
13	252CD	**857.19**	864.68	0.87	862.32	0.60	**857.19**	0.00	867.11	1.16	858.45	0.15
14	320CD	**1080.55**	1095.40	1.37	1089.35	0.81	**1080.55**	0.00	1093.69	1.22	1081.05	0.05
15	396CD	**1337.92**	1359.94	1.65	1352.39	1.08	1342.53	0.34	1360.68	1.70	1341.46	0.26
16	480CD	**1612.50**	1639.11	1.65	1634.27	1.35	1620.85	0.52	1639.24	1.66	1617.48	0.31
17	240CD	**707.76**	708.90	0.16	708.85	0.15	**707.76**	0.00	709.56	0.25	**707.76**	0.00
18	300CD	**995.13**	1002.42	0.73	1002.15	0.71	995.13	0.00	1012.66	1.76	996.55	0.14
19	360CD	**1365.60**	1374.24	0.63	1371.67	0.44	1365.97	0.03	1381.01	1.13	1366.75	0.08
20	420CD	**1818.25**	1830.80	0.69	1830.98	0.70	1820.02	0.10	1842.57	1.34	1824.46	0.34
Average				0.82		0.63		0.17		1.19		0.22
Normalized time (s)			3662.0		233.4		13005.5		43.4		14127.9	

Table 4.9. *GWKC data set, detailed computational results part* III.

#	n	Best known	GGW11 Value	GGW11 %	CM12 Value	CM12 %	JCL12 Value	JCL12 %	VCGLR12 Value	VCGLR12 %	SUO13 Value	SUO13 %
1	240C	**5623.47**	**5623.47**	0.00	5643.97	0.36	**5623.47**	0.00	5625.10	0.03	5657.74	0.61
2	320C	**8404.61**	8447.92	0.52	8445.70	0.49	8419.50	0.18	8419.25	0.17	8447.92	0.52
3	400C	**11030.80**	11036.22	0.05	11036.20	0.05	**11030.80**	0.00	11036.22	0.05	11036.22	0.05
4	480C	**13592.88**	13624.52	0.23	13629.30	0.27	13615.20	0.16	13624.53	0.23	13624.53	0.23
5	200C	**6460.98**	**6460.98**	0.00	**6460.98**	0.00	**6460.98**	0.00	**6460.98**	0.00	**6460.98**	0.00
6	280C	**8403.25**	8412.90	0.11	8412.90	0.11	**8403.25**	0.00	8412.90	0.11	8412.90	0.11
7	360C	**10102.68**	10195.59	0.92	10183.50	0.80	10184.40	0.81	10134.90	0.32	10195.58	0.92
8	440C	**11635.34**	11663.55	0.24	11673.70	0.33	11671.00	0.31	**11635.34**	0.00	11710.47	0.65
9	255CD	**579.71**	**579.71**	0.00	582.95	0.56	581.73	0.35	581.08	0.24	583.24	0.61
10	323CD	**736.26**	737.28	0.14	739.95	0.50	738.50	0.30	738.92	0.36	741.96	0.77
11	399CD	**912.84**	913.35	0.06	920.54	0.84	914.98	0.23	914.37	0.17	921.46	0.94
12	483CD	**1102.69**	1102.76	0.01	1111.76	0.82	1109.93	0.66	1105.97	0.30	1113.30	0.96
13	252CD	**857.19**	**857.19**	0.00	865.27	0.94	861.92	0.55	859.08	0.22	**857.19**	0.00
14	320CD	**1080.55**	**1080.55**	0.00	1085.76	0.48	1082.52	0.18	1081.99	0.13	**1080.55**	0.00
15	396CD	**1337.92**	1338.19	0.02	1355.86	1.34	1351.13	0.99	1341.95	0.30	1347.13	0.69
16	480CD	**1612.50**	1613.66	0.07	1633.84	1.32	1629.78	1.07	1616.92	0.27	1624.55	0.75
17	240CD	**707.76**	**707.76**	0.00	708.17	0.06	707.83	0.01	707.84	0.01	**707.76**	0.00
18	300CD	**995.13**	**995.13**	0.00	1000.86	0.58	1000.27	0.52	996.95	0.18	995.65	0.05
19	360CD	**1365.60**	**1365.60**	0.00	1373.07	0.55	1367.31	0.13	1366.39	0.06	1366.29	0.05
20	420CD	**1818.25**	**1818.25**	0.00	1832.65	0.79	1827.39	0.50	1819.75	0.08	1821.16	0.16
Average				0.12		0.56		0.35		0.16		0.40
Normalized time (s)			138898.5		18770.0		200978.4		3387.5		39382.3	

It is hard to single out metaheuristics that are robust with respect to parameter settings since it is not common to publish detailed parameter tuning results. However, it is safe to say that the metaheuristics we characterize as flexible in the next paragraph are relatively insensitive to parameter settings since they often are able to solve different variants of the VRP using the same set of parameters.

