INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

The Granular Tabu Search and Its Application to the Vehicle-Routing Problem

Paolo Toth, Daniele Vigo,

To cite this article:
Paolo Toth, Daniele Vigo, (2003) The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. INFORMS Journal on Computing 15(4):333-346. http://dx.doi.org/10.1287/ijoc.15.4.333.24890

The Granular Tabu Search and Its Application to the Vehicle-Routing Problem

Paolo Toth • Daniele Vigo
Dipartimento di Elettronica, Informatica e Sistemistica,
Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
ptoth@deis.unibo.it • dvigo@deis.unibo.it

We describe a new variant, called *granular tabu search*, of the well-known tabu-search approach. The method uses an effective intensification/diversification tool that can be successfully applied to a wide class of graph-theoretic and combinatorial-optimization problems. Granular tabu search is based on the use of drastically restricted neighborhoods, not containing the moves that involve only elements that are not likely to belong to good feasible solutions. These restricted neighborhoods are called *granular*, and may be seen as an efficient implementation of candidate-list strategies proposed for tabu-search algorithms. Results of computational testing of the proposed approach on the well-known symmetric capacitated and distance-constrained vehicle-routing problem are discussed, showing that the approach is able to determine very good solutions within short computing times.
(*Heuristic Algorithms; Tabu Search; Capacitated Vehicle-Routing Problem; Distance-Constrained Vehicle-Routing Problem*)

1. Introduction

Tabu search is one of the most widely used and effective heuristic approaches available for the solution of optimization problems. It was first proposed by Glover (1989) and since then hundreds of applications to a wide variety of continuous and combinatorial optimization problems have shown its versatility and effectiveness (see Glover 1996 and Glover and Laguna 1997 for extensive surveys on tabu-search applications).

Tabu search belongs to the wide family of heuristic-search methods that have been proposed in the last two decades, known as *metaheuristics*. This family also includes other famous general heuristic paradigms such as simulated annealing and genetic algorithms. For a general introduction to metaheuristics and a presentation of the basic characteristics of the most important paradigms we refer to Aarts and Lenstra (1997).

One of the main ingredients of the success of metaheuristics, and in particular of tabu search, is certainly the simplicity of their basic structure. These methods extend and improve the *local-search* techniques that were widely used since the early 1960s to improve the solutions found by other heuristics.

Local-search algorithms start from a (generally) feasible solution and iteratively move to a new (and better) solution by selecting it from a *neighborhood* of the current solution. The neighborhood of a solution is the set of other solutions that can be obtained from it by means of simple modifications. The modification that allows one to pass from one solution to a neighboring one is the neighborhood structure, and the transition from one solution to another is called a

0899-1499/03/1504/0333
1526-5528 electronic ISSN

move. The *cardinality* of a neighborhood is the number of moves that are neighbors of a generic solution.

At each iteration, the best solution of the neighborhood that improves the current one is selected as the new current solution and the process is iterated until no improving move exists, i.e., until the current solution is a *local optimum* with respect to the current neighborhood. The time required by each iteration of a local-search algorithm depends on both the cardinality of the neighborhood and on the time needed to generate each solution, check its feasibility, and evaluate its cost. In most cases, the time per iteration is bounded by a polynomial function of the instance size. The number of iterations to be performed to reach the local optimum may be large and, in the worst case, generally grows exponentially with the instance size.

Since the quality of the local optimum found can be very poor with respect to the optimal solution, tabu search, as well as other metaheuristics based on local search, adopts a specific strategy to escape from the local optima and continue the search toward possibly-better solutions. At each iteration, the best move of the neighborhood is always performed even if it is non-improving, thus allowing one to move away from the local optimum once it has been reached. However, to avoid the occurrence of loops resulting from the movement back to the local optimum at the next iteration, a short-term memory mechanism is coupled with this extended-search strategy. The basic way of introducing such short-term memory is the so-called *tabu list*, which stores some *attributes* identifying the moves that produced the solutions most recently visited. During examination of the current neighborhood all the moves that have attributes equal to those stored in the tabu list are considered tabu and discarded. Normally, tabu search is stopped after a given number of overall or non-improving iterations.

Although better than descent local-search algorithms, the above-described simple tabu-search algorithm may need too many iterations before solutions of acceptable quality are detected. Therefore, practical tabu-search implementations normally include several additional features. For a complete explanation of all the ingredients of tabu-search algorithms we refer to Glover and Laguna (1997).

One of the results of the evolution made by practical tabu search algorithms is that they easily lose the appealing simplicity of the version that only uses a compact short-term memory design, paying for increased effectiveness with considerable complexity. This sometimes results in the presence of a high number of parameters, for which good values have to be detected experimentally (see Golden et al. 1998).

The emphasis in the development and testing of heuristic algorithms for combinatorial optimization problems is often on the quality of the solution obtained rather than the trade-off between the quality of the solution and the computing time required to obtain it. In fact, tabu-search algorithms are the current champions for the solution of many hard optimization problems. However, the computing times needed to detect the best solutions are generally very large when compared to traditional methods, like constructive heuristics, which require relatively short computing times.

In this paper we examine a general and effective search-intensification tool that may be used with tabu search as well as with other heuristic and metaheuristic algorithms, and that can be applied to a wide class of graph-theoretic and optimization problems. The method is based on use of drastically restricted neighborhoods, called *granular neighborhoods*, obtained from standard ones by removing moves that involve only elements that are not likely to belong to high-quality feasible solutions. Therefore, granular neighborhoods lead to less myopic searches in which only potentially promising moves are actually evaluated at each iteration.

Within the general settings of tabu search, granular neighborhoods may be seen as an implementation of *candidate-list* strategies (see Glover and Laguna 1997). In fact, the search is restricted to a set of elite neighboring solutions, and the criterion used to select them is fixed in advance. Similar ideas have been used by various authors to speed up local-search algorithms. For example, Johnson and McGeoch (1997) described the use of *neighbor lists* within 2-opt and 3-opt algorithms for the traveling salesman problem.

Since granular neighborhoods may be examined in considerably less time than complete ones, at each iteration it is possible to search a much larger multiple

neighborhood, defined as the union of different granular neighborhoods. In this way the time required to reach high-quality solutions is often much smaller than that required when the corresponding complete neighborhoods are used. Finally, the structure of a multiple granular neighborhood may be dynamically varied during the evolution of the algorithm to diversify the search.

To improve clarity of exposition, in this paper we describe in detail the *granular tabu search* (GTS) method by applying it to a specific problem: the well-known symmetric *vehicle-routing problem* (VRP) with vehicle-capacity and route-maximum-length constraints, which is one of the most important and difficult combinatorial optimization problems. However, as previously mentioned, granular neighborhoods may be applied to a wide variety of problems such as the traveling salesman problem and other routing, sequencing, and network-design problems in which the solution cost is defined as the sum of the costs of the selected elements. Toth and Vigo (1997) presented an application of a preliminary version of GTS to constrained directed spanning trees.

