Chapter 5
Hybridizations of GRASP with Path-Relinking

Paola Festa and Mauricio G.C. Resende

Abstract. A greedy randomized adaptive search procedure (GRASP) is a metaheuristic for combinatorial optimization. GRASP heuristics are multistart procedures which apply local search to a set of starting solutions generated with a randomized greedy algorithm or semi-greedy method. The best local optimum found over the iterations is returned as the heuristic solution. Path-relinking is a search intensification procedure that explores paths in the neighborhood solution space connecting two good-quality solutions. A local search procedure is applied to the best solution found in the path and the local optimum found is returned as the solution of path-relinking. The hybridization of path-relinking and GRASP adds memory mechanisms to GRASP. This chapter describes basic concepts of GRASP, path-relinking, and the hybridization of GRASP with path-relinking.

5.1 Introduction

A combinatorial optimization problem can be defined by a finite ground set $E = (1,\ldots,n)$, a set of feasible solutions $F \subseteq 2^E$, and an objective function $f : 2^E \mapsto \mathbb{R}$. In this chapter, we consider optimization problems in their minimization form, where an optimal solution $S^* \in F$ is sought such that $f(S^*) \leq f(S)$, for all $S \in F$. The ground set E, the set of feasible solutions F, and the objective function f are defined for each specific problem. Many combinatorial optimization problems are computationally intractable, i.e. they fall into the category of NP-hard problems [32].

Paola Festa
Department of Mathematics and Applications, University of Napoli Federico II,
80126 Napoli, Italy
e-mail: paola.festa@unina.it

Mauricio G.C. Resende
Algorithms and Optimization Research Department, AT&T Labs Research,
Florham Park, NJ 07932 USA
e-mail: mgcr@research.att.com

E.-G. Talbi (Ed.): Hybrid Metaheuristics, SCI 434, pp. 135–155.
springerlink.com

Much progress has been made in the direction of exact methods for combinatorial optimization, such as branch and bound, branch and cut, and dynamic programming [72, 76]. These methods, however, suffer from the *curse of dimensionality*, i.e. they tend to break down as the size of the instance being solved increases. Likewise, approximation algorithms [74, 75], which provide a guaranteed suboptimal solution to hard combinatorial optimization problems, have also experienced significant progress. Although interesting in theory, approximation algorithms are often outperformed in practice by more straightforward heuristics with no particular performance guarantees.

Metaheuristics [33, 36] are general high-level procedures that coordinate simple heuristics and rules to find good (often optimal) approximate solutions to combinatorial optimization problems. They include genetic algorithms, simulated annealing, tabu search, scatter search, ant colonies, variable neighborhood search, GRASP, and path-relinking . There are many ways to classify metaheuristics. These include, trajectory-based versus population-based, nature-inspired versus non-nature inspired, memoryless versus memory-based, etc. Genetic algorithms, for example, are nature-inspired, population-based, with memory. Tabu search are trajectory-based with memory. GRASP is trajectory-based.

Hybrid metaheuristics combine one or more algorithmic ideas from different metaheuristics and sometimes even from outside the traditional field of metaheuristics. The main motivation to hybridize metaheuristics is to make up for the shortcomings of one metaheuristic with special characteristics of the other. In this chapter, we consider the hybridization of two metaheuristics: GRASP and path-relinking.

GRASP, or greedy randomized adaptive search procedures [25, 26, 30, 31, 59], is a metaheuristic for combinatorial optimization. GRASP heuristics are multistart procedures which apply local search to a set of starting solutions generated with a randomized greedy algorithm or semi-greedy method. The best local optimum found over the iterations is returned as the heuristic solution. Since GRASP iterations are independent of one another, GRASP heuristics do not make use of solutions produced throughout the search, i.e. they do not have any memory mechanism.

One way to add memory to GRASP is its hybridization with path-relinking. Path-relinking [35, 60, 63] is a search intensification procedure that explores paths in the neighborhood solution space connecting two good-quality solutions. A local search procedure is applied to the best solution found in the path and the local optimum found is returned as the solution of path-relinking.