Flexibility. It is desirable for metaheuristics to be able to solve other variants of the VRP with little or no changes. Such variants could, for example, include multiple depots, pickups and deliveries, or time windows. Such a feature is especially important for software developers since they often face the challenge of producing vehicle routing solvers for several users who rarely have exactly the same VRP. We have surveyed such flexible metaheuristics in Section 4.6, and we highlight those of Cordeau, Laporte, and Mercier [15], Pisinger and Ropke [63], Cordeau and Maischberger [16], Vidal et al. [85], and Subramanian, Uchoa, and Ochi [76] as metaheuristics that have proven their ability to solve many VRP variants. The genetic algorithm of Prins [64] is also rather flexible since the method has been extended to many other VRP types even though the original paper was focused on the VRP.

Robustness. One would like metaheuristics to return a good solution consistently since in real life an occasional poor solution can lead to financial losses or a loss of faith in the metaheuristic. The robustness of a heuristics can to some extent be judged by inspecting the detailed results of Tables 4.5–4.9. It can be seen that there exists a high correlation between robust results and the good overall solution quality measured in Tables 4.3 and 4.4.

Taking Advantage of Parallel Processing. Compared with what we have witnessed between 1960 and 2000, the speed of a single processing unit has improved relatively slowly in recent years. At the same time, CPUs have been equipped with more and more cores, thus enabling parallel processing on standard desktop computers. It looks as if this trend will continue in the near future, and it therefore seems advisable to parallelize algorithms in order to take full advantage of the current and next generation of computers. It is possible to use parallel processing for almost any existing metaheuristic, simply by launching several copies of the metaheuristic on several processors, each starting from a different solution. However, more sophisticated approaches are of course possible, as we have pointed out in Section 4.5. We note that among the metaheuristics mentioned in this section, those of Groër, Golden, and Wasil [32], Jin, Crainic, and Løkketangen [41] and Cordeau and Maischberger [16] can already take advantage of parallel processing, and population-based methods are typically easy to parallelize since the work involved in combining solutions from the population or in improving individual solutions can often be performed in parallel. Some authors have suggested parallelizing other classical metaheuristics (see, for example, Alba [2] and Crainic and Toulouse [17]).

4.8 ▪ Conclusions and Future Research Directions

The past 10 years have seen the emergence of several powerful metaheuristics for the VRP, mostly derived from the hybridization of concepts initially developed independently from each other. The best known algorithms typically combine several search principles rooted in early local search heuristics and genetic algorithms. The best metaheuristics make use of sophisticated neighborhoods, exact optimization methods, decom-

position strategies, and cooperation principles. Heuristics are also becoming more flexible in the sense that they can sometimes be applied, without any structural change, to a wide range of VRP extensions and variations.

Even though we do not yet know the optimal solutions for most of the instances used in the computational comparison, it is safe to say that current metaheuristics are capable of producing high quality solutions for instances with up to 500 customers, and solution quality has improved steadily over the last decade. However, the gains in solution quality are now becoming marginal, an indication that the current solution methodologies may have reached a plateau.

It may now be worthwhile considering new more realistic and varied large-scale benchmark instances and expending more efforts toward conceptual simplicity, parsimony of parameters, robustness, and flexibility. We also believe that the research effort in the coming years will be aimed at developing a new generation of metaheuristics that can yield the same solution quality as those of today but within significantly less time. One promising avenue of research toward this goal lies in the hybridization of heuristic concepts with MIP solvers, which are becoming increasingly powerful.

We have witnessed the emergence of new research streams in the field of vehicle routing, namely in the areas of green logistics, city logistics, and humanitarian logistics. We expect that these emerging fields will continue to grow in the coming years. Some other problem characteristics, such as synchronization, service consistency, balanced work allocation, congestion, and speed optimization, have emerged, giving rise to a new generation of difficult problems which require methodological developments.