The paper is organized as follows. In §2 the VRP is briefly defined and the main local-search and tabu-search issues for the VRP are discussed. The granular neighborhood approach is introduced in §3, where granular neighborhoods for the VRP are described and their efficient exploration is discussed. An overall GTS algorithm for the VRP is given in §4. Results of computational testing on several test instances from the literature are discussed in §5, showing that the proposed approach is able to determine very good solutions within computing times that are one or two orders of magnitude smaller than those required by most of the metaheuristic approaches in the literature.

2. Local Search and Metaheuristics for the VRP

The VRP is a well-known and extremely difficult combinatorial optimization problem that finds many practical applications in the design and management of distribution systems. The VRP is concerned with the determination of optimal routes used by a fleet of vehicles stationed at a central depot to serve a set of customers with known demands. In the basic version of the problem, also known as the *capacitated VRP*, all vehicles are identical, capacity restrictions for the vehicles are imposed, and the objective is to minimize the total routing cost (or length) of the routes. This problem has been extensively studied since the early 1960s and in the last few years many new heuristic and exact approaches have been presented (for updated reviews see, e.g., Laporte 1997, Toth and Vigo 1998, 2002, and Laporte et al. 2000). The practical difficulty of the VRP is witnessed by the fact that the largest problems that can be consistently solved to proven optimality by the most effective exact algorithms proposed so far contain no more than 50 customers, and larger instances may be solved only in particular cases. Hence, instances with hundreds of customers arising in practical applications may only be addressed by heuristic methods.

The VRP may be defined as the following graph-theoretic problem. Let $G = (V, A)$ be a complete graph, where $V = \{1, \ldots, n+1\}$ is the vertex set and A is the arc set. Vertices $j = 1, \ldots, n$ correspond to the customers, each with a nonnegative *demand* d_j, whereas vertex $n+1$ corresponds to the depot (with $d_{n+1} = 0$).

A nonnegative *cost* c_{ij} is associated with each arc $(i, j) \in A$. When $c_{ij} = c_{ji}$ for all $i, j \in V$, the problem is called a *symmetric* VRP (simply denoted VRP in the following). The cost matrix satisfies the *triangle inequality* if and only if $c_{ik} + c_{kj} \geq c_{ij}$ for all $i, j, k \in V$. Most of the papers presenting heuristic approaches for the VRP consider the case in which the vertices are associated with given points of the plane and the cost c_{ij}, for each arc $(i, j) \in A$, is defined as the Euclidean distance between the two points corresponding to vertices i and j. The corresponding cost matrix is symmetric and satisfies the triangle inequality, and the resulting problem is called a *Euclidean* VRP.

A set of K identical vehicles, each with capacity C, is available at the depot. Each vehicle may perform at most one route.

The VRP then consists of finding a collection of exactly K simple *circuits* (i.e., the vehicle routes) with minimum total cost, defined as the sum of the costs

of the corresponding arcs, and such that:

(i) each circuit visits the depot vertex;

(ii) each customer vertex is visited once and by exactly one circuit;

(iii) the sum of the demands of the vertices visited by a circuit does not exceed the vehicle capacity C.

Some papers consider a variant of the VRP known as the *distance-constrained VRP* (DVRP), where each vertex is also associated with a nonnegative *service time* q_i, and for each route the total length of the arcs plus the service times of the served customers may not exceed a given maximum length L. In the DVRP instances proposed in the literature, the length of each arc is equal to its cost, and service times of customers are either all zero or all equal to a common value $\bar{q}$ (i.e., $q_i = \bar{q}$ for $i = 1, \ldots, n$).

Local search algorithms for the VRP generally use simple neighborhoods based on arc exchanges or customer movements, such as those used for other graph-theoretic problems. In particular, given the current solution s, a *k-exchange* neighborhood is made up of all the moves obtained by removing k arcs used in s and replacing them with k other arcs that define a new solution. Since the cardinality of a k-exchange neighborhood is $O(n^k)$, to limit the computing time required for examination of the neighborhood, only small values for k (i.e., $k = 2, 3$) are used in practical local search algorithms.

Figures 1–3 show examples of exchange neighborhoods with $k \leq 4$ that are typically used in local search methods for the VRP. In these figures, π_i and σ_i denote the index of the vertex preceding and following a given vertex $i \in V$ in the current solution, respectively. Figure 1 illustrates a *two exchange*. Figure 2 illustrates two types of *three exchanges*. The first one is also known as *customer insertion*, since it removes a customer from its current position and reinserts it in a different position in the same or in another route. Or (1976) generalized the insertion move by considering insertion of two and three consecutive customers, as the two-customers *Or exchange* shown in Figure 2(b). Finally, Figure 3 shows a particular type of *four exchange*, known as a *swap exchange*, where two pairs of consecutive arcs are removed.

It is easy to see that the cardinality of all the above neighborhoods is $O(n^2)$; in fact, although they may involve more than two arcs at a time, one or two of these arcs are implicitly defined after the first two arcs to be removed are chosen. The feasibility and cost of the new solutions associated with each of the above moves may be evaluated in constant time (see, e.g., Kindervater and Savelsbergh 1997 and Vigo 1996). Therefore, the complete exploration of these basic neighborhoods can be performed in $O(n^2)$ time.

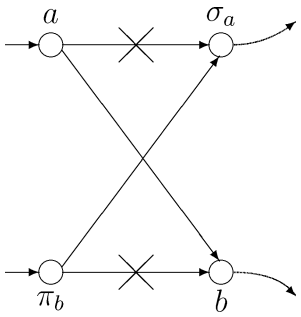

Figure 1 A Two Exchange.
Note. Removed arcs are marked with a cross.

More complex neighborhoods, based on ejection chains or on network flow, are presented in Rego and Roucairol (1996), and in Xu and Kelly (1996). The reader is referred to Gendreau et al. (1997), Golden et al. (1998), and Laporte et al. (2000) for a thorough analysis of metaheuristic approaches proposed for the VRP and of their performances.

Metaheuristic approaches, particularly those based on tabu search, are able to determine high-quality solutions for the VRP. For example, when considering the widely used set of fourteen VRP and DVRP test problems described in Christofides and Eilon (1969), and in Christofides et al. (1979), most of the best known solutions were determined by tabu-search algorithms. Moreover, the average percentage deviation of the solutions found by the most recent algorithms, with respect to the best known solutions, is generally smaller than 1% (see Golden et al. 1998).