This chapter describes basic concept of GRASP, path-relinking, and the hybridization of GRASP with path-relinking. In Section 5.2 we describe the main building blocks of GRASP. In Section 5.3 we consider path-relinking and, in Section 5.4, address issues related to the hybridization of GRASP with path-relinking and evolutionary path-relinking. A hybridization of GRASP with path-relinking and Lagrangean relaxation is discussed in Section 5.5. In Section 5.6 we consider parallel implementation of GRASP with path-relinking heuristics. Finally, concluding remarks are made in Section 5.7.

5.2 GRASP

Given a feasible solution $S \in F$ of a combinatorial optimization problem, a neighborhood $N(S)$ of S is a subset of F such that each element in $N(S)$ is "close" to S and can be obtained applying some elementary operation (or move) to S that changes one or more components of S. Consider the search space graph $G = (F,M)$, where the node set F is the set of feasible solutions and the edges in the set M correspond to moves in the neighborhood structure, i.e. $(S,S') \in M$ if and only if $S,S' \in F$, $S \in N(S')$, and $S' \in N(S)$.

Local search seeks a locally optimum solution in G, i.e. a solution $\hat{S} \in F$ such that $f(\hat{S}) \leq f(S)$, for all $S \in N(\hat{S})$. It starts from some solution $S^0 \in F$. At any iteration k, it seeks an improving solution $S^{k+1} \in N(S^k)$ such that $f(S^{k+1}) < f(S^k)$. On one hand, if a *first-improving* strategy is used, any improving solution S^{k+1} can be accepted. On the other hand, when a *best-improving* strategy is adopted, the improving solution S^{k+1} is the best-valued in the neighborhood, i.e. $f(S^{k+1}) = \min\{f(S) : S \in N(S^k)\}$. Local search terminates when a locally optimum solution is found. The effectiveness of local search depends strongly on the structure of the solution space graph $G = (F,M)$, the objective function f, and the starting solution $S^0 \in F$.

When designing a local search algorithm, one has the flexibility to design different neighborhoods and to select different starting solutions. Usually there is less flexibility in selecting an objective function. Some attention is needed in the design of neighborhoods since the complexity of each iteration k of local search is $O(|N(S^k)|)$. A neighborhood that is exponentially large will result in a local search with exponentially large computational complexity. Another cause of exponential computational complexity in local search is an exponentially small reduction in the objective function value when moving from a solution to a neighbor.

Since it is possible to select the starting solution S^0, a possible strategy is a *multi-start* algorithm, where local search is applied to a series of starting solutions $S_1^0, S_2^0, \ldots, S_q^0$ and the best local optimum found by the procedure is returned.

A straightforward way to implement such a multi-start algorithm is to generate each starting solution at random. A drawback to this approach is the fact that the quality of randomly-generated solutions is not very good and the number of moves needed to reach a global optimum is usually large. Not only does this result in long running times, it also increases the chance that local search will encounter a suboptimal local optimum along the way and get trapped there. The number of local optima with better cost than a randomly generated solution is usually larger than the number of local optima with better cost than a greedy solution.

A greedy algorithm builds a solution to a combinatorial optimization problem, one element of the ground set at a time. Given a partial solution, all possible candidate elements of the ground set (i.e. those elements that can be added to the partial solution without causing infeasibility) are ranked according to a myopic benefit associated with their inclusion in the solution and the next element to be added to the solution is one among the best-valued. Using a greedy algorithm to generate starting solutions for a multi-start algorithm is not recommended since the generated solutions would differ very little one from another. However, a good characteristic

Fig. 5.1 Pseudo-code of a generic GRASP

```
begin GRASP
1   f* ← ∞;
2   while stopping criterion not satisfied do
3       S ← RandomizedGreedy(·);
4       if S is not feasible then
5           S ← Repair(S);
6       end-if
7       S ← LocalSearch(S);
8       if f(S) < f* then
9           S* ← S;
10          f* ← f(S);
11      end-if
12  end-while
13  return S*;
end
```

of greedy solutions is their quality. Usually, fewer moves are needed to go from a greedy solution to a locally optimum than what is needed to go to a local optimum from a randomly generated solution.