Bibliography

[1] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen, *A survey of very large-scale neighborhood search techniques*, Discrete Applied Mathematics, 123 (2002), pp. 75–102.

[2] E. Alba, ed., *Parallel Metaheuristics: A New Class of Algorithms*, Wiley-Interscience, Hoboken, NJ, 2005.

[3] K. Altinkemer and B. Gavish, *Parallel savings based heuristics for the delivery problem*, Operations Research, 39 (1991), pp. 456–469.

[4] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, Princeton, NJ, 2006.

[5] M. L. Balinski and R. E. Quandt, *On an integer program for a delivery problem*, Operations Research, 12 (1964), pp. 300–304.

[6] R. Battiti and G. Tecchiolli, *The reactive tabu search*, ORSA Journal on Computing, 6 (1994), pp. 128–140.

[7] J. Baxter, *Depot location: A technique for the avoidance of local optima*, European Journal of Operational Research, 18 (1984), pp. 208–214.

[8] J. E. Beasley, *Route first-cluster second methods for vehicle routing*, Omega, 11 (1983), pp. 403–408.

[9] E. K. BURKE, M. HYDE, G. KENDALL, G. OCHOA, E. ÖZCAN, AND J. R. WOODWARD, *A classification of hyper-heuristic approaches*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., Springer, New York, 2010, pp. 449–468.

[10] P. CHEN, H.-K. HUANG, AND X.-Y. DONG, *Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem*, Expert Systems with Applications, 37 (2010), pp. 1620–1627.

[11] N. CHRISTOFIDES, A. MINGOZZI, AND P. TOTH, *The vehicle routing problem*, in Combinatorial Optimization, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, eds., Wiley, Chichester, UK, 1979, pp. 315–338.

[12] G. CLARKE AND J. W. WRIGHT, *Scheduling of vehicles from a central depot to a number of delivery points*, Operations Research, 12 (1964), pp. 568–581.

[13] J.-F. CORDEAU, M. GENDREAU, A. HERTZ, G. LAPORTE, AND J.-S. SORMANY, *New heuristics for the vehicle routing problem*, in Logistics Systems: Design and Optimization, A. Langevin and D. Riopel, eds., Springer, New York, 2005, pp. 279–297.

[14] J.-F. CORDEAU, M. GENDREAU, G. LAPORTE, J.-Y. POTVIN, AND F. SEMET, *A guide to vehicle routing heuristics*, Journal of the Operational Research Society, 53 (2002), pp. 512–522.

[15] J.-F. CORDEAU, G. LAPORTE, AND A. MERCIER, *A unified tabu search heuristic for vehicle routing problems with time windows*, Journal of the Operational Research Society, 52 (2001), pp. 928–936.

[16] J.-F. CORDEAU AND M. MAISCHBERGER, *A parallel iterated tabu search heuristic for vehicle routing problems*, Computers & Operations Research, 39 (2012), pp. 2033–2050.

[17] T. G. CRAINIC AND M. TOULOUSE, *Parallel meta-heuristics*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., Springer, New York, 2010, pp. 497–541.

[18] J.-C. CRÉPUT, A. HAJJAM, A. KOUKAM, AND O. KUHN, *Self-organizing maps in population based metaheuristic to the dynamic vehicle routing problem*, Journal of Combinatorial Optimization, 24 (2012), pp. 437–458.

[19] G. B. DANTZIG AND J. H. RAMSER, *The truck dispatching problem*, Management Science, 6 (1959), pp. 80–91.

[20] M. DESROCHERS AND T. W. VERHOOG, *A matching based savings algorithm for the vehicle routing problem*, Techical Report, Les Cahiers du GERAD, G-89-04, HEC, Montréal, Canada, 1989.

[21] G. DUECK, *New optimization heuristics: The great deluge algorithm and the record-to-record travel*, Journal of Computational Physics, 104 (1993), pp. 86–92.

[22] G. DUECK AND T. SCHEUER, *Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing*, Journal of Computational Physics, 90 (1990), pp. 161–175.

[23] B. A. FOSTER AND D. M. RYAN, *An integer programming approach to the vehicle scheduling problem*, Operational Research Quarterly, 27 (1976), pp. 367–384.

[24] M. GENDREAU, A. HERTZ, AND G. LAPORTE, *A tabu search heuristic for the vehicle routing problem*, Management Science, 40 (1994), pp. 1276–1290.