Table 1 summarizes computing times required by tabu-search algorithms when applied to the 14 benchmark VRP and DVRP instances, or to a subset of them. Observe that the times reported for the algorithms by Taillard (1993) and Rochat and Taillard (1995) are those needed to reach solutions within 1% of the best ones.

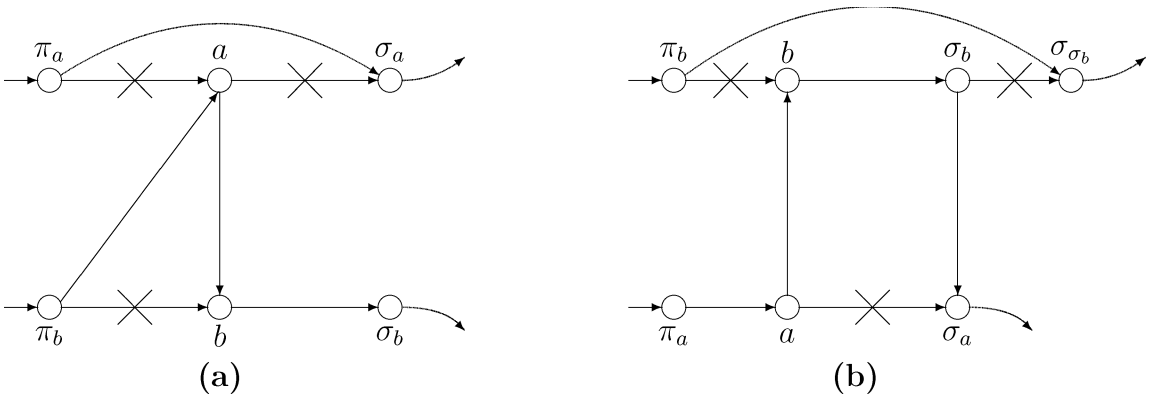

Figure 2 Three Exchanges: (a) Customer Insertion, (b) Two-Customer or Exchange
Note. Removed arcs are marked with a cross.

It is clearly difficult to compare the performance of algorithms whose testing was done using different computers, particularly for non-numerical applications such as tabu-search algorithms. However, a very crude indication of the machine relative speed may be derived from the Mflops measure reported in Table 1, and obtained by Dongarra (2001) by using as benchmark a classic numerical application: the solution of a linear system. Within this simplified setting, the Mflops ratios may be used to convert CPU times of different machines. In our computational testing we used a Pentium 200 MHz CPU (15 Mflops, according to Dongarra 2001), and we include in Table 1 the raw "conversion factor" ρ, which is the multiplicative constant to transform the CPU time of a given machine into Pentium 200 MHz CPU time.

As observed by Golden et al. (1998), for these tabu-search algorithms the growth rate of computing time with respect to problem size is about quadratic, and this leads to time requirements of thousands of minutes on fast workstations to solve problems with realistic size (i.e., with 300 to 500 customers). These computing times are generally not compatible with most real-world applications of vehicle-routing optimization, particularly in the on-line or semi-on-line context.

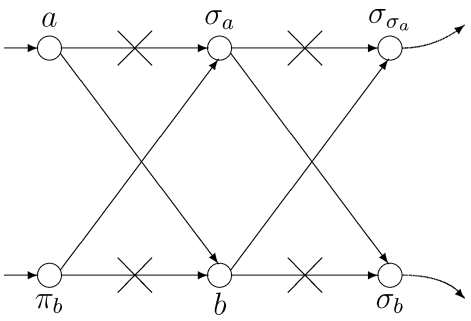

Figure 3 A Particular Four Exchange: The Customer Swap
Note. Removed arcs are marked with a cross.

3. Granular Neighborhoods

One of the reasons for the large computing-time requirements of tabu-search approaches for the VRP is that these methods normally need to perform several thousand iterations to obtain high-quality solutions. Each iteration generally consists of the exploration of the above-mentioned exchange neighborhoods and requires at least $O(n^2)$ time.

Several approaches have been proposed in the literature to reduce the computing time of neighborhood exploration within local search, such as random sampling, candidate-list strategies, and parallel implementations of the search. We defined an effective implementation of the candidate-list strategy, which leads to drastic reductions in the computing time required by each tabu-search iteration. This objective is reached by using neighborhoods that can be examined in much less time than can the traditional ones, but without considerably affecting the quality of the solutions found. To this end, we propose to derive *granular neighborhoods* as restrictions of other known neighborhoods, by discarding a large quantity of unpromising moves and actually exploring only a

Table 1 Comparison of the Computing Times, in Minutes of Different CPUs, Required by Tabu-Search Algorithms When Applied to Classic Euclidean VRP and DVRP Benchmark Instances

	Time (minutes)				
Reference	Max	Average	CPU Used	Mflops	ρ[5]
Taillard (1993)[1]	65	22	SG 4D/35	4.3[2]	0.30
Rochat and Taillard (1995)[1]	45	16	SG Indigo 100MHz	15[2]	1.00
Gendreau et al. (1994)	100	45	SG 36MHz	5.7[3]	0.40
Rego and Roucairol (1996)	52	15	Sun Sparc4 IPC	2.5–4.3[2]	0.25
Xu and Kelly (1996)	368	130	DEC Alpha	26.0–43.0[2,4]	2.00

[1]Time required to reach solutions within 1% from the best known ones.
[2]According to Dongarra (2001).
[3]According to Gendreau et al. (1994).
[4]Five times faster than SG 36MHz according to Golden et al. (1998).
[5]Raw CPU time conversion factor relative to a Pentium 200 MHz PC, 15 Mflops.

small subset of them, containing the most promising ones.

In our opinion, this choice has several positive aspects. First of all, it is found to be one of the least "intrusive" ways of modifying a successful solution approach while keeping its main features intact, and particularly the basic structure of the neighborhoods used within the search. Moreover, it increases applicability of the proposed method and simplifies its extension to other problems for which tabu search and other metaheuristics proved to be effective, but not efficient in terms of time requirements. Last but not least, it allows one to evaluate in a direct way the benefits of the proposed method with respect to a tabu-search algorithm that uses the same neighborhoods.

3.1. Simple Granular Neighborhoods for the VRP

The restriction that leads to the granular neighborhood's starting from the original one must be clearly performed in an efficient way. To illustrate this concept further let us consider the VRP and show how simple granular neighborhoods may be defined for this problem.

When, as is generally the case, the VRP is defined on a complete graph, it may be observed that "long" (i.e., high-cost) arcs have a small probability of being part of high-quality solutions. As an example, Table 2 considers some classic Euclidean VRP test problems. In the first five problems the customers are distributed uniformly in the plane, whereas in the last two they are clustered around uniformly distributed points. For each problem the table reports its name, the number of customers n, the number of available vehicles K, and the value of the best known solution z^*. In addition, the average cost of the arcs in the best known solution, $\bar{z}^* = z^*/(n+K)$, is compared with the minimum, maximum, and average arc cost in the complete graph.