A tradeoff between a greedy solution and a random solution is a *semi-greedy* or *randomized greedy* solution [38]. A semi-greedy heuristic is also a constructive procedure that builds a solution, one element of the ground set at a time. Like a greedy algorithm, in a semi-greedy algorithm, all possible candidate elements are ranked according to a myopic benefit associated with their inclusion in the solution. Instead of selecting one among the best-valued elements as the next one to be added to the solution, a restricted candidate list (RCL) is built with a set of good-valued candidates. One element from the RCL is selected at random and is added to the partial solution.

Hart and Shogan [38] proposed a multi-start procedure that uses a semi-greedy method but without local search. GRASP is a multi-start procedure which uses a semi-greedy method to generate starting solutions for local search. Since solutions produced by the algorithm of Hart and Shogan are not necessarily local optima, GRASP solutions are almost always better than semi-greedy solutions.

Figure 5.1 shows pseudo-code for a generic GRASP. GRASP iterations are carried out in lines 2 to 12. In line 3, the procedure attempts to build a feasible semi-greedy solution. Since this is not always possible because there is no backtracking in the greedy algorithm, a repair procedure may have to be applied in line 5 to achieve feasibility. An example of such a case can be seen in the GRASP for the generalized quadratic assignment problem of Mateus et al. [46]. A feasible solution S is used as the starting solution for the local search in line 7. If the local optimum S is better than the incumbent, then, in lines 9 and 10, it is saved as S^* and its objective function value as f^*. In line 13, the best solution found over all GRASP iterations is returned as the GRASP solution.

Fig. 5.2 Pseudo-code of the semi-greedy GRASP construction phase

```
begin GreedyRandomized
1   S ← ∅;
2   Initialize set of candidates C;
3   Evaluate the incremental cost of candidates;
4   while C ≠ ∅ do
5       Build the RCL;
6       Select s ∈ RCL at random;
7       S ← S ∪ {s};
8       Update C;
9       Reevaluate the incremental costs;
10  end-while
11  return S or indication that S is infeasible;
end
```

5.2.1 GRASP Construction

The GRASP construction phase in line 3 of the pseudo-code of Figure 5.1 combines greedy and randomized characteristics. The first implementations of GRASP made use of the semi-greedy algorithms of Hart and Shogan [38]. Figure 5.2 shows pseudo-code for a generic version of the semi-greedy algorithm of Hart and Shogan.

The semi-greedy construction builds a solution S, one element at a time. In line 1 of the pseudo-code, solution S is initialized empty. The elements of the ground set than can be feasibly added to the solution are called candidates. This set is initialized in line 2 and the costs of adding each candidate element to the solution is determined in line 3. The solution is built in the loop in lines 4 to 10. This loop is repeated while there remain candidate elements. When $C = \emptyset$, solution S can be either feasible or not. In the case that S is infeasible, a repair procedure will need to be called in the main GRASP procedure. Otherwise S is returned in line 11. In line 5, a restricted candidate list (RCL) is set up from which an element s is selected at random in line 6. This element is added to the partial solution in line 7. In line 8 the candidate set C is updated to reflect the inclusion of s in S. Finally, in line 9 the incremental costs are computed for each element of C.

Hart and Shogan [38] proposed two ways to construct the RCL. The first, called *cardinality based* , takes as input a parameter k and places the k elements with best incremental cost in the RCL. The second scheme is called *value based*. Let c^{min} and c^{max} denote, respectively, the minimum and maximum incremental cost of the candidate elements and let α be a real number in the interval $[0,1]$. A threshold $\tau = c^{min} + \alpha \cdot (c^{max} - c^{min})$ is computed and all candidate elements having incremental cost at most τ are placed in the RCL. Notice that the parameter α controls the amount of randomness and greediness in the construction process. If $\alpha = 0$, the construction is purely greedy. If $\alpha = 1$, the construction is random. By controlling the value of α, the algorithm designer can control how much greediness and/or

randomness characterizes the construction which in turn controls the intensification and diversification of the search.