[25] M. GENDREAU, G. LAPORTE, AND J.-Y. POTVIN, *Metaheuristics for the capacitated VRP*, in The Vehicle Routing Problem, P. Toth and D. Vigo, eds., SIAM, Philadelphia, 2002, pp. 129–154.

[26] M. GENDREAU AND J.-Y. POTVIN, eds., *Handbook of Metaheuristics*, Springer, New York, 2010.

[27] ——, *Tabu search*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., vol. 146, Springer, New York, 2010, pp. 41–59.

[28] F. GLOVER, *Heuristics for integer programming using surrogate constraints*, Decision Sciences, 8 (1977), pp. 156–166.

[29] ——, *Future paths for integer programming and links to artificial intelligence*, Computers & Operations Research, 13 (1986), pp. 533–549.

[30] B. L. GOLDEN, E. A. WASIL, J. P. KELLY, AND I.-M. CHAO, *The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results*, in Fleet Management and Logistics, T. G. Crainic and G. Laporte, eds., Kluwer, Boston, 1998, pp. 33–56.

[31] C. GROËR, B. L. GOLDEN, AND E. A. WASIL, *A library of local search heuristics for the vehicle routing problem*, Mathematical Programming Computation, 2 (2010), pp. 79–101.

[32] ——, *A parallel algorithm for the vehicle routing problem*, INFORMS Journal on Computing, 23 (2011), pp. 315–330.

[33] P. HANSEN, N. MLADENOVIĆ, J. BRIMBERG, AND J. A. MORENO PÉREZ, *Variable neighborhood search*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., Springer, New York, 2010, pp. 61–86.

[34] H. HASHIMOTO, T. IBARAKI, S. IMAHORI, AND M. YAGIURA, *The vehicle routing problem with flexible time windows and traveling times*, Discrete Applied Mathematics, 154 (2006), pp. 2271–2290.

[35] H. HASHIMOTO, M. YAGIURA, AND T. IBARAKI, *An iterated local search algorithm for the time-dependent vehicle routing problem with time windows*, Discrete Optimization, 5 (2008), pp. 434–456.

[36] K. HELSGAUN, *An effective implementation of the Lin-Kernighan traveling salesman heuristic*, European Journal of Operational Research, 126 (2000), pp. 106–130.

[37] V. C. HEMMELMAYR, K. F. DOERNER, AND R. F. HARTL, *A variable neighborhood search heuristic for periodic routing problems*, European Journal of Operational Research, 195 (2009), pp. 791–802.

[38] J. H. HOLLAND, *Adaptation in Natural and Artificial systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, The University of Michigan Press, Ann Arbor, MI, 1975.

[39] T. IBARAKI, S. IMAHORI, M. KUBO, T. MASUDA, T. UNO, AND M. YAGIURA, *Effective local search algorithms for routing and scheduling problems with general time-window constraints*, Transportation Science, 39 (2005), pp. 206–232.

[40] S. IRNICH, B. FUNKE, AND T. GRÜNERT, *Sequential search and its application to vehicle-routing problems*, Computers & Operations Research, 33 (2006), pp. 2405–2429.

[41] J. JIN, T. G. CRAINIC, AND A. LØKKETANGEN, *A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems*, European Journal of Operational Research, 222 (2012), pp. 441–451.

[42] D. S. JOHNSON AND L. A. MCGEOCH, *The traveling salesman problem: A case study in local optimization*, in Local Search in Combinatorial Optimization, E. H. L. Aarts and J. K. Lenstra, eds., Princeton University Press, Princeton, NJ, 1997, pp. 215–310.

[43] ——, *Experimental analysis of heuristics for the STSP*, in The Traveling Salesman Problem and Its Variations, G. Gutin and A. P. Punnen, eds., Kluwer, Dordrecht, 2002, pp. 369–443.

[44] S. KIRKPATRICK, C. D. GELATT, AND M. P. VECCHI, *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680.

[45] J. KYTÖJOKI, T. NUORTIO, O. BRÄYSY, AND M. GENDREAU, *An efficient variable neighborhood search heuristic for very large scale vehicle routing problems*, Computers & Operations Research, 34 (2007), pp. 2743–2757.

[46] G. LAPORTE AND F. SEMET, *Classical heuristics for the capacitated VRP*, in The Vehicle Routing Problem, P. Toth and D. Vigo, eds., SIAM, Philadelphia, 2002, ch. 5, pp. 109–128.