Table 2 Comparison of the Average Arc Cost in the Best-Known Solution with Respect to the Minimum, Maximum, and Average Arc Cost in the Graph for Some Classic Euclidean VRP Instances

					Arc Cost		
Problem	n	K	z^*	$\bar{z}^*$	Min	Max	Average
E051-05e	50	5	524.61	9.54	2.24	85.63	33.75
E076-10e	75	10	835.26	9.83	2.24	85.28	34.13
E101-08e	100	8	826.14	7.65	1.41	91.83	34.64
E151-12c	150	12	1028.42	6.35	0.00	91.83	33.92
E200-17c	199	17	1291.45	5.98	0.00	91.83	33.24
E101-10c	100	10	819.56	7.45	1.00	96.18	40.27
E121-07c	120	7	1042.11	8.21	0.00	114.98	54.52

By considering the distribution of the arc costs associated with this type of test problem it may be easily seen that the large majority of the arcs have cost larger than $\bar{z}^*$. A similar behavior can be observed in almost all known test problems for VRP and DVRP. Therefore, a possible way to speed up the search of a neighborhood is to limit as much as possible the evaluation of moves that try to insert "long" arcs in the current solution.

We pursued this basic idea in developing the granular neighborhoods that we used for the VRP. Starting from the original complete graph $G = (V, A)$, we define a new sparse graph $G' = (V, A')$, with $|A'| \ll n^2$. This sparse graph includes all the arcs that should be considered for inclusion in the current solution: for example, all the "short" arcs and a relevant subset of other important arcs, such as those incident to the depot and those belonging to the best solutions encountered so far. As will be discussed later, the search of a granular neighborhood considers only the moves that are *generated* by arcs belonging to graph G', i.e., the moves that involve at least one "good" arc. Note that this does not mean that "long" arcs are never inserted in the current solution, but that moves involving only "long" arcs are not considered.

In our implementation we adopted a very simple filtering rule to define "short" arcs. An arc is *short*, hence it belongs to the sparse graph G', if its cost is not greater than the *granularity threshold* value

$$\vartheta = \beta \cdot \frac{z'}{(n+K)}, \tag{1}$$

where β is a suitable positive *sparsification parameter*, and z' is the value of a heuristic solution determined by any fast traditional heuristic: in our computational testing we used the well-known algorithm by Clarke and Wright (1964). The resulting granular neighborhood may be seen as a "scaling" of the original neighborhood induced by the sparse graph where long arcs were removed.

Our computational experience confirmed our initial assumption. A basic tabu-search algorithm (described in §4) that uses the granular version of basic neighborhoods was able to determine high-quality solutions within running times comparable to those of constructive heuristics. Figure 4 further illustrates these results by showing typical behavior, in terms of quality of the solution obtained within 10,000 iterations, of the basic tabu-search algorithm using granular neighborhoods defined by different values of the sparsification parameter β ranging from 0.5 to 5.0. Perhaps surprisingly, the solution quality is not a monotone increasing function of the sparsification parameter, i.e., of the overall computational effort. In fact, with the benchmark problems in the literature considered in §5, the best results are typically obtained with β values between 1.0 and 2.5, which select about 10–20% of the arcs of the complete graph.

In some particular cases, different filtering rules may be better suited than the simple cost-based one described above. For example, when considering highly clustered instances it may be preferable to adopt a more traditional vertex-based filtering rule that selects for each vertex the, say, h shortest arcs incident to it. Another possibility is to incorporate in the rule information provided by arc reduced costs, computed with respect to an available lower bound. In this way we may avoid including into the solution arcs that do not improve on the current best known solution.

3.2. Efficient Search and Diversification

The key factor in the granular paradigm is that the search of granular neighborhoods may be efficiently implemented, in quite a natural way, by explicitly taking into account the sparse graph $G' = (V, A')$ associated with the neighborhood. As previously mentioned, the arc set A' includes all the "short" arcs, e.g., the arcs whose cost is not greater than the granularity threshold ϑ, and a set I of other important arcs, such as those incident to the depot or belonging to high-quality solutions:

$$A' = \{(i, j) \in A : c_{ij} \leq \vartheta\} \cup I. \tag{2}$$

The arcs of A' are directly used as *move generators* to determine the other arcs involved in a particular move of the original neighborhood. As an example, let us consider the basic neighborhoods described in §2. It is easy to see that an arc $(a, b) \in A'$ induces a unique move of each of these neighborhoods, namely, that involving the arcs illustrated in Figures 1–3. In fact, in these moves, once arc (a, b) is chosen, all the other arcs involved in the move are implicitly defined. Therefore, by using any efficient data structure to store the sparse graph (e.g., biconnected chains), the total number of moves evaluated in the search of a granular neighborhood is $O(|A'|)$. Moreover, for the VRP the evaluation of the basic moves can be performed in constant time, thus leading to an overall time complexity of $O(|A'|)$ for the

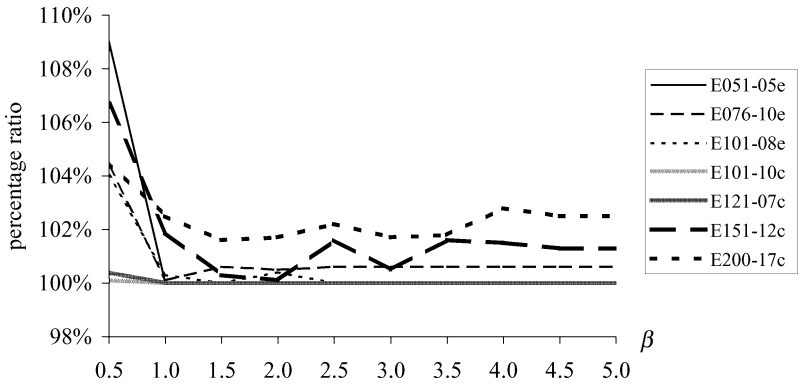

Figure 4 **Performance of a Tabu-Search Algorithm with Granular Neighborhoods and Different Values of the Sparsification Parameter β, When Applied to Classic VRP Instances**

complete search of the granular neighborhood of a solution.

Another important issue connected with the granular paradigm is that a dynamic modification of the structure of the sparse graph associated with the granular neighborhood provides a simple way of including intensification and diversification during the search. For example, by modifying the sparsification parameter β, the number of arcs currently included in the sparse graph is altered, hence a possibly different new solution is obtained at the end of the search. To this end, the tabu-search algorithm may alternate between long intensification steps, associated with a small β value, and short diversification steps in which β is considerably increased and the evaluation of the moves is possibly modified to favor the possible inclusion of new (longer) arcs in the solution.