One way to mix intensification and diversification is to randomly generate a different α at each GRASP iteration. Prais and Ribeiro [54] proposed a scheme they call *Reactive GRASP* in which the parameter α is self-tuned to favor values which resulted in better quality solutions in previous GRASP iterations. They define $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ to be the set of possible values for α. Initially, the probability of choosing a value α_i is $p_i = 1/m$, $i = 1, \ldots, m$. Furthermore, let f^* be the objective function value of the incumbent solution and let A_i be the average value of all solutions found using $\alpha = \alpha_i$, $i = 1, \ldots, m$. The selection probabilities are periodically recomputed by taking $p_i = q_i / \sum_{j=1}^{m} q_j$, with $q_i = f^*/A_i$ for $i = 1, \ldots, m$. The value of q_i will be larger for values of $\alpha = \alpha_i$ that lead, on average, to the best solutions. Larger values of q_i correspond to more suitable values for the parameter α. The probabilities associated with these more appropriate values will then increase when they are reevaluated. This reactive strategy is not limited to semi-greedy procedures where membership in the RCL depends on relative quality. It can be extended to the other greedy randomized construction schemes, all of which need to balance greediness with randomization.

In addition to the semi-greedy construction scheme, other alternative greedy randomized construction criteria have been proposed. Three such alternatives are the *random plus greedy*, the *sampled greedy* [61], and the construction by *cost perturbation* [15] schemes.

In random plus greedy, the first p components of the constructed solution are selected at random, one at a time. The remaining components are then added to the solution in a greedy fashion. In this scheme, parameter p controls the amount of randomness and/or greediness in the solution. Small values of p result in a greedy-like construction while large values of p correspond to a random-like construction.

Sampled greedy also makes use of a parameter p to control the amount of greediness and/or randomness in the construction process. At each step of sampled greedy construction process the procedure builds a RCL by sampling $\min\{p, |C|\}$ elements of the candidate set C. The incremental cost associated with adding each element of the RCL into the solution is evaluated. An element with the best-valued incremental cost is added to the partial solution. The balance between greediness and randomness is controlled by the value of parameter p. Small values of p lead to solutions constructed in a more random fashion while large values of p lead to solutions constructed in a more greedy fashion.

Construction by cost perturbation makes use of the problem data to balance the amount of randomness and greediness in the construction process. Some construction algorithm, such as, for example, an approximation algorithm, is applied to the problem where the data is randomly perturbed. The constructed solution is then evaluated using the original data. This way, by controlling the amount of perturbation, the construction will result in either a more random construction or a more greedy one.

5.2.2 Other Local Search Strategies

In addition to the first-improvement and best-improvement local search scheme described earlier in Section 5.2, other hybrid schemes have been proposed. These involve the replacement of the above mentioned local search schemes with more sophisticated local improvement methods, such as variable neighborhood descent [7, 23, 44, 67, 68], variable neighborhood search [15, 29], tabu search [16, 19, 40, 73], simulated annealing [18, 42], iterated local search [69], and very large scale neighborhood search [34].

5.2.3 Stopping Criteria

As any multi-start procedure, GRASP iterates until some stopping criterion is satisfied. Such criteria could be maximum number of iterations, maximum number of iterations without improvement of the incumbent solution, maximum running time, or solution quality at least as good as a given target value. With the exception of the last criterion, all other rules suffer from the same drawback, i.e. they cannot provide any information regarding the quality of the solution returned.

Stochastic-based stopping rules for GRASP and similar stochastic local search algorithms have been proposed, e.g. [9, 13, 21, 39, 49], but computational studies with these proposals are lacking.

Ribeiro et al. [66] study the distribution of solution values obtained by two GRASP procedures. For both procedures, the authors show that these solution values fit a normal distribution. With this observation they propose a probabilistic stopping rule for GRASP.

Let $f_1, f_2, \ldots, f_k$ be a sample formed by the first k solution values generated by GRASP. Furthermore, let μ^k and σ^k be, respectively, the estimated mean and the standard deviation of the sample. Define X to be the random variable representing the value of the local minimum found at each iteration. We assume that $X \sim N(\mu^k, \sigma^k)$, i.e. X is normally distributed with mean μ^k and standard deviation σ^k. Let $f_X^k(\cdot)$ and $F_X^k(\cdot)$ be, respectively, the probability density and the cumulative probability distribution function of X. If UB^k is the smallest solution value over the first k GRASP iterations, the probability of finding a solution at least as good UB^k in the next iteration can be estimated as $F_X^k(UB^k) = \int_{-\infty}^{UB^k} f_N^x(\tau)d\tau$. This probability is always reevaluated when the incumbent solution improves. It is reevaluated periodically even if no change in the value of the incumbent is observed. For a given threshold value β, the Ribeiro et al. probabilistic stopping rule is to stop the GRASP iterations whenever $F_X^k(UB^k) \leq \beta$. The pseudo-code in Figure 5.3 shows a GRASP with the probabilistic stopping rule.

