1 METAHEURISTICS AND APPLICATIONS TO OPTIMIZATION PROBLEMS IN TELECOMMUNICATIONS

Simone L. Martins[1] and Celso C. Ribeiro[1]

[1]Department of Computer Science
Universidade Federal Fluminense
Niterói, Rio de Janeiro 22410-240, Brazil
simone@ic.uff.br
celso@ic.uff.br

Abstract: Recent years have witnessed huge advances in computer technology and communication networks, entailing hard optimization problems in areas such as network design and routing. Metaheuristics are general high-level procedures that coordinate simple heuristics and rules to find good approximate solutions to computationally difficult combinatorial optimization problems. Among them, we find simulated annealing, tabu search, GRASP, VNS, genetic algorithms, and others. They are some of the most effective solution strategies for solving optimization problems in practice and have been applied to a very large variety of problems in telecommunications. In this chapter, we review the main components that are common to different metaheuristics. We also describe the main principles associated with several metaheuristics and we give templates for basic implementations of them. Finally, we present an account of some successful applications of metaheuristics to optimization problems in telecommunications.

Keywords: Metaheuristics, telecommunications, networks, network design, network routing.

1.1 INTRODUCTION

Recent years have witnessed huge advances in computer technology and communication networks, entailing hard optimization problems in areas such as network design and routing (Martins et al., 2004). They often concern the minimization of the costs involved in the design of the networks or the optimization of their performance.

Combinatorial optimization problems in telecommunications and other areas involve finding optimal solutions from a discrete set of feasible solutions. However, even with the advent of new computer technologies and parallel processing, many of these problems cannot be solved to optimality in reasonable computation times, due to their inner nature or to their size. Moreover, reaching optimal solutions is meaningless in many practical situations, since we are often dealing with rough simplifications of reality and the available data is not precise. The goal of approximate algorithms (or heuristics) is to quickly produce good approximate solutions, without necessarily providing any guarantee of solution quality.

Metaheuristics are general high-level procedures that coordinate simple heuristics and rules to find good (often optimal) approximate solutions to computationally difficult combinatorial optimization problems. Among them, we find simulated annealing, tabu search, GRASP, genetic algorithms, scatter search, VNS, ant colonies, and others. They are based on distinct paradigms and offer different mechanisms to escape from locally optimal solutions, contrarily to greedy algorithms or local search methods. Metaheuristics are among the most effective solution strategies for solving combinatorial optimization problems in practice and they have been applied to a very large variety of areas and situations. The customization (or instantiation) of some metaheuristic to a given problem yields a heuristic to the latter.

In this chapter, we consider the combinatorial optimization problem of

$$\text{minimizing } f(S) \text{ subject to } S \in X,$$

defined by a ground set $E = \{e_1, \ldots, e_n\}$, a set of feasible solutions $X \subseteq 2^E$, and an objective function $f : 2^E \to \mathbb{R}$. We seek an optimal solution $S^* \in X$ such that $f(S^*) \leq f(S)$, $\forall S \in X$. The ground set E, the objective function f, and the feasible set X are specific to each problem.

Principles and building blocks that are common to different metaheuristics are reviewed in the next section. The main metaheuristics and templates of their basic implementations are described in Section 1.3. We also comment on hybridizations combining components from different metaheuristics, that are currently among the most effective algorithms for solving real-life problems. Applications of metaheuristics to several problems in telecommunications and network design and routing are reviewed in Section 1.4.

1.2 PRINCIPLES AND BUILDING BLOCKS

Several components are common to different metaheuristics. They are often blended using different strategies and additional features that distinguish one metaheuristic from the other.

1.2.1 Greedy algorithms

In a greedy algorithm, solutions are progressively built from scratch. At each iteration, a new element from the ground set E is incorporated into the partial solution under construction, until a complete feasible solution is obtained. The selection of the next element to be incorporated is determined by the evaluation of all candidate elements according to a greedy evaluation function. This greedy function usually represents the incremental increase in the cost function due to the incorporation of this element into the partial solution under construction. The greediness criterion establishes that the element with the smallest incremental increase is selected, with ties being arbitrarily broken. Figure 1.1 provides a template for a greedy algorithm for a minimization problem.

```
procedure GreedyAlgorithm()
1.  $S \leftarrow \emptyset$;
2.  Evaluate the incremental cost of each element $e \in E$;
3.  while $S$ is not a complete feasible solution do
4.      Select the element $s \in E$ with the smallest incremental cost;
5.      $S \leftarrow S \cup \{s\}$;
6.      Update the incremental cost of each element $e \in E \setminus S$;
7.  end_while;
8.  return $S$;
end.
```

Figure 1.1 Greedy algorithm for minimization.

The solutions obtained by greedy algorithms are not necessarily optimal. Greedy algorithms are often used to build initial solutions to be explored by local search or metaheuristics.

1.2.2 Randomization and greedy randomized algorithms

Randomization plays a very important role in algorithm design. Metaheuristics such as simulated annealing, GRASP, and genetic algorithms rely on randomization to sample the search space. Randomization can also be used to break ties, so as that different trajectories can be followed from the same initial solution in multistart methods or to sample fractions of large neighborhoods. One particularly important use of randomization appears in the context of greedy algorithms.

```
procedure GreedyRandomizedAlgorithm(Seed)
1.  $S \leftarrow \emptyset$;
2.  Evaluate the incremental costs of each element $e \in E$;
3.  while $S$ is not a complete solution do
4.      Build the restricted candidate list (RCL);
5.      Select an element $s$ from the RCL at random;
6.      $S \leftarrow S \cup \{s\}$;
7.      Update the incremental cost of each element $e \in E \setminus S$;
8.  end;
9.  return $S$;
end.
```

Figure 1.2 Greedy randomized algorithm for minimization.

Greedy randomized algorithms are based on the same principle of pure greedy algorithms, but make use of randomization to build different solutions at different runs. Figure 1.2 illustrates the pseudo-code of a greedy randomized algorithm for minimization. At each iteration, the set of candidate elements is formed by all elements that can be incorporated into the partial solution under construction without destroying feasibility. As before, the selection of the next element is determined by the evaluation of all candidate elements according to a greedy evaluation function. The evaluation of the elements by this function leads to the creation of a restricted candidate list (RCL) formed by the best elements, i.e. those whose incorporation into the current partial solution results in the smallest incremental costs. The element to be incorporated into the partial solution is randomly selected from those in the RCL. Once the selected element has been incorporated into the partial solution, the incremental costs are reevaluated.

Greedy randomized algorithms are used e.g. in the construction phase of GRASP heuristics or to create initial solutions to population metaheuristics such as genetic algorithms or scatter search; see Section 1.3. Randomization is also a major component of metaheuristics such as simulated annealing and VNS, in which a solution in the neighborhood of the current one is randomly generated at each iteration; see also Section 1.2.