4. A Granular Tabu-Search Algorithm for the VRP

In this section we briefly discuss a basic *granular tabu-search* (GTS) algorithm for the VRP and the DVRP. Our main aim is to test the impact of the granular neighborhood paradigm within the tabu-search approach. Therefore, we implemented a tabu-search algorithm that explores a multiple neighborhood obtained as the union of granular neighborhoods and that includes some of the main features proposed in the early work on tabu search for the VRP, such as Taillard (1993) and Gendreau et al. (1994).

The GTS algorithm is initialized with the heuristic solution obtained by the well-known savings heuristic by Clarke and Wright (1964). Since this heuristic does not guarantee determining a solution with the prescribed number of vehicles, whenever the initial solution uses more than K vehicles, the less-loaded routes are removed and the customers they contain are inserted into the best position, with respect to the routing cost, in one of the remaining K routes. Note that the resulting initial solution may be infeasible, violating the vehicle-capacity or the maximum-route-length constraints.

Visiting of infeasible solutions is allowed during the search and, as usual, their costs are modified by adding to the routing cost a penalty α_C multiplied by the sum of the excess loads of the overloaded routes, plus a penalty α_D multiplied by the sum of the excess lengths of infeasible routes with respect to the maximum-length constraint (see Gendreau et al. 1994). The values of the penalties α_C and α_D are dynamically updated during the search within the range $[\alpha_{\min}, \alpha_{\max}]$. In particular, every n_i iterations, if all the previous n_i visited solutions were feasible with respect to the capacity constraints, then α_C is set to $\max\{\alpha_{\min}, \alpha_C/2\}$, whereas if all were infeasible it is set to $\min\{\alpha_{\max}, 2\alpha_C\}$. The updating rule for α_D is analogous. In our computational testing, $\alpha_{\min} = 1$ and $\alpha_{\max} = 6400$; both α_C and α_D are initially set to 100, and $n_i = 10$.

The GTS algorithm uses a multiple granular neighborhood based on the four basic neighborhoods described in §2. The granular neighborhood is

obtained by considering the arcs in the sparse graph defined by all the arcs below the current granularity threshold, plus all arcs incident to the depot, and those belonging to the best solution found and to the current one. After each iteration, the arcs inserted by the performed move are added to the sparse graph. The sparse graph is rebuilt from scratch every $2n$ iterations. We used a sparsification factor $\beta = 1.25$, which computationally gave the best performance. However, for Euclidean VRP and DVRP instances, β values between 1 and 2 are generally appropriate.

A move is considered *tabu* if it tries to reinsert an arc removed in one of the previous moves. The tabu tenure t for each move performed is an integer uniformly distributed random variable in the interval $[t_{\min}, t_{\max}]$. We used $t_{\min} = 5$ and $t_{\max} = 10$, as in Gendreau et al. (1994).

Finally, we used the granularity-based diversification described in the previous section. Whenever the current best solution is not improved after n_d iterations, the sparsification factor is increased to β_d, a new sparse graph is determined, and n_h iterations are performed starting from the best solution found. Then, the sparsification factor is reset to the original value and the search continues. We used $n_d = 15n$, $\beta_d = 1.75$, and $n_h = n$.

When we adopted features proposed by previous work, such as dynamic updating of the infeasibility penalties and the tabu tenure definition, we used the parameter values given in the corresponding paper. The values of the remaining parameters were experimentally determined as those allowing for the best compromise between solution quality and computational effort.

5. Computational Results

The GTS algorithm was coded in FORTRAN 77 and tested on a Pentium 200 MHz PC. The computational testing considered several Euclidean VRP and DVRP instances from the literature with up to about 500 customers, which are generally used as a standard benchmark for VRP algorithms (see Golden et al. 1998).

In the following, the test instances are denoted by a name that allows one to determine their characteristics quickly. In particular, the names have the form $tnnn-kkp$, where t is the instance type, equal to 'E' for Euclidean VRPs and to 'D' for Euclidean DVRPs; nnn is the number of vertices of the corresponding graph (including the depot vertex); kk is the number of available vehicles; finally p, the last letter of the name, identifies the paper where the instance data were first given. For example, E051-05e denotes the classic 50-customer instance proposed by Christofides and Eilon (1969), and E045-04f denotes the 44-customer instance described by Fisher (1994).

Our benchmark includes the fourteen classic Euclidean VRP and DVRP instances described in Christofides and Eilon (1969) and Christofides et al. (1979), the three Euclidean VRP instances proposed by Fisher (1994), and the set of twenty large-scale Euclidean VRP and DVRP instances recently proposed by Golden et al. (1998). Table 3 summarizes the main data of the instances we considered. In particular, for each instance the table gives the number of customers (n), the number of available vehicles (K), the vehicle capacity (C) and, for DVRP instances, the route maximum length (L) and the service time (q) for each customer. The table also gives, for each instance, the reference to the paper where its data were first described, the best solution value known so far, and the paper where such a solution is reported. The last column of the table contains the values of the new best solution found during our overall testing activity. All test instance data, as well as most of the best known solution data, may be obtained from the authors.

The new instances described by Golden et al. (1998) require a couple of comments. First of all, the best solution values known so far of the instances marked with an asterisk in Table 3 differ from the best values reported in Table 15 of Golden et al. (1998). In fact, as confirmed by Xu (1998), the Xu and Kelly (1996) algorithm was run on these instances with a cost matrix obtained by ignoring the fractional part of the customer coordinates. For the marked instances the Xu and Kelly solution values reported in Golden et al. (1998) are smaller than the correct ones, which are no longer the best known values. Moreover, the available solution to instance D241-10k is infeasible since it violates some maximum-route-length constraints. Although only some of the Xu and Kelly solutions

Table 3 Summary of the Data of the VRP and DVRP Instances Used for Computational Testing