Fig. 5.3 Pseudo-code of a generic GRASP with a probabilistic stopping rule

```
begin GRASP(β)
1   f* ← ∞; k ← 0;
2   repeat
3       S ← RandomizedGreedy(·);
4       if S is not feasible then
5           S ← Repair(S);
6       end-if
7       S ← LocalSearch(S);
8       if f(S) < f* then
9           S* ← S;
10          f* ← f(S);
11      end-if
12      k ← k + 1;
13      f_k ← f(S);
14      UB^k ← f(S*);
15      Update μ^k and σ^k of f_1, ..., f_k;
16      Compute F_X^k(UB^k) = ∫_{-∞}^{UB^k} f_N^x(τ)dτ;
17  until F_X^k(UB^k) < β
18  return S*;
end
```

5.3 Path-Relinking

From Section 5.2 recall the search space graph $G = (F, M)$, where the node set F is the set of feasible solutions and the edges in the set M correspond to moves in the neighborhood structure, i.e. $(S, S') \in M$ if and only if $S, S' \in F$, $S \in N(S')$, and $S' \in N(S)$. Given two solutions $S, T \in F$, the *path-relinking* operator [35] explores a path $\mathscr{P}(S,T)$ in G connecting S and T with the objective of finding solutions $S^* \in \mathscr{P}(S,T)$ for which $f(S^*) < \min\{f(S), f(T)\}$. If both S and T are good-quality solutions, then one can think of path-relinking as a search intensification procedure, which explores regions of the solution space spanned by both S and T.

Suppose path-relinking is to be done between two solutions $S \in F$ and $T \in F$. Let S be called the *initial* solution and T the *guiding* solution. One or more paths connecting these solutions in G can be explored. Local search can be applied to the best solution in each of these paths since there is no guarantee as to the local optimality of the best solution in the path.

Let $S' \in F$ be some solution in $\mathscr{P}(S,T)$. During path-relinking not all solutions in $N(S')$ are allowed to follow S' on the path $\mathscr{P}(S,T)$. Path-relinking restricts the choice to those solutions in $N(S')$ that share more attributes, or elements, with T than S' does. We denote by $N_T(S')$ this restricted neighborhood which consists of all neighbors of S' obtained by introducing into S' attributes of T not present in S'. To select the solution that follows S' on $\mathscr{P}(S,T)$, the most common choice is the greedy choice, i.e. the best-valued solution in $N_T(S')$.

Fig. 5.4 Pseudo-code of a greedy path-relinking operator

```
begin PathRelinking(S,T)
1   f* ← min{f(S),f(T)};
2   S* ← argmin{f(S),f(T)};
3   S' ← S;
4   while |Δ(S',T)| > 1 do
5       S_δ = argmin{f(Ŝ) | Ŝ ∈ N_T(S')};
6       if f(S_δ) < f* then
7           S* ← S_δ;
8           f* ← f(S_δ);
9       end-if
10      S' ← S_δ;
11  end-while
12  S* ← LocalSearch(S*);
13  return S*;
end
```

Let $\Delta(S',T)$ be the set of attributes present in T but not in S'. Introducing in S' any element $\delta \in \Delta(S',T)$ leads to a solution $S_\delta \in N_T(S')$ that can be reached by traversing edge $(S',S_\delta) \in M$. Figure 5.4 shows a pseudo-code for a basic greedy path-relinking operator. This operator scans a path from the initial solution S to the guiding solution T. In the first two lines, the best solution S^* and its value f^* are initialized and in line 3 the current solution S' is initialized to the initial solution S. The loop from line 4 to line 11 is repeated while there are attributes in the guiding solution that are not present in the current solution S'. Among all solutions in the restricted neighborhood $N_T(S')$ of S', a best-valued solution S_δ is selected in line 5. If this solution is the best seen so far, it and its value are recorded in lines 7 and 8. The current solution S' is updated in line 10 to S_δ. After examining the entire path from S to T, local search is applied to the best solution in line 12 and the resulting local optimum is returned as the solution of path-relinking in line 13.