1.2.3 Neighborhoods

A neighborhood of a solution S is a set $N(S) \subseteq X$. Each solution $S' \in N(S)$ is reached from S by an operation called a *move*. Normally, two neighbor solutions S and $S' \in N(S)$ differ by only a few elements. Neighborhoods may also eventually contain infeasible solutions not in X.

A solution S^* is a local optimum with respect to a given neighborhood $N()$ if $f(S^*) \leq f(S), \forall S \in N(S^*)$. Local search methods are based on the exploration of solution neighborhoods, searching for improving solutions until a local optimum is found.

The definition of a neighborhood is not unique. Metaheuristics such as VNS make use of multiple neighborhood structures. A metaheuristic may also modify the neighborhood, by excluding some of the possible moves and introducing others. Such modifications might also lead to the need of changes in the nature of solution evaluation. The strategic oscillation approach (Glover, 1996; 2000; Glover and Laguna, 1997) illustrates this intimate relationship between changes in neighborhood and changes in evaluation.

1.2.4 Local search

Solutions generated by greedy algorithms are not necessarily optimal, even with respect to simple neighborhoods. A local search technique attempts to improve solutions in an iterative fashion, by successively replacing the current solution by a better solution in a neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The pseudo-code of a basic local search algorithm for a minimization problem is given in Figure 1.3. It starts from a solution S and makes use of a neighborhood structure $N()$.

```
procedure LocalSearch($S$)
1.  while $S$ is not locally optimal do
2.      Find $S' \in N(S)$ with $f(S') < f(S)$;
3.      $S \leftarrow S'$;
4.  end;
5.  return $S$;
end.
```

Figure 1.3 Local search algorithm for minimization.

The effectiveness of a local search procedure depends on several aspects, such as the neighborhood structure, the neighborhood search technique, the speed of evaluation of the cost function, and the starting solution. The neighborhood search may be implemented using either a *best-improving* or a *first-improving* strategy. In the case of a best-improving strategy, all neighbors are investigated and the current solution is replaced by the best neighbor. In the case of a first-improving strategy, the current solution moves to the first neighbor whose cost function value is smaller than that of the current solution. In practice, we observe that quite often both strategies lead to the same final solution, but in smaller computation times when the first-improving strategy is used. We also observe that premature convergence to a non-global local minimum is more likely to occur with a best-improving strategy.

1.2.5 Restricted neighborhoods and candidate lists

Glover and Laguna (1997) point out that the use of strategies to restrict neighborhoods and to create candidate lists is essential to restrict the number of solutions examined in a given iteration, in situations where the neighborhoods are very large or their elements are expensive to evaluate.

Their goal consists in attempting to isolate regions of the neighborhood containing desirable features and inserting them into a list of candidates for close examination. The efficiency of candidate list strategies can be enhanced by the use of memory structures for efficient updates of move evaluations from one iteration to another. The effectiveness of a candidate list strategy should be evaluated in terms of the quality of the best solution found in some specified amount of computation time. Strategies such as aspiration plus, elite candidate list, successive filter strategy, sequential fan candidate list, and bounded change candidate list are reviewed in Glover and Laguna (1997).

Ribeiro and Souza (2000) used a candidate list strategy to significantly speedup the search for the best neighbor in their tabu search heuristic for the Steiner problem in graphs, based on quickly computed estimates of move values. Moves with bad estimates were discarded. Restricted neighborhoods based on filtering unpromising solutions with high evaluations are discussed, for example, in Martins et al. (1999) and Resende and Ribeiro (2003c).

1.2.6 Adaptive memory

The core idea of metaheuristics such as tabu search is the use of an *adaptive memory*, contrarily to memoryless approaches such as simulated annealing and rigid memory designs typical of branch-and-bound strategies. Such memory structures operate by reference to four principal dimensions: recency, frequency, quality, and influence. Memory can be explicit (full solutions are recorded, typically consisting of elite solutions visited during the search) or attributive (attributes that change in moving from one solution to another are recorded).

Fleurent and Glover (1999) and Fernandes and Ribeiro (2005) have successfully used adaptive memory strategies to improve the solutions constructed by multistart procedures.

1.2.7 Intensification and diversification

Two important components of metaheuristics are *intensification* and *diversification*. Intensification strategies encourage move combinations and solution features historically found good or return to explore attractive regions of the solution space to visit them more thoroughly. The implementation of intensification strategies enforces the investigation of neighborhoods of elite solutions and makes use of explicit memory to do so.

Diversification strategies encourage the search to examine unvisited solutions of the solution space or to generate solutions that significantly differ from those previously visited. Penalty and incentive functions are often used in this context.

1.2.8 Path-relinking

Path-relinking was originally proposed by Glover (1996) as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search. Starting from one or more elite solutions, paths in the solution space leading toward other elite solutions are generated and explored in the search for better solutions. To generate paths, moves are selected to introduce attributes in the current solution that are present in the elite guiding solution. Path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions, by favoring these attributes in the selected moves.

The algorithm in Figure 1.4 illustrates the pseudo-code of the path-relinking procedure applied to a pair of solutions S_s (starting solution) and S_t (target solution). The procedure starts by computing the symmetric difference $\Delta(S_s, S_t)$ between the two solutions, i.e. the set of moves needed to reach S_t from S_s. A path of solutions is generated linking S_s to S_t. The best solution $\bar{S}$ in this path is returned by the algorithm. At each step, the procedure examines all moves $m \in \Delta(S, S_t)$ from the current solution S and selects the one which results in the least cost solution, i.e. the one which minimizes $f(S \oplus m)$, where $S \oplus m$ is the solution resulting from applying move m to solution S. The best move m^* is made, producing solution $S \oplus m^*$. The set of available moves is updated. If necessary, the best solution $\bar{S}$ is updated. The procedure terminates when S_t is reached, i.e. when $\Delta(S, S_t) = \emptyset$.

```
procedure PathRelinking($S_s, S_t$)
1.  Compute the symmetric difference $\Delta(S_s, S_t)$;
2.  $\underline{f} \leftarrow \min\{f(S_s), f(S_t)\}$;
3.  $\underline{S} \leftarrow \text{argmin}\{f(S_s), f(S_t)\}$;
4.  $S \leftarrow S_s$;
5.  while $\Delta(S, S_t) \neq \emptyset$ do
6.      $m^* \leftarrow \text{argmin}\{f(S \oplus m) : m \in \Delta(S, S_t)\}$;
7.      $\Delta(S \oplus m^*, S_t) \leftarrow \Delta(S, S_t) \setminus \{m^*\}$;
8.      $S \leftarrow S \oplus m^*$;
9.      if $f(S) < \underline{f}$ then
10.         $\underline{f} \leftarrow f(S)$;
11.         $\underline{S} \leftarrow S$;
12.     end_if;
13. end_while;
14. return $\underline{S}$;
end.
```

Figure 1.4 Path-relinking procedure for minimization.