Name	n	K	C	L	$\bar{q}$	Data Source	Prev. Best Solution	Solution Source		New Best Solution
E045-04f	44	4	2010	—	—	Fisher (1994)	723.54	Fisher (1994)		
E051-05e	50	5	160	—	—	Christofides and Eilon (1969)	524.61	Gendreau et al. (1994)		
E072-04f	71	4	30000	—	—	Fisher (1994)	241.97	Fisher (1994)		
E076-10e	75	10	140	—	—	Christofides and Eilon (1969)	835.26	Taillard (1993)		
E101-08e	100	8	200	—	—	Christofides and Eilon (1969)	826.14	Taillard (1993), Gendreau et al. (1994)		
E101-10c	100	10	200	—	—	Christofides et al. (1979)	819.56	Taillard (1993)		
E121-07c	120	7	200	—	—	Christofides et al. (1979)	1042.11	Taillard (1993)		
E135-07f	134	7	2210	—	—	Fisher (1994)	1162.96	Rochat and Taillard (1995)		
E151-12c	150	12	200	—	—	Christofides et al. (1979)	1028.42	Taillard (1993)		
E200-17c	199	17	200	—	—	Christofides et al. (1979)	1291.45	Rochat and Taillard (1995)		
E241-22k	240	22	200	—	—	Golden et al. (1998)	720.44	Golden et al. (1998)	*	709.9
E253-27k	252	27	1000	—	—	Golden et al. (1998)	881.04	Golden et al. (1998)		868.7
E256-14k	255	14	1000	—	—	Golden et al. (1998)	587.09	Golden et al. (1998)		
E301-28k	300	28	200	—	—	Golden et al. (1998)	1029.21	Golden et al. (1998)	*	1014.82
E321-30k	320	30	1000	—	—	Golden et al. (1998)	1103.69	Golden et al. (1998)		1096.18
E324-16k	323	16	1000	—	—	Golden et al. (1998)	746.56	Golden et al. (1998)		
E361-33k	360	33	200	—	—	Golden et al. (1998)	1403.05	Golden et al. (1998)	*	1389.14
E397-34k	396	34	1000	—	—	Golden et al. (1998)	1364.23	Golden et al. (1998)		1363.34
E400-18k	399	18	1000	—	—	Golden et al. (1998)	932.68	Golden et al. (1998)		
E421-41k	420	41	200	—	—	Golden et al. (1998)	1875.17	Golden et al. (1998)	*	
E481-38k	480	38	1000	—	—	Golden et al. (1998)	1656.66	Golden et al. (1998)		1650.42
E484-19k	483	19	1000	—	—	Golden et al. (1998)	1137.18	Golden et al. (1998)		1136.05
D051-06c	50	6	160	200	10	Christofides et al. (1979)	555.43	Taillard (1993), Gendreau et al. (1994)		
D076-11c	75	11	140	160	10	Christofides et al. (1979)	909.68	Taillard (1993)		
D101-09c	100	9	200	230	10	Christofides et al. (1979)	865.94	Taillard (1993), Gendreau et al. (1994)		
D101-11c	100	11	200	1040	90	Christofides et al. (1979)	866.35	Osman (1993)		
D121-11c	120	11	200	720	50	Christofides et al. (1979)	1541.14	Taillard (1993)		
D151-14c	150	14	200	200	10	Christofides et al. (1979)	1162.55	Taillard (1993)		
D200-18c	199	18	200	200	10	Christofides et al. (1979)	1395.85	Rochat and Taillard (1995)		
D201-05k	200	5	900	1800	0	Golden et al. (1998)	6702.73	Golden et al. (1998)		6680.36
D241-10k	240	10	550	650	0	Golden et al. (1998)	5834.60	Golden et al. (1998)	*	5736.15
D281-08k	280	8	900	1500	0	Golden et al. (1998)	9016.93	Golden et al. (1998)		8712.76
D321-10k	320	10	700	900	0	Golden et al. (1998)	8566.04	Golden et al. (1998)	+	8553.03
D361-09k	360	9	900	1300	0	Golden et al. (1998)	11047.69	Golden et al. (1998)	+	10515.33
D401-10k	400	10	900	1200	0	Golden et al. (1998)	11649.06	Golden et al. (1998)	+	11402.75
D441-11k	440	11	900	1200	0	Golden et al. (1998)	12250.06	Golden et al. (1998)	+	12036.24
D481-12k	480	12	1000	1600	0	Golden et al. (1998)	14639.32	Golden et al. (1998)		14336.36

for the new instances are available, namely, those for which the algorithm gave the best solution, it is likely that the solution values reported in the XK column of Table 15 in Golden et al. (1998) are incorrect for all the instances in which the customer coordinates have a fractional part. These are all the eight DVRP instances and the VRP instances denoted by E241-22k, E301-28k, E361-33k and E421-41k. As a consequence, the best solution values reported in our Table 3 for all the new instances with fractional customer coordinates were determined from Table 15 of Golden et al. (1998) by ignoring the results of the Xu and Kelly algorithm.

As a second comment, we observe that the solution values marked by a plus sign in Table 3 were not determined by any algorithm, but are relative to hand-made solutions, constructed by considering the particular geometric structure of the instance.

The aim of our computational testing is to evaluate the behavior of the GTS algorithm in terms of tradeoff between solution quality and overall computing time required. To this end, for each set of test instances, we compare the results of the GTS algorithm with those of the known tabu-search algorithms that were tested on the same instances, for which computing times are available, and which presented the best performance in terms of solution quality or required computing time. Observe that in the first three tables of computational results we have not considered the tabu-search algorithms by Taillard (1993) or Rochat and Taillard (1995). In fact, although for many test problems these algorithms were able to determine the best known solutions, their overall computing time is not reported. However, these approaches will be considered in the last part of this section where the time required to reach a prescribed solution quality is addressed.

Table 4 compares the results obtained on the fourteen classic VRP and DVRP test instances by GTS (columns marked GTS), with those of the tabu-search algorithms by Gendreau et al. (1994), Xu and Kelly (1996), and Rego and Roucairol (1996) (columns marked GHL, XK, and RR, respectively). Table 5 considers the three real-world Euclidean VRP instances described by Fisher (1994) and compares the results obtained by GTS with those of the tabu-search algorithm by Xu and Kelly (1996) (XK). For each instance the tables report the percentage ratio of the solution value obtained by each algorithm with respect to the best known solution value, as well as the computing time expressed in minutes. The CPU used by the different authors, the relative speed expressed in Mflops, and the ρ factor to convert the computing times approximately into Pentium 200 MHz CPU times are reported in Table 1.

The results of the XK algorithm for the DVRP instances are those reported in Golden et al. (1998), since Xu and Kelly (1996) had not tested their algorithm on DVRP instances.