5.3.1 Flavors of Path-Relinking

The scheme shown in the pseudo-code of Figure 5.4 can be implemented as different variants of path-relinking, including forward, backward, back and forward, mixed, and greedy randomized. In *forward* path-relinking, the starting solution S' is such that $S' = \text{argmax}\{f(S),f(T)\}$. Conversely, in *backward* path-relinking, the starting solution S' is such that $S' = \text{argmin}\{f(S),f(T)\}$. When carrying out path-relinking, the neighborhood of the initial solution is explored more thoroughly than that of the guiding solution. Since the quality of the initial solution in backward path-relinking is better than that of the initial solution in forward path-relinking, backward path-relinking usually performs better than forward path-relinking. Better yet is *back and forward* path-relinking, where a backward path-relinking is applied first and then a forward path-relinking follows. Back and forward path-relinking finds, by

```
begin MixedPathRelinking(S,T)
1   f* ← min{f(S),f(T)};
2   S* ← argmin{f(S),f(T)};
3   S' ← S;
4   while |Δ(S',T)| > 1 do
5       S_δ = argmin{f(Ŝ) | Ŝ ∈ N_T(S')};
6       if f(S_δ) < f* then
7           S* ← S_δ;
8           f* ← f(S_δ);
9       end-if
10      T' ← S_δ;
11      S' ← T;
12      T ← T';
13  end-while
14  S* ← LocalSearch(S*);
15  return S*;
end
```

Fig. 5.5 Pseudo-code of mixed path-relinking operator

definition, solutions that are at least as good as either backward or forward path-relinking, but at the expense of taking about twice as long as either.

In contrast to back and forward path-relinking, a less expensive way to explore the neighborhoods of the initial and guiding solutions is with *mixed path-relinking* [35, 65]. In mixed path-relinking, the roles of initial and guiding solutions are exchanged after each move. This way, two paths are generated, one emanating from the initial solution and the other from the guiding solution. The paths eventually meet at some solution about half way between the two input solutions. A pseudo-code for mixed path-relinking is shown in Figure 5.5.

If ties are broken deterministically in greedy path-relinking, the procedure will always generate the same path when applied to a given input pair $\{S,T\}$. Since the number of paths connecting the input pair grows exponentially with $|\Delta(S,T)|$, exploring a single path can be limiting. *Greedy randomized adaptive* path-relinking [12, 24] uses a semi-greedy move selection strategy that enables exploration of different paths when applied to the same input pair. Instead of making the greedy move choice as in line 5 of the pseudo-code in Figure 5.4, greedy randomized adaptive path-relinking builds a restricted candidate list of moves, one of which is selected at random to lead to the next solution along the path.

Good-quality solutions tend to be located near other good-quality solutions. Consequently good solutions found by path-relinking are usually found near S or T. Resende et al. [56] showed this was the case for the max-min diversity problem (see Figure 5.6). In *truncated* path-relinking, only a partial path is explored. The search is limited to solutions where only a small portion of the attributes of the guiding solution are introduced and consequently the running time to apply path-relinking is reduced.

Mateus et al. [46] observed that path-relinking can fail when $N_T(S') = \emptyset$ in line 5 of the pseudo-code of Figure 5.4. In such a case a repair procedure is applied to S' in an attempt to move from S' to some solution S'' such that $N_T(S'') \neq \emptyset$.

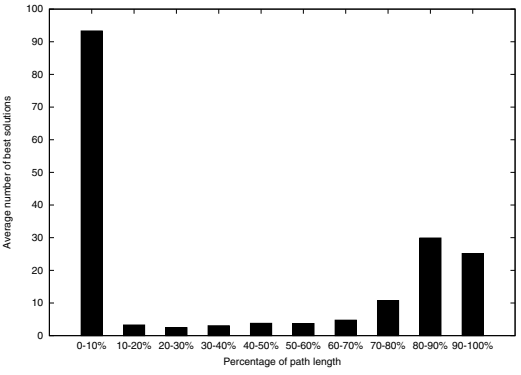

Fig. 5.6 Average number of best solution found at different depths of the path from the initial solution to the guiding solution on instances of the max-min diversity problem [56].