Path-relinking may also be viewed as a constrained local search strategy applied to the initial solution S_s, in which only a limited set of moves can be performed and uphill moves are allowed. Several alternatives have been considered and combined in recent successful implementations of path-relinking in conjunction with tabu search, GRASP, and genetic algorithms (Aiex et al., 2003; 2005; Binato et al., 2001; Canuto et al., 2001; Festa et al., 2002; Martins et al., 2004; Reeves and Yamada, 1998; Resende and Ribeiro, 2003a; Ribeiro and Rosseti, 2002; Ribeiro et al., 2002; Ribeiro and Vianna, 2003; Souza et al., 2003): periodical, forward, backward, back and forward, mixed, randomized, truncated, and post-optimization relinking. Resende and Ribeiro (2003b) reviewed these alternatives, showing that they involve trade-offs between computation time and solution quality. In particular, Ribeiro et al. (2002) observed that exploring two different trajectories for each pair (S_s, S_t) takes approximately twice the time needed to explore only one of them, with very marginal improvements in solution quality. They have also observed that if only one trajectory is to be investigated, better solutions are found when the relinking procedure starts from the best among S_s and S_t. Since the neighborhood of the initial solution is much more carefully explored than that of the guiding one, starting from the best of them gives the algorithm a better chance to investigate in more detail the neighborhood of the most promising solution. For the same reason, the best solutions are usually found closer to the initial solution than to the guiding solution, allowing pruning the relinking trajectory before the latter is reached.

1.3 METAHEURISTICS AND TEMPLATES

In this section we review the main principles and ideas involved with five widely and successfully used metaheuristics and we give templates for their implementation: simulated annealing, tabu search, GRASP, VNS, and genetic algorithms.

1.3.1 Simulated annealing

Simulated annealing was introduced by Kirkpatrick et al. (1983) and its basic ideas come from an analogy with the physical annealing process. In physics, annealing is a thermal process for obtaining low energy states of a solid in a heat bath. Computer simulation methods based on Monte Carlo techniques are used to model this process. A solid in state i has an energy E_i and a new state j is generated by applying a perturbation mechanism that leads to the energy level E_j. If the difference $E_j - E_i$ is less than or equal to zero, the state of the solid is changed to j. Otherwise, the state transition is performed with probability $e^{(E_i - E_j)/(K_B T)}$, where T is the temperature of the heat bath and K_B is the Boltzmann constant.

The principle of the simulated annealing technique lies in the following analogy between the physical process and a combinatorial optimization problem: solutions of the combinatorial problem are equivalent to states of the physical problem and their cost to the energy of a state. Neighboring solutions correspond to states generated by the perturbation method.

```
procedure SimulatedAnnealing()
1.  Generate an initial solution $S_0$ and set $S \leftarrow S_0$;
2.  Compute the initial temperature $T_0$ and set $T \leftarrow T_0$;
3.  while stopping criterion is not reached do
4.      while thermal equilibrium is not reached do
5.          Obtain a neighbor solution $S' \in N(S)$ at random;
6.          Compute $\Delta E = f(S') - f(S)$;
7.          if $\Delta E < 0$ then $S \leftarrow S'$;
8.          else if $e^{-\Delta E/(K_B T)} > random[0,1)$ then $S \leftarrow S'$;
9.          end_if
10.     end_while
11.     Decrease $T$ according with the annealing schedule;
12. end_while;
13. return $S$;
end.
```

Figure 1.5 Template of a simulated annealing heuristic for minimization.

The pseudo-code of a simulated annealing algorithm for a minimization problem is described in Figure 1.5. First, an initial solution S_0 is computed and the temperature is set at T_0. Iterations are performed until a certain *stopping criterion* is met. In most implementations, the latter corresponds to bringing the temperature to a very small value close to zero. For each temperature T, the inner loop is performed until *thermal* *equilibrium* is reached. The latter is often implemented as a fixed number of iterations which depends on the temperature. A neighbor solution S' and the variation ΔE in the objective value are computed at each iteration. The new solution S' replaces the incumbent if $\Delta E < 0$, i.e., if the new solution is better. The same happens with probability $e^{-\Delta E/(K_B T)}$ in case the new solution is worse than the current one. Once *thermal equilibrium* is reached, the temperature T is reduced according with the *annealing schedule*. In most implementations the new temperature is geometrically reduced, by the multiplication of the current temperature by a constant smaller than one.

Simulated annealing starts with large values for the temperature T, allowing bad solutions to replace the incumbent. As the algorithm executes, the temperature T is decreased and it becomes harder to accept bad solutions. No further deteriorations are accepted when T approaches to zero. Convergence relies on several implementation choices: (a) the initial value T_0 of the temperature, (b) the *annealing schedule* to reduce the temperature, (c) the *stopping criterion*, and (d) the conditions for reaching the *thermal equilibrium* at each temperature. Convergence and strategies for such implementation choice are discussed, for example, in Aarts and Korst (2002), Aarts and Korst (1989), and Henderson et al. (2003).

The main advantage of simulated annealing is its simplicity of implementation. Diversification is allowed by accepting bad solutions with decreasing probability. However, the speed of convergence and solution quality rely on several implementation choices that may not be easy to tune. Furthermore, the algorithm is memoryless and does not use cost information about complete neighborhoods.

1.3.2 Tabu search

The fundamental ideas of *tabu search* were independently proposed by Glover (1986) and Hansen (1986). The approach was later developed by Glover (1989) and Glover (1990). It may be viewed as a dynamic neighborhood method that makes use of memory to drive the search by escaping from local optima and avoiding cycling (Glover and Laguna, 1997). Contrarily to memoryless heuristics such as simulated annealing, and to methods that use rigid memory structures such as branch-and-bound, tabu search makes use of flexible and adaptive memory designs.

For any incumbent solution S, in the case of tabu search the neighborhood $N(S)$ over which local search is applied is not a static set. Instead, it can change according to the history of the search. At each local search iteration, tabu search looks for the neighbor solution that most improves the objective function. However, some neighbors in $N(S)$ are forbidden and discarded. The set of forbidden (tabu) neighbors is stored in a short-term memory formed by the `TabuTenure` last visited solutions, so as to preclude the search from returning to previously visited solutions. The list of forbidden solutions is usually implemented as a list of forbidden moves, that discard a broader subset of solutions. Contrarily to plain local search procedures, tabu search chooses the move which least deteriorates the objective function value in case there are no improving moves.