On these widely used test instances, GTS proved able to determine, in a quite-short computing time, very good solutions whose quality is comparable to or better than the solutions obtained by other tabu-search approaches from the literature. In particular,

Table 4 Comparison of the Results of Tabu-Search Algorithms on the Fourteen Classic VRP and DVRP Test Instances

	Best	GTS			GHL		XK		RR	
Instance	Sol.	Sol.	%	Time	%	Time	%	Time	%	Time
E051-05e	524.61	524.61	100.00	0.81	100.00	6.00	100.00	29.92	100.00	0.85
E076-10e	835.26	838.60	100.40	2.21	100.06	53.80	100.00	48.80	100.27	16.80
E101-08e	826.14	828.56	100.29	2.39	100.40	18.40	100.00	71.93	100.17	33.90
E101-10c	819.56	819.56	100.00	1.10	100.00	16.00	100.00	56.61	100.00	1.22
E121-07c	1042.11	1042.87	100.07	3.18	103.01	22.20	100.00	91.23	100.14	6.30
E151-12c	1028.42	1033.21	100.47	4.51	100.75	58.80	100.11	149.90	102.52	27.20
E200-17c	1291.45	1318.25	102.08	7.50	102.42	90.90	100.55	272.52	103.64	16.25
VRP average			100.47	3.10	100.95	38.01	100.09	102.99	100.96	14.65
D051-06c	555.43	555.43	100.00	0.86	100.00	13.50	100.00	30.67	100.00	3.17
D076-11c	909.68	920.72	101.21	2.75	100.39	54.60	106.15	102.13	100.00	23.10
D101-09c	865.94	869.48	100.41	2.90	100.00	25.60	101.78	98.15	100.27	8.60
D101-11c	866.37	866.37	100.00	1.41	100.00	65.70	105.64	152.98	100.02	9.42
D121-11c	1541.14	1545.51	100.28	9.34	102.12	59.20	105.02	201.75	100.59	2.00
D151-14c	1162.55	1173.12	100.91	5.67	101.31	71.00	—	168.08	101.40	15.55
D200-18c	1395.85	1435.74	102.86	9.11	101.62	99.80	103.11	368.37	101.79	52.02
DVRP average			100.81	4.58	100.78	55.63	103.62	160.30	100.58	16.26
Overall average			100.64	3.84	100.86	46.82	101.72	131.65	100.77	15.45

Note. Computing times are expressed in minutes of different CPUs.

Table 5 Comparison of the Results of Tabu-Search Algorithms on the Test Instances Proposed by Fisher (1994)

Instance	Best Sol.	GTS Sol.	GTS %	GTS Time	XK %	XK Time
E045-04f	723.54	727.75	100.58	0.67	100.00	1.43
E072-04f	241.97	241.97	100.00	0.60	101.06	86.65
E135-07f	1162.96	1165.88	100.25	10.64	101.18	191.46
Average			100.28	3.97	100.75	93.18

Note. Computing times are expressed in minutes of different CPUs.

the GTS solution is worse than the GHL one only in one out of the seven VRP instances, and the gap with respect to the best known solutions of GTS is half that of GHL. For the DVRP instances, GHL is strictly better than GTS in three out of seven cases, and the average percentage ratios for the two algorithms are almost the same. The computing times of the GTS algorithm are on average about one fifth the equivalent computing times of GHL.

As to the comparison between GTS and XK on the seven classic VRP instances, XK generally obtains better solutions, whereas GTS is considerably more effective than XK on the DVRP and the Fisher's VRP instances. The computing times of the GTS algorithm are on average 50–70 times smaller than the equivalent computing times of XK.

Finally, the performance of GTS is quite similar to that of algorithm RR both in terms of solution quality and overall computing times. In particular, GTS performs slightly better on VRP instances and slightly worse on DVRP ones and the average equivalent computing times are almost the same.

Table 6 considers the twenty new large-scale VRP and DVRP instances proposed by Golden et al. (1998). In this case, the results of GTS are compared with

Table 6 Comparison of the Results on the Test Instances Proposed by Golden et al. (1998)

Instance	Best Sol.	GTS Sol.	GTS %	GTS Time	XK %	XK Time	RTR %	RTR Time
E241-22k	720.44	711.07	98.70	14.29	103.72	2314.00	100.00	5.69
E253-27k	881.04	868.80	98.61	11.43	100.00	1465.77	100.00	6.01
E256-14k	587.09	593.35	101.07	11.67	100.34	340.20	100.00	23.01
E301-28k	1029.21	1016.83	98.80	21.45	103.63	4101.02	100.00	8.15
E321-30k	1103.69	1096.18	99.32	14.51	101.30	1577.30	100.00	21.83
E324-16k	746.56	751.66	100.68	15.83	100.00	501.82	100.35	31.49
E361-33k	1403.05	1400.96	99.85	30.06	102.34	5718.38	100.00	12.42
E397-34k	1364.23	1369.44	100.38	18.45	100.99	4340.07	100.00	32.62
E400-18k	932.68	936.04	100.36	33.12	100.00	852.72	100.18	69.19
E421-41k	1875.17	1915.83	102.17	43.05	103.19	10839.73	100.00	31.05
E481-38k	1656.66	1652.32	99.74	23.07	100.00	8943.45	100.08	47.55
E484-19k	1137.18	1147.14	100.88	42.90	100.31	1151.10	100.00	101.09
VRP average			100.05	23.32	101.32	3512.13	100.05	32.51
D201-05k	6702.73	6697.53	99.92	2.38	—	591.40	100.00	11.24
D241-10k	5834.60	5736.15	98.31	4.98	—	802.87	100.00	3.68
D281-08k	9016.93	8963.32	99.41	4.65	—	913.70	100.00	18.79
D321-10k	8566.04	8553.03	99.85	8.28	—	898.53	105.09	22.66
D361-09k	11047.69	10547.44	95.47	11.66	—	1062.73	101.50	22.55
D401-10k	11649.06	11402.75	97.89	12.94	—	1749.27	101.98	40.04
D441-11k	12250.06	12036.24	98.25	11.08	—	1586.20	102.16	111.37
D481-12k	14639.32	14910.62	101.85	15.13	—	2432.42	100.00	122.61
DVRP average			98.87	8.89	—	1254.64	101.34	44.12
Overall average			99.58	17.55	101.32	2609.13	100.57	37.15

Note. Computing times are expressed in minutes of different CPUs.

those of the Xu and Kelly (1996) algorithm (XK) and with those of a deterministic annealing heuristic, called RTR, described in Golden et al. (1998). Algorithm RTR was run on a Pentium 100 MHz PC (about 12 Mflops, see Dongarra 2001, leading to $\rho = 0.8$). On these instances, GTS determined 13 new best solutions, improving on average 1% the solutions of RTR. (During the overall testing activity we actually detected the 16 new best solutions reported in Table 3.) As to the computing times, these are of the same order of magnitude as those of the RTR algorithm and about 300 times smaller than those of XK.

When run with the complete neighborhoods, the tabu search used as a basis for the GTS algorithm obtained solutions on average slightly better than those of GTS, but required computing times of the same order of magnitude as the XK and GHL algorithms. This clearly illustrates the positive impact of candidate-list strategies within the local search methods. On the one hand, granular neighborhoods may be easily introduced into any existing local search algorithm and lead to drastic reductions of the computational effort: The GTS algorithm is several orders of magnitude faster than a basic tabu search using complete neighborhoods, and requires on large-scale instances about the same amount of computing times as effective constructive heuristics. On the other hand, the introduction of granular neighborhoods does not substantially affect the overall efficacy of the approach: the quality of the solutions obtained by GTS is comparable to those of the best available algorithms.