[47] A. LE BOUTHILLIER, T. G. CRAINIC, AND P. KROPF, *A guided cooperative search for the vehicle routing problem with time windows*, Intelligent Systems, IEEE, 20 (2005), pp. 36–42.

[48] F. LI, B. L. GOLDEN, AND E. A. WASIL, *Very large-scale vehicle routing: new test problems, algorithms, and results*, Computers & Operations Research, 32 (2005), pp. 1165–1179.

[49] S. LIN, *Computer solutions of the traveling salesman problem*, Bell System Technical Journal, 44 (1965), pp. 2245–2269.

[50] S. LIN AND B. W. KERNIGHAN, *An effective heuristic algorithm for the traveling-salesman problem*, Operations Research, 21 (1973), pp. 498–516.

[51] H. R. LOURENÇO, O. C. MARTIN, AND T. STÜTZLE, *Iterated local search: Framework and applications*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., Springer, New York, 2010, pp. 363–397.

[52] V. MANIEZZO, T. STÜTZLE, AND S. VOSS, eds., *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, Springer, New York, 2009.

[53] D. MESTER AND O. BRÄYSY, *Active-guided evolution strategies for large-scale capacitated vehicle routing problems*, Computers & Operations Research, 34 (2007), pp. 2964–2975.

[54] N. MLADENOVIĆ AND P. HANSEN, *Variable neighborhood search*, Computers & Operations Research, 24 (1997), pp. 1097–1100.

[55] P. MOSCATO AND C. COTTA, *A modern introduction to memetic algorithms*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., Springer, New York, 2010, pp. 141–183.

[56] I. MUTER, S. I. BIRBIL, AND G. SAHIN, *Combination of metaheuristic and exact algorithms for solving set covering-type optimization problems*, INFORMS Journal on Computing, 22 (2010), pp. 603–619.

[57] Y. NAGATA AND O. BRÄYSY, *Edge assembly-based memetic algorithm for the capacitated vehicle routing problem*, Networks, 54 (2009), pp. 205–215.

[58] Y. NAGATA AND S. KOBAYASHI, *Edge assembly crossover: A high-power genetic algorithm for the travelling salesman problem*, in Proceedings of the 7th International Conference on Genetic Algorithms, East Lansing, MI, Morgan Kaufmann, San Francisco, 1997, pp. 450–457.

[59] M. D. NELSON, K. E. NYGARD, J. H. GRIFFIN, AND W. E. SHREVE, *Implementation techniques for the vehicle routing problem*, Computers & Operations Research, 12 (1985), pp. 273–283.

[60] A. G. NIKOLAEV AND S. H. JACOBSON, *Simulated annealing*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., Springer, New York, 2010, pp. 1–39.

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

[62] H. PAESSENS, *The savings algorithm for the vehicle routing problem*, European Journal of Operational Research, 34 (1988), pp. 336–344.

[63] D. PISINGER AND S. ROPKE, *A general heuristic for vehicle routing problems*, Computers & Operations Research, 34 (2007), pp. 2403–2435.

[64] C. PRINS, *A simple and effective evolutionary algorithm for the vehicle routing problem*, Computers & Operations Research, 31 (2004), pp. 1985–2002.

[65] ——, *A GRASP × evolutionary local search hybrid for the vehicle routing problem*, in Bio-Inspired Algorithms for the Vehicle Routing Problem, F. Pereira and J. Tavares, eds., Springer, Berlin, Heidelberg, 2009, pp. 35–53.

[66] M. REIMANN, K. F. DOERNER, AND R. F. HARTL, *D-Ants: Savings based ants divide and conquer the vehicle routing problem*, Computers & Operations Research, 31 (2004), pp. 563–591.

[67] J. RENAUD, F. F. BOCTOR, AND G. LAPORTE, *An improved petal heuristic for the vehicle routing problem*, Journal of the Operational Research Society, 47 (1996), pp. 329–336.

[68] M. G. C. RESENDE AND C. C. RIBEIRO, *Greedy randomized adaptive search procedures: Advances, hybridizations, and applications*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., Springer, New York, 2010, pp. 283–319.

[69] M. G. C. RESENDE, C. C. RIBEIRO, F. GLOVER, AND R. MARTÍ, *Scatter search and path-relinking: Fundamentals, advances, and applications*, in Handbook of Metaheuristics, M. Gendreau and J.-Y. Potvin, eds., Springer, New York, 2010, pp. 87–107.