The basic steps of a simple tabu search heuristic for minimization are described in the pseudo-code of Figure 1.6. First, an initial solution S_0 is computed and the short-term memory is initialized. Iterations are performed until a certain *stopping* *criterion* is met. In most implementations, the latter corresponds to an upper limit on the number of consecutive moves without improvement in the best solution value. Procedure `SelectBestNeighbor` returns the best non-forbidden neighbor solution S'. The best known solution is eventually updated. If the tabu list is full, then the oldest forbidden solution is removed from T. The incumbent solution solution is inserted into the tabu list and is replaced by the best neighbor S'.

```
procedure TabuSearch();
1.  Generate an initial solution $S_0$ and set $S \leftarrow S_0$;
2.  $S^* \leftarrow S$;
3.  $T \leftarrow \emptyset$;
4.  while stopping criterion is not reached do
5.      $S' \leftarrow$ SelectBestNeighbor($N(S) \setminus T$);
6.      if $f(S') < f(S^*)$ then
7.          $S^* \leftarrow S'$;
8.      end_if;
9.      if $|T| =$ TabuTenure then
10.         Remove from $T$ the oldest solution;
11.     end_if;
12.     $T \leftarrow T \cup S$;
13.     $S \leftarrow S'$;
14. end_while;
15. return $S^*$;
end.
```

Figure 1.6 Template of a short-term memory tabu search heuristic for minimization.

The tabu search scheme above described makes use of very simple ideas and can be further extended by the incorporation of more sophisticated features. Among them, we may cite the use of aspiration criteria to override the tabu status of unvisited solutions, the use of medium-term and long-term memories to implement intensification and diversification procedures, the use of candidate list strategies to speedup the search, and the use of hashing tables to speedup and filter the search.

Tabu search is certainly among the most effective approaches for solving hard combinatorial optimization problems. It has been successfully applied to a wide range of problems in many domains. However, implementations of tabu search often involve setting many parameters, that have to be appropriately tuned for achieving good performance in practice. The interested reader is referred to Glover and Laguna (1997) for a thorough study of tabu search and related ideas.

1.3.3 GRASP

GRASP (Greedy Randomized Adaptive Search Procedure) (Feo and Resende, 1989; 1995; Resende and Ribeiro, 2003b;c; Ribeiro, 2002) is a multistart metaheuristic, in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result. The pseudo-code in Figure 1.7 illustrates the main blocks of a GRASP procedure for minimization, in which `MaxIterations` iterations are performed and `Seed` is used as the initial seed for the pseudo-random number generator.

```
procedure GRASP(MaxIterations, Seed)
1.  Set $f^* \leftarrow \infty$;
2.  for $k = 1, \ldots,$ MaxIterations do
3.      $S \leftarrow$ GreedyRandomizedAlgorithm(Seed);
4.      $S \leftarrow$ LocalSearch($S$);
5.      if $f(S) < f^*$ then
6.          $S^* \leftarrow S$;
7.          $f^* \leftarrow f(S)$;
8.      end_if;
9.  end_for;
10. return $S^*$;
end.
```

Figure 1.7 Template of a GRASP heuristic for minimization.

An especially appealing characteristic of GRASP is the ease with which it can be implemented. Few parameters need to be set and tuned, and therefore development can focus on implementing efficient data structures to assure quick iterations. Basic implementations of GRASP rely exclusively on two parameters: the number `MaxIterations` of iterations and the parameter used to limit the size of the restricted candidate list within the greedy randomized algorithm used by the construction phase. In spite of its simplicity and ease of implementation, GRASP is a very effective metaheuristic and produces the best known solutions for many problems (see also Festa and Resende (2002) for an extensive survey of applications).

GRASP as originally proposed is a memoryless procedure, in which each iteration does not make use of information gathered in previous iterations. Path-relinking is a major enhancement that is able to add memory to the basic GRASP procedure, leading to significant improvements in solution time and quality. The use of path-relinking within a GRASP procedure, as an intensification strategy applied to each locally optimal solution, was first proposed by Laguna and Martí (1999). It was followed by several extensions, improvements, and successful applications (Aiex et al., 2003; 2005; Canuto et al., 2001; Resende and Ribeiro, 2003a;b;c; Resende and Werneck, 2004; Ribeiro and Rosseti, 2002; Ribeiro et al., 2002; Souza et al., 2003).

In this context, path-relinking is applied to pairs of solutions, one of them being a locally optimal solution and the other randomly chosen from a pool with a limited number `MaxElite` of elite solutions found along the search. The pool of elite solutions is originally empty. Since we wish to maintain a pool of good but diverse solutions, each locally optimal solution obtained by local search is considered as a candidate to be inserted into the pool if it is sufficiently different from every other solution currently in the pool. If the pool already has `MaxElite` solutions and the candidate is better than the worst of them, then a simple strategy is to have the former replacing the latter.

The pseudo-code in Figure 1.8 illustrates the main steps of a GRASP procedure using path-relinking to implement a memory-based intensification strategy.

```
procedure GRASPwithPathRelinking(MaxIterations,Seed)
1.  Set $f^* \leftarrow \infty$;
2.  Set $Pool \leftarrow \emptyset$;
3.  for $k = 1, \dots,$ MaxIterations do
4.      $S \leftarrow$ GreedyRandomizedAlgorithm(Seed);
5.      $S \leftarrow$ LocalSearch($S$);
6.      if $k > 1$ then
7.          Randomly select a solution $S' \in Pool$;
8.          $S \leftarrow$ PathRelinking(S',S);
9.      endif;
10.     if $f(S) < f^*$ then
11.         $S^* \leftarrow S$;
12.         $f^* \leftarrow f(S)$;
13.     end_if;
14.     Add $S$ to $Pool$ if it satisfies the membership conditions;
15. end_for;
16. return $S^*$;
end.
```

Figure 1.8 Template of a GRASP with path-relinking heuristic for minimization.

1.3.4 Variable neighborhood search

VNS (Variable Neighborhood Search) is a metaheuristic proposed by Hansen and Mladenović (1999; 2002; 2003); Mladenović and Hansen (1997) based on the systematic change of neighborhood. It makes use of a finite set of k_{max} pre-selected neighborhood structures identified as $N_1, N_2, \dots, N_{k_{max}}$ that may be induced from one or more metric functions introduced in the solution space X. We denote by $N_k(S)$ the set of solutions in the kth neighborhood of S.

Figure 1.9 summarizes the main steps of the pseudo-code of a VNS heuristic. First, an initial solution S_0 is computed. Iterations of the outer loop are performed until a certain *stopping criterion* is met. The current neighborhood is set to $k = 1$ and neighborhoods N_1 to $N_{k_{max}}$ of the current solution are progressively scanned within the inner loop. The search starts from the lowest order neighborhood $N_1(S)$ of the current solution. The initial step of each iteration is a random perturbation move applied to generate a solution S' in the current neighborhood $N_k(S)$. Next, a locally optimal solution S'' is obtained by the application of local search to S'. If solution S'' is better than the incumbent S, then the former replaces the latter and the search resumes from neighborhood $N_1(S)$. Otherwise, the current solution S is not modified,

k is incremented by one, and the search resumes from a higher order neighborhood of S.

```
procedure VNS($k_{max}$, Seed)
1.   Generate an initial solution $S_0$ and set $S \leftarrow S_0$;
2.   while stopping criterion is not reached do
3.       $k \leftarrow 1$;
4.       while $k \leq k_{max}$ do
5.           Obtain a neighbor solution $S' \in N_k(S)$ at random;
6.           $S'' \leftarrow$ LocalSearch($S'$);
7.           if $f(S'') < f(S)$ then
8.               $S \leftarrow S''$;
9.               $k \leftarrow 1$;
10.          else
11.              $k \leftarrow k + 1$;
12.          endif;
13.      end_while;
14.  end_while;
15.  return $S$;
end.
```

Figure 1.9 Template of a VNS heuristic for minimization.