In addition, GTS preserves the typical behavior of tabu-search algorithms in which there is a very quick initial improvement of the solution quality with respect to the starting solution. To this end, Table 7 compares, for the seven classic VRP instances, the time, in minutes, required to reach a prescribed solution quality of the GTS algorithm with those of the tabu-search algorithms by Taillard (1993) (T), Rochat and Taillard (1995) (RT), and Xu and Kelly (1996) (XK). Note that the data of Table 7 relative to the T, RT and XK algorithms are obtained from the corresponding papers. Hence, the best known solution values used as a reference to determine the relative error of the heuristic solutions may be slightly different for each paper.

Table 7 Time, in Minutes of Different CPUs, to Reach a Prescribed Solution Quality

	GTS		T		RT		XK	
Instance	5%	1%	5%	1%	5%	1%	5%	1%
E051-05e	0.00	0.02	0.12	0.82	0.05	0.18	0.65	0.77
E076-10e	0.00	0.79	0.05	0.88	0.03	1.13	0.05	6.72
E101-08e	0.02	1.05	0.20	9.67	0.25	15.00	0.15	2.38
E101-10c	0.00	0.01	1.35	5.67	0.85	5.83	0.09	2.39
E121-07c	0.00	0.01	76.67	—	11.50	45.00	0.52	1.18
E151-12c	0.05	2.12	1.43	63.33	0.42	30.00	4.09	38.41
E200-17c	0.55	—	1.25	50.00	1.48	—	6.22	201.18
Average	0.09	0.67	11.58	21.73	2.08	16.19	1.68	36.15

Table 7 shows that GTS obtains solutions that are within 5% of the best known ones in a few seconds, whereas solutions that are 1% from the best ones are generally obtained within 10–25% of the overall computing time. To obtain solutions of comparable quality the other tabu-search algorithms in the literature require computing times that are at least one order of magnitude larger.

6. Conclusions

In this paper we presented and tested an effective implementation of candidate-list strategies to be used within tabu-search algorithms for a wide class of graph-theoretic and combinatorial optimization problems.

The proposed approach, called *granular tabu search*, was applied to the well-known symmetric capacitated and distance-constrained vehicle-routing problem for which several tabu-search algorithms have been presented in the literature. These algorithms are able to obtain high-quality solutions but often require a large amount of computing time to solve large instances.

Granular tabu search is based on the use of *granular* neighborhoods, which include a small number of "promising" moves. We proposed a simple strategy to obtain granular neighborhoods from standard ones and discussed their efficient search.

Our computational testing shows the importance of using appropriate candidate-list strategies and their impact in creating better methods: on standard test instances from the literature, granular tabu search is

able to determine very good solutions within short computing times.

Acknowledgments

This research was supported by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and by Consiglio Nazionale delle Ricerche (CNR), Italy.

The authors are indebted to Mauro Dell'Amico, who suggested the name *granular neighborhoods*, and to Fred Glover, Eric Taillard, the Associate Editor, and three anonymous referees for their thorough comments. Thanks are also due to Andrea Cenni for his help in programming and to the Laboratory of Operations Research at D.E.I.S. for support in computational testing.

References

Aarts, E. H., J. K. Lenstra, eds. 1997. *Local Search in Combinatorial Optimization*. Wiley, Chichester, UK.

Christofides, N., S. Eilon. 1969. An algorithm for the vehicle dispatching problem. *Oper. Res. Quart.* **20** 309–318.

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.

Clarke, G., J. W. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **12** 568–581.

Dongarra, J. J. 2001. Performance of various computers using standard linear equations software. Technical report CS-89-85, Computer Science Department, University of Tennessee, Knoxville, TN.

Fisher, M. L. 1994. Optimal solution of vehicle routing problems using minimum *k*-trees. *Oper. Res.* **42** 626–642.

Gendreau, M., A. Hertz, G. Laporte. 1994. A tabu search heuristic for the vehicle routing problem. *Management Sci.* **40** 1276–1290.

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

Glover, F. 1989. Tabu search—Part I. *ORSA J. Comput.* **1** 190–206.

Glover, F. 1996. Tabu search and adaptive memory programming—Advances, applications and challenges. R. S. Barr, R. V. Helagson, J. L. Kennington, eds. *Interfaces in Computer Science and Operations Research*. Kluwer, Boston, MA, 1–75.

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

Golden, B. L., E. A. Wasil, J. P. Kelly, I. 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, Boston, MA, 33–56.

Johnson, D. S., L. A. McGeoch. 1997. The traveling salesman problem: A case study. E. H. Aarts, J. K. Lenstra, eds. *Local Search in Combinatorial Optimization*. Wiley, Chichester, UK, 215–310.

Kindervater, G. A. P., M. W. P. Savelsbergh. 1997. Vehicle routing: Handling edge exchanges. E. H. Aarts, J. K. Lenstra, eds. *Local Search in Combinatorial Optimization*. Wiley, Chichester, UK, 337–360.

Laporte, G. 1997. Vehicle routing. M. Dell'Amico, F. Maffioli, S. Martello, eds. *Annotated Bibliographies in Combinatorial Optimization*. Wiley, Chichester, UK, 223–240.

Laporte, G., M. Gendreau, J-Y. Potvin, F. Semet. 2000. Classical and modern heuristics for the vehicle routing problem. *Internat. Trans. Oper. Res.* **7** 285–300.

Or, I. 1976. Traveling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking. Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Osman, I. H. 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. Oper. Res.* **41** 421–451.

Rego, C., C. Roucairol. 1996. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. I. H. Osman, J. P. Kelly, eds. *Meta-heuristics: Theory & Applications*. Kluwer, Boston, MA.

Rochat, Y., E. Taillard. 1995. Probabilistic diversification and intensification in local search for vehicle routing. *J. Heuristics* **1** 147–167.

Taillard, E. 1993. Parallel iterative search methods for vehicle routing problem. *Networks* **23** 661–673.

Toth, P., D. Vigo. 1997. A granular tabu search algorithm for constrained arborescence problems. *Proc. 16th Internat. Sympos. Math. Programming*, Lausanne, Switzerland, 277–278.

Toth, P., D. Vigo. 1998. Exact solution of the vehicle routing problem. T. G. Crainic, G. Laporte, eds. *Fleet Management and Logistics*. Kluwer, Boston, MA, 1–31.

Toth, P., D. Vigo, eds. 2002. *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications.* SIAM, Philadelphia, PA.

Vigo, D. 1996. A heuristic algorithm for the asymmetric capacitated vehicle routing problem. *Eur. J. Oper. Res.* **89** 108–126.

Xu, J. 1998. Personal communication.

Xu, J., J. Kelly. 1996. A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Sci.* **30** 379–393.

Accepted by Michel Gendreau; received December 1998; revised October 2001; accepted April 2002.