[70] Y. ROCHAT AND É. D. TAILLARD, *Probabilistic diversification and intensification in local search for vehicle routing*, Journal of Heuristics, 1 (1995), pp. 147–167.

[71] S. ROPKE AND D. PISINGER, *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows*, Transportation Science, 40 (2006), pp. 455–472.

[72] D. M. RYAN, C. HJORRING, AND F. GLOVER, *Extensions of the petal method for vehicle routeing*, Journal of the Operational Research Society, 44 (1993), pp. 289–296.

[73] H. SANTOS, L. OCHI, E. MARINHO, AND L. DRUMMOND, *Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem*, Neurocomputing, 70 (2006), pp. 70–77.

[74] V. I. SARVANOV AND N. N. DOROSHKO, *The approximate solution of the travelling salesman problem by a local algorithm with scanning neighborhoods of factorial cardinality in cubic time (in Russian)*, in Software: Algorithms and Programs 31, Mathematical Institute of the Belarusian Academy of Sciences, Minsk, 1981, pp. 11–13.

[75] P. SHAW, *A new local search algorithm providing high quality solutions to vehicle routing problems*, Technical report, University of Strathclyde, Glasgow, 1997.

[76] A. SUBRAMANIAN, E. UCHOA, AND L. S. OCHI, *A hybrid algorithm for a class of vehicle routing problems*, Computers & Operations Research, 40 (2013), pp. 2519–2531.

[77] É. D. TAILLARD, *Parallel iterative search methods for vehicle routing problems*, Networks, 23 (1993), pp. 661–673.

[78] É. D. TAILLARD, L. M. GAMBARDELLA, M. GENDREAU, AND J.-Y. POTVIN, *Adaptive memory programming: A unified view of metaheuristics*, European Journal of Operational Research, 135 (2001), pp. 1–16.

[79] C. D. TARANTILIS, *Solving the vehicle routing problem with adaptive memory programming methodology*, Computers & Operations Research, 32 (2005), pp. 2309–2327.

[80] C. D. TARANTILIS, A. K. ANAGNOSTOPOULOU, AND P. P. REPOUSSIS, *Adaptive path relinking for vehicle routing and scheduling problems with product returns*, Transportation Science, 47 (2013), pp. 356–379.

[81] P. M. THOMPSON AND H. N. PSARAFTIS, *Cyclic transfer algorithms for multivehicle routing and scheduling problems*, Operations Research, 41 (1993), pp. 935–946.

[82] P. TOTH AND A. TRAMONTANI, *An integer linear programming local search for capacitated vehicle routing problems*, in The Vehicle Routing Problem: Latest Advances and New Challenges, B. L. Golden, S. Raghavan, and E. A. Wasil, eds., Springer, New York, 2008, pp. 275–295.

[83] P. TOTH AND D. VIGO, eds., *The Vehicle Routing Problem*, SIAM, Philadelphia, 2002.

[84] ——, *The granular tabu search and its application to the vehicle-routing problem*, INFORMS Journal on Computing, 15 (2003), pp. 333–346.

[85] T. VIDAL, T. G. CRAINIC, M. GENDREAU, N. LAHRICHI, AND W. REI, *A hybrid genetic algorithm for multidepot and periodic vehicle routing problems*, Operations Research, 60 (2012), pp. 611–624.

[86] T. VIDAL, T. G. CRAINIC, M. GENDREAU, AND C. PRINS, *A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows*, Computers & Operations Research, 40 (2013), pp. 475–489.

[87] ——, *Heuristics for multi-attribute vehicle routing problems: A survey and synthesis*, European Journal of Operational Research, 231 (2013), pp. 1–21.

[88] ——, *A unified solution framework for multi-attribute vehicle routing problems*, European Journal of Operational Research, 234 (2014), pp. 658–673.

[89] C. VOUDOURIS AND E. TSANG, *Guided local search and its application to the traveling salesman problem*, European Journal of Operational Research, 113 (1999), pp. 469–499.

[90] C. WALSHAW, *A multilevel approach to the travelling salesman problem*, Operations Research, 50 (2002), pp. 862–877.

[91] P. WARK AND J. HOLT, *A repeated matching heuristic for the vehicle routeing problem*, Journal of the Operational Research Society, 45 (1994), pp. 1156–1167.

[92] E. E. ZACHARIADIS AND C. T. KIRANOUDIS, *A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem*, Computers & Operations Research, 37 (2010), pp. 2089–2105.