VNS is also a memoryless heuristic, being very easy to implement and relying on very few parameters: the *stopping criterion* and the number k_{max} of neighborhoods. The former may be the maximum number of iterations, the maximum number of iterations between two improvements, or the maximum number of times the highest order neighborhood $N_{k_{max}}$ is attained. Although this is not necessary, successive neighborhoods N_k are nested in most implementations of VNS, i.e., $N_1(S) \subset N_2(S) \subset \ldots \subset N_{k_{max}}(S)$. In this case, solutions in neighborhood $N_k(S)$ are progressively more distant from S as k increases. We refer to Aloise et al. (2005) for a successful application of VNS in which the neighborhoods are not nested.

1.3.5 Genetic algorithms

Genetic algorithms were first introduced and investigated by Holland (1975; 1992). They are population-based methods that use selection and recombination operators to generate new solutions in the search space, imitating the process of natural selection and evolution of species. Solutions are evaluated in terms of their fitness, which most often corresponds to the value of the objective function itself. In some implementations, the fitness of a solution may also account for penalties due to infeasibilities.

Contrarily to other metaheuristics reviewed in this chapter, genetic algorithms explore a population of solutions and not a single trajectory evolving from a unique initial solution. Given a population of solutions, the main operators involved in standard implementations of genetic algorithms are:

- `SelectMates`: a subset of solutions is selected, either randomly or using fitness information to privilege promising solutions.
- `RecombineParents`: pairs of solutions are combined and new solutions are generated (crossover operation).
- `ApplyMutation`: a few solutions are randomly selected and go through small modifications in their structure (mutation operation).
- `SelectBest`: the best solutions are selected, while the others are eliminated.

```
procedure GeneticAlgorithm()
1.  Generate the initial population $P$ of solutions;
2.  $k \leftarrow 0$;
3.  EvaluateFitness($P$);
4.  while stopping criterion is not reached do
5.      $Parents \leftarrow$ SelectMates($P$);
6.      $Children \leftarrow$ RecombineParents($Parents$);
7.      $Children \leftarrow$ ApplyMutation($Children$);
8.      EvaluateFitness($Children$);
9.      $P \leftarrow$ SelectBest($Children \cup P$);
10.     $k \leftarrow k+1$;
11. end_while
12. return the best solution in $P$;
end.
```

Figure 1.10 Template of a genetic algorithm.

The pseudo-code of the basic steps of a genetic algorithm are described in Figure 1.10. The procedure starts by generating an initial population P of solutions, either entirely randomly or using a greedy randomized algorithm. The fitness of each solution S in the population is evaluated. The loop is performed until some *stopping criterion* is reached. Most applications use the number of iterations, the stabilization of the population, or the stabilization of the best solution in the population. Each iteration of this loop handles a generation of the population. The process of obtaining a new generation starts by the selection of a subset *Parents* of solutions from the current population. Solutions are pairwise combined by crossover operations and a set *Children* of off-springs results. A small subset of the latter go through mutations and the fitness of every newly generated solution is evaluated. Finally, the best among the old and the new solutions are selected to form a new generation of solutions and a new iteration starts.

Classical genetic algorithms do not incorporate any problem-specific knowledge and rely almost entirely on randomization for mate selection, crossover, mutation, and selection. Hybrid genetic algorithms, also called memetic algorithms, make use of specific knowledge available to solve the problem. One of such strategies consists in applying local search to some elements in the generation.

Genetic algorithms are very appealing and easy to implement. However, many implementation choices for each of the above operators are possible. A good implementation requires an appropriate solution encoding as finite length data strings and well tuned strategies for the generation of the initial population, mate selection, crossover, mutation, and selection; see e.g. Reeves (1993) and Reeves (2003).

Scatter search (Glover, 1996; 2000; Glover et al., 2003) and ant colony optimization (Dorigo and Stützle, 2003) are other population-based metaheuristics.

1.3.6 Hybridizations

Hybrid heuristics combining features and strategies borrowed from different metaheuristics are among the most efficient algorithms for finding good approximate solutions for combinatorial problems.

We illustrate this issue with some examples involving the hybridization of GRASP with other metaheuristics. Since GRASP and VNS are complementary in the sense that they make use of randomization at different stages, Festa et al. (2002) studied different variants and combinations of GRASP and VNS for the max-cut problem, showing that hybrid implementations outperform other algorithms in the literature by finding and improving the best known solutions for several benchmark instances.

GRASP has also been used in conjunction with genetic algorithms. The construction phase of a GRASP may be applied to generate the initial population for a genetic algorithm. The genetic algorithm of Ahuja et al. (2000) for the quadratic assignment problem makes use of the GRASP proposed by Li et al. (1994) to create the initial population. A similar approach was used in Armony et al. (2000).

The hybridization of GRASP with tabu search was first studied by Laguna and González-Velarde (1991). Delmaire et al. (1999) considered two approaches. In the first, GRASP is applied as a powerful diversification strategy in the context of a tabu search procedure. The second approach is an implementation of the Reactive GRASP algorithm (Prais and Ribeiro, 2000), in which the local search phase is strengthened by tabu search. Results reported for the capacitated location problem show that the hybrid approaches perform better than the isolated methods previously used.

1.4 APPLICATIONS IN TELECOMMUNICATIONS

The outbreak of new technologies in telecommunications networks, together with the demand for more computer intensive applications, leads to huge developments and needs in network design and routing. As most of these problems are NP-hard and exact methods can only solve small problems, approximate algorithms based on metaheuristics play a very important role in their solution.

Communication networks consist of nodes that can be computers, database repositories, instruments like tomography equipments or radio transmitters, connected by data transmission links such as copper cables, optical fibers, satellite and radio links. Their design involves making decisions on many issues like the number of nodes and their locations, routing paths, capacity installation, wavelength allocation, and frequency assignment. The main objective is often to obtain a least cost network configuration subject to constraints involving issues such as delay, throughput, reliability, link capacity, and crosstalk level.

Applications of metaheuristics to problems in telecommunications are plentiful in the literature. We give below a partial account of some of these applications, organized according to the main metaheuristic used to solve each problem.

Kim et al. (2000) worked out a simulated annealing algorithm to allocate nominal channels to the cells of a mobile radio system in such a way that the average blocking in the entire system is minimized. Experimental results show that their proposal gives the lowest overall blocking probability and the largest number of simultaneously usable channels, when compared with other three heuristics found in the literature.

Randall et al. (2002) showed results obtained by a simulated annealing algorithm developed to find paths in a network which minimize the cost of transporting required origin-destination flows subject to specified link, node capacity, node degree, and chain hop-limit constraints. They showed that the heuristics found good results for very large network design and in smaller times than another genetic algorithm also developed for this problem.

Amaldi et al. (2003) proposed two randomized greedy procedures and a tabu search algorithm for finding the location of universal mobile telecommunication system base stations to maximize the traffic covered and to minimize installation costs. The greedy function takes into account the fraction of traffic covered and the installation costs. The randomized greedy procedures provided good approximate solutions from medium- to large-size realistic instances, and the tabu search algorithm was able to further improve the solutions obtained by the greedy procedures.

Cox and Sanchez (2000) presented a new heuristic algorithm for designing least-cost telecommunications networks to carry traffic from cell sites to wireless switches while meeting survivability, capacity, and technical compatibility constraints. A short term tabu search metaheuristic was introduced (that do not use longer-term features, such as search intensification and diversification strategies), with embedded knapsack and network flow sub-problems. It has proved highly effective in designing such back-haul networks for carrying personal communications services traffic. It solved challenging problems for conventional branch-and-bound solvers in minutes instead of hours and found lower-cost solutions. Applied to real-world network design problems, the heuristic has successfully identified designs that save over 20% compared to the best previously known designs.

Fink et al. (1999) present a more general problem formulation for ring network design problems and associated methods that apply to a broad range of problems. They implemented a simulated annealing and a tabu search algorithm using a metaheuristics framework, so that no calibrations or specializations with respect to a specific problem type were performed. Reactive tabu search presented better results than simulated annealing.

Gendron et al. (2000) proposed a heuristic approach for a network loading problem that alternates construction and local search phases. Initially, a construction method provides a feasible solution, while at subsequent construction steps, a diversification approach is adopted for exploiting information gathered along previous iterations.

Two local search procedures are used: a pure descent search and a tabu search heuristic. An implementation of GRASP is also proposed.

Girard et al. (2001) examined the use of tabu search for the solution of an access network design problem that uses ADM equipments, which are both concentrators and multiplexers, so that demand can be sent on different SONET channels. They compared a simple tabu algorithm and a more elaborated one, using simulated and real instances. They showed that the latter found improved solutions in much less computation times. They also reported that the differences between the results obtained by tabu search and the exact values were quite small for some instances

Hao et al. (1998) presented a tabu search algorithm for assigning frequencies in mobile radio networks, while minimizing electromagnetic interference. Experimental results show that the tabu search algorithm largely outperforms other algorithms based on simulated annealing, constraint programming, graph coloring, and steepest descent, also developed for this problem. Castelino and Stephens (1999) developed a surrogate constraint tabu thresholding algorithm for the same problem. The aim of this work was to determine whether using surrogate constraints can improve the solutions found. The use of surrogate constraints consists in combining problem constraints to provide information for guiding the search, instead of using them in isolation. They compared their algorithm to a thresholding tabu search and showed that the surrogate thresholding tabu found better results within the same computation time.

Klincewicz (1992) proposed two heuristics based on tabu search and GRASP for the *p*-hub location problem. The objective is to overcome the difficulty that local search algorithms encounter. The local search procedure of the GRASP algorithm is based on a 2-exchange neighborhood. The same author (Klincewicz, 2002) also proposed heuristics based on GRASP and tabu search for locating hubs in a communication or transportation network. The greedy function of the GRASP heuristic takes into account the amount of originating and terminating traffic of each possible location, while the local search procedure uses a swap neighborhood.

Noronha and Ribeiro (2004) developed a tabu search approach to routing and wavelength assignment in all-optical networks. Their algorithm combines the computation of alternative routes for the light-paths with the solution of a partition coloring problem. The computational experiments showed that their approach outperforms the previously best known heuristic for the problem (Manohar et al., 2002).

Pamuk and Sepil (2001) proposed a tabu search algorithm to perform optimal location and allocation of hubs in switched networks. They developed three strategies to generate initial solutions and another three to search for an allocation scheme. Nine versions of tabu search heuristics were developed by combining these strategies. Experimental results showed that the three initial solution strategies perform in a similar way and the performance of allocation schemes depends on the type of instance.

Xu et al. (1997) employed tabu search for designing a least-cost telecommunications network where the alternate routing paths can be changed dynamically from hour to hour. They conducted computational experiments on data drawn from real and simulated problems and showed that the tabu search algorithm found better results than two other less computer intensive strategies. They concluded that the the most effective tabu search heuristic was the one which integrates tabu search memories, probabilistic rules, and a periodic solution recovery strategy.

Abello et al. (1999; 2002) developed GRASP heuristics to approximately solve the maximum clique and maximum quasi-clique in very large graphs. Their approach is used in data mining to extract communities of interest from telephone call detail graphs.

Canuto et al. (2001) developed a GRASP heuristic for the prize-collecting Steiner tree problem, with applications in the design of telecommunications access networks. They proposed a multi-start local search algorithm and path-relinking is used to improve the solutions found by local search. VNS is used as a post-optimization procedure. Their results show that the local search with perturbations approach found optimal solutions on nearly all of the instances tested, in much smaller computation than an exact branch-and-cut algorithm.

Gabrel et al. (2003) presented and compared approximate algorithms for discrete cost multicommodity network optimization problems. Firstly, extensions of classical greedy heuristics, based on link-rerouting and flow-rerouting heuristics, are presented in details. Secondly, a new approximate solution algorithm, which basically consists of a heuristic implementation of the exact Benders-type cutting plane generation method, is proposed. All these algorithms are extensively compared on randomly generated graphs up to 50 nodes and 90 links.

Li et al. (2000) addressed the problem of server replication and placement. In a multicast network, packets are forwarded from a source server to groups of receivers along a distribution tree, where the source is the root, the receivers are the leaves, and the multicast capable routers are the internal nodes. The problem consists in placing multiple replicated servers within the multicast-capable routers. Several heuristics are proposed, including a GRASP. The greedy function is the router cost function.

Prais and Ribeiro (2000) developed a reactive GRASP heuristic for traffic assignment in TDMA communication satellites. A geostationary communication satellite has a number of spot beam antennas covering geographically distributed areas. According to the slot switching configuration and on the on-board switch, the uplink traffic received at the satellite has to be immediately sent to ground areas through a set of transponders. The slot switching configurations are determined through the solution of a time slot assignment problem, which is equivalent to the decomposition of a nonnegative traffic matrix into the sum of a family of switching mode matrices.

Resende and Ribeiro (2003a) proposed a family of heuristics for routing private virtual circuits. The GRASP with path-relinking variant was able to significantly improve upon some simple heuristics currently used in traffic engineering, at the expense of additional computation time.

Srinivasan et al. (2000) presented an approach for efficient design of a signaling network for a network of software switches supporting Internet telephony. The optimal load balancing for given demand forecast is formulated as a quadratic assignment problem, which is solved with a GRASP.

Armony et al. (2000) implemented a genetic algorithm to determine how traffic should be routed on self-healing rings, a problem known as the stacked ring design problem. The objective is to optimize the trade-off between the cost of equipments to implement the rings and the cost of exchanging traffic among rings. The initial population of the proposed GA is generated by GRASP. They showed that their GA found good quality solutions using much less time than CPLEX, and also that it always improved the initial solutions generated by GRASP.

Buriol et al. (2005) presented a hybrid genetic algorithm for solving the OSPF weight setting problem, combining a genetic algorithm with a local search procedure applied to improve the solutions obtained by crossover. Experimental results showed that the hybrid heuristic found better solutions and led to a more robust implementation than the best known heuristic in the literature.

He and Mort (2000) elaborated a hybrid genetic algorithm, where the objective was to minimize the number of point-to-point transmission (hops) for all source-destination pairs and also to minimize the number of congested nodes and links for the routing table. The initial population was created using an heuristic based on a shortest path algorithm with a minimum hop metric. They showed that the hybrid genetic algorithm found solutions with a small number of congested links and nodes than the shortest path with hops heuristic.

Poon et al. (2000) described the GenOSys tool developed to optimize the design of secondary and distribution networks used on typical copper access network cabling to connect customers. Its objective is to determine the best locations for distribution points and to identify geographically advantageous tree-structured sub-networks to aggregate cables from customers. The tool allows the user to enter data about the network and provides information which can be used for ducting and cabling using the hybrid genetic algorithm. A practical problem on a network of 240 nodes was solved in less than 30 minutes on a Pentium 200 MHz.

Watanabe et al. (2001) proposed a new type of parallel genetic algorithm model for multi-objective optimization problems. It was applied to solve an antenna arrangement problem in mobile communications. Their algorithm showed a very good performance when compared to other methods.

Ant colony optimization was applied by Varela and Sinclair (1999) to the problem of routing and wavelength-allocation in a multi-wavelength all-optical virtual-wavelength-path routed transport network. Three variants were proposed and the best one provided results that approached those of an earlier problem-specific heuristic on small- and medium-sized networks.

Wittner et al. (2003) developed a swarm algorithm to find a path of resources from a client terminal to a service provider, such that all resources in the path conform with constraints and preferences of a request profile specified by the user. Given a network composed of users, terminals, and services that have individual profiles containing quality of service parameters, the objective is to search for resource paths for each peer-to-peer communication.

Some conclusions can be drawn from the above analysis of telecommunication applications. Tabu search is often applied and usually presents good performance when compared to other heuristics. The main difficulty when using this technique is the calibration of parameters. Simulated annealing also requires a great effort to tune implementation parameters.

GRASP is a memoryless metaheuristic that is quite simple to implement, because very few parameters have to be tuned. GRASP is largely and successfully used in telecommunication applications. It is often showed that the use of techniques which add memory to GRASP, such as path-relinking and reactive GRASP, always improves the results found by simpler implementations.

Genetic algorithms also present good performance, but quite often take large computation times. Most applications use hybrid algorithms, in which a heuristic is used to generate the initial population. All results show that hybrid genetic algorithms perform much better than pure ones.

Bibliography

E. Aarts and J. Korst. Selected topics in simulated annealing. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 1–37. Kluwer, 2002.

E.H.L. Aarts and J. Korst. *Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing*. Wiley, 1989.

J. Abello, P.M. Pardalos, and M.G.C. Resende. On maximum clique problems in very large graphs. In J. Abello and J. Vitter, editors, *External memory algorithms and visualization*, volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 119–130. American Mathematical Society, 1999.

J. Abello, M.G.C. Resende, and S. Sudarsky. Massive quasi-clique detection. *Lecture Notes in Computer Science*, 2286:598–612, 2002.

R.K. Ahuja, J.B. Orlin, and A. Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Computers and Operations Research*, 27:917–934, 2000.

R.M. Aiex, S. Binato, and M.G.C. Resende. Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing*, 29:393–430, 2003.

R.M. Aiex, M.G.C. Resende, P.M. Pardalos, and G. Toraldo. GRASP with path-relinking for the three-index assignment problem. *INFORMS Journal on Computing*, 17:224–247, 2005.

D.J. Aloise, D. Aloise, C.T.M. Rocha, C.C. Ribeiro, J.C. Ribeiro Filho, and L.S.S. Moura. Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics*, 2005. To appear.

E. Amaldi, A. Capone, and F. Malucelli. Planning UMTS base station location: Optimization models with power control and algorithms. *IEEE Transactions on Wireless Communications*, 2:939–952, 2003.

M. Armony, J.C. Klincewicz, H. Luss, and M.B. Rosenwein. Design of stacked self-healing rings using a genetic algorithm. *Journal of Heuristics*, 6:85–105, 2000.

S. Binato, H. Faria Jr., and M.G.C. Resende. Greedy randomized adaptive path relinking. In J.P. Sousa, editor, *Proceedings of the IV Metaheuristics International Conference*, pages 393–397, 2001.

L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 2005. In press.

S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38:50–58, 2001.

D. Castelino and N. Stephens. A surrogate constraint tabu thresholding implementation for the frequency assignment problem. *Annals of Operations Research*, 86: 259–270, 1999.

L.A. Cox and J.R. Sanchez. Designing least-cost survivable wireless backhaul networks. *Journal of Heuristics*, 6:525–540, 2000.

H. Delmaire, J.A. Díaz, E. Fernández, and M. Ortega. Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. *INFOR*, 37:194–225, 1999.

M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 251–285. Kluwer, 2003.

T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

E.L.R. Fernandes and C.C. Ribeiro. A multistart constructive heuristic for sequencing by hybridization using adaptive memory. *Electronic Notes in Discrete Mathematics*, 2005. In press.

P. Festa, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Randomized heuristics for the max-cut problem. *Optimization Methods and Software*, 7, 2002.

P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer, 2002.

A. Fink, G. Schneidereit, and S. Voss. Solving general ring network design problems by meta-heuristics. In M. Laguna and J. L. González, editors, *Computing Tools for Modeling, Optimization and Simulation(Interfaces in Computer Science and Operations Research)*, pages 91–113. Kluwer Academic Publishers, 1999.

C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11: 198–204, 1999.

V. Gabrel, , A. Knippel, and M. Minoux. A comparison of heuristics for the discrete cost multicommodity network optimization problem. *Journal of Heuristics*, 9:429–445, 2003.

B. Gendron, J.-Y. Potvin, and P. Soriano. Diversification strategies in local search for a nonbifurcated network loading problem. *European Journal of Operational Research*, 142:231–241, 2000.

A. Girard, B. Sansó, and L. Dadjo. A tabu search algorithm for access network design. *Annals of Operations Research*, 106:229–262, 2001.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 1986.

F. Glover. Tabu search - Part I. *ORSA Journal on Computing*, 1:190–206, 1989.

F. Glover. Tabu search - Part II. *ORSA Journal on Computing*, 2:4–32, 1990.

F. Glover. Tabu search and adaptive memory programing – Advances, applications and challenges. In R.S. Barr, R.V. Helgason, and J.L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer, 1996.

F. Glover. Multi-start and strategic oscillation methods – Principles to exploit adaptive memory. In M. Laguna and J.L. Gonzáles-Velarde, editors, *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pages 1–24. Kluwer, 2000.

F. Glover and M. Laguna. *Tabu Search*. Kluwer, 1997.

F. Glover, M. Laguna, and R. Martí. Scatter search and path relinking: Advances and applications. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 1–35. Kluwer, 2003.

P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Congress on Numerical Methods in Combinatorial Optimization*, Capri, 1986.

P. Hansen and N. Mladenović. An introduction to variable neighbourhood search. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Metaheuristics: Advances and trends in local search procedures for optimization*, pages 433–458. Kluwer, 1999.

P. Hansen and N. Mladenović. Developments of variable neighborhood search. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 415–439. Kluwer, 2002.

P. Hansen and N. Mladenović. Variable neighborhood search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer, 2003.

J. Hao, R. Dorne, and P. Galinier. Tabu search for frequency assignment in mobile radio networks. *Journal of Heuristics*, 4:47–62, 1998.

L. He and N. Mort. Hybrid genetic algorithms for telecommunications network back-up routeing. *BT Technol. J.*, 18:42–50, 2000.

D. Henderson, S.H. Jacobson, and A.W. Johnson. The theory and practice of simulated annealing. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 287–319. Kluwer, 2003.

J.H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

J.H. Holland. Genetic algorithms. *Scientific American*, 267:44–50, 1992.

S-H. Kim, K-N Chang, and S. Kim. A channel allocation for cellular mobile radio systems using simulated annealing. *Telecommunication Systems*, 14:95–106, 2000.

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

J.G. Klincewicz. Avoiding local optima in the *p*-hub location problem using tabu search and grasp. *Annals of Operations Research*, 40:283–302, 1992.

J.G. Klincewicz. Enumeration and search procedures for a hub location problem with economies of scale. *Annals of Operations Research*, 110:107–122, 2002.

M. Laguna and J.L. González-Velarde. A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent Manufacturing*, 2:253–260, 1991.

M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

B. Li, F. Chen, and L. Yin. Server replication and its placement for reliable multicast. In *Proceedings of the IEEE ICCCN-00*, pages 396–401, 2000.

Y. Li, P.M. Pardalos, and M.G.C. Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems*, volume 16 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 237–261. American Mathematical Society, 1994.

P. Manohar, D. Manunath, and R.K. Shevgaonkar. Routing and wavelength assignment in optical networks from edge disjoint path algorithms. *IEEE Communication Letters*, 5:211–213, 2002.

S.L. Martins, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Greedy randomized adaptive search procedures for the Steiner problem in graphs. In P.M. Pardalos, S. Rajasejaran, and J. Rolim, editors, *Randomization Methods in Algorithmic Design*, volume 43 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 133–145. American Mathematical Society, 1999.

S.L. Martins, C.C. Ribeiro, and I. Rosseti. Applications and parallel implementations of metaheuristics in network design and routing. *Lecture Notes in Computer Science*, 3285:205–213, 2004.

N. Mladenović and P. Hansen. Variable neighbourhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

T.F. Noronha and C.C. Ribeiro. Routing and wavelength assignment by partition coloring. Technical report, Department of Computer Science, Universidade Federal Fluminense, Niterói, Rio de Janeiro 22410-240, Brazil, 2004. To appear in *European Journal of Operational Research*.

F.S. Pamuk and C. Sepil. A solution to the hub center problem via a single-relocation algorithm with tabu search. *IIE Transactions*, 33:399–411, 2001.

K.F. Poon, A. Conway, G. Wardrop, and J. Mellis. Successful application of genetic algorithms to network design and planning. *BT Technol. J.*, 18:32–41, 2000.

M. Prais and C.C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12: 164–176, 2000.

M. Randall, G. McMahon, and S. Sugden. A simulated annealing approach to communication network design. *J. of Combinatorial Optimization*, 6:55–65, 2002.

C. Reeves. Genetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 65–82. Kluwer, 2003.

C.R. Reeves. Genetic algorithms. In C.R. Reeves, editor, *Modern heuristic techniques for combinatorial problems*, pages 151–196. Wiley, 1993.

C.R. Reeves and T. Yamada. Genetic algorithms, path relinking and the flowshop sequencing problem. *Evolutionary Computation*, 6:45–60, 1998.

M.G.C. Resende and C.C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41:104–114, 2003a.

M.G.C. Resende and C.C. Ribeiro. GRASP and path-relinking: Recent advances and applications. In T. Ibaraki and Y. Yoshitomi, editors, *Proceedings of the Fifth Metaheuristics International Conference*, pages T6–1 – T6–6, 2003b.

M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer, 2003c.

M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10:59–88, 2004.

C.C. Ribeiro. GRASP: Une métaheuristique gloutone et probabiliste. In J. Teghem and M. Pirlot, editors, *Optimisation approchée en recherche opérationnelle*, pages 153–176. Hermès, 2002.

C.C. Ribeiro and I. Rosseti. A parallel GRASP heuristic for the 2-path network design problem. *Lecture Notes in Computer Science*, 2400:922–926, 2002.

C.C. Ribeiro and M.C. Souza. Tabu search for the Steiner problem in graphs. *Networks*, 36:138–146, 2000.

C.C. Ribeiro, E. Uchoa, and R.F. Werneck. A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal on Computing*, 14:228–246, 2002.

C.C. Ribeiro and D.S. Vianna. A genetic algorithm for the phylogeny problem using an optimized crossover strategy based on path-relinking. *Revista Tecnologia da Informação*, 3(2):67–70, 2003.

M.C. Souza, C. Duhamel, and C.C. Ribeiro. A GRASP with path-relinking heuristic for the capacitated minimum spanning tree problem. In M.G.C. Resende and J. Souza, editors, *Metaheuristics: Computer Decision Making*, pages 627–658. Kluwer, 2003.

A. Srinivasan, K.G. Ramakrishnan, K. Kumaram, M. Aravamudam, and S. Naqvi. Optimal design of signaling networks for Internet telephony. In *IEEE INFOCOM 2000*, March 2000.

G.N. Varela and M.C. Sinclair. Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1809–1816, Washington D.C., 1999. IEEE Press.

S. Watanabe, T. Hiroyasu, and M. Miki. Parallel evolutionary multi-criterion optimization for mobile telecommunication networks optimization. In *Proceedings of the EUROGEN 2001 Conference*, pages 167–172, Athens, 2001.

O. Wittner, P.E. Heegaard, and B. Helvik. Scalable distributed discovery of resource paths in telecommunication networks using cooperative ant-like agents. In *Proceedings of the 2003 Congress on Evolutionary Computation*, Canberra, 2003.

J. Xu, S.Y. Chiu, and F. Glover. Tabu search for dynamic routing communications network design. *Telecommunications Systems*, 8:55–77, 1997.