5.3.2 Path-Relinking and Elite Sets

An *elite set* or *pool* $\mathscr{E}$ of solutions is a fixed-size set of good-quality and diverse solutions. The quality of a solution S is with respect to its objective function value $f(S)$ while the diversity between two solutions S and T is with respect to $\Delta(S,T)$. When initially populating $\mathscr{E}$, a candidate solution S is inserted into $\mathscr{E}$ if it differs from all other solutions already in $\mathscr{E}$, i.e. if $|\Delta(S,T)| \neq 0$, for all $T \in \mathscr{E}$.

If a solution S is inserted into $\mathscr{E}$ when it is already full, it must replace some solution $T \in \mathscr{E}$. A candidate solution S is inserted into $\mathscr{E}$ if one of the following two conditions is satisfied:

1. $f(S) < f(T)$ for all $T \in \mathscr{E}$;
2. Condition (1) does not hold but $f(S) < f(T)$ for some $T \in \mathscr{E}$ and $|\Delta(S,T)| > \varepsilon$ for all $T \in \mathscr{E}$, where ε is an input parameter used to control the diversity of the elite solutions.

Once a solution S is accepted to enter the elite set, it must replace a solution $T \in \mathscr{E}$. T should be such that its replacement by S in $\mathscr{E}$ results in an elite set with smaller average objective function value and minimizes the impact on diversity of $\mathscr{E}$. A strategy [61] that achieves this goal is to select, among all solutions $T \in \mathscr{E}$ that have worse objective function value than S, the one that is most similar to S, i.e. select

$$T = \operatorname*{argmin}_{T' \in \mathscr{E}} \{|\Delta(S,T')| \text{ such that } f(T') > f(S)\}.$$

One way to combine path-relinking and elite sets is through *evolutionary* path-relinking [61]. Given an initial elite set, evolutionary path-relinking evolves the elite set applying the path-relinking operator among pair of elite set solutions. Two variants of evolutionary path-relinking have been proposed. The first, proposed in Resende and Werneck [61], works with a series of elite sets. At step k, pairs of solutions in $\mathscr{E}_k$ are relinked one pair at a time. The resulting solution of each path-relinking operation is a candidate for inclusion in elite set $\mathscr{E}_{k+1}$. The acceptance and replacement selection rules described above are used to determine if a candidate

is accepted by $\mathscr{E}_{k+1}$ and to determine which elite solution in $\mathscr{E}_{k+1}$ it will replace. The procedure stops when the best solution in elite set $\mathscr{E}_{k+1}$ has the same objective function value as the best solution in elite set $\mathscr{E}_k$. The second scheme, proposed by Resende et al. [56] works with a single elite set $\mathscr{E}$. While there remain pairs of solutions in $\mathscr{E}$ that have not yet been relinked, the path-relinking operator is applied to the pair and the resulting solution is a candidate to enter $\mathscr{E}$. The acceptance and replacement selection rules are applied as described above.

5.4 GRASP with Path-Relinking and Evolutionary Path-Relinking

Laguna and Martí [41] proposed the first hybridization of GRASP with path-relinking. In their implementation, the elite set is made up of only three solutions. Each GRASP solution (local minimum obtained by the local search procedure) is relinked with a randomly chosen elite set solution. If the solution resulting from the path-relinking operator is better than the best elite solution, it replaces the worst elite solution.

Since 1999, much work has been done to improve the hybridization of GRASP with path-relinking [58, 59]. The pseudo-code in Figure 5.7 is a template for implementation of GRASP with path-relinking heuristics. The iterations of GRASP with path-relinking are carried out in lines 2 to 17. Lines 3 to 7 comprise the two phases of GRASP, producing a locally optimal solution S. In the case that the elite set $\mathscr{E}$ is not yet full, then in lines 9 to 11 S is added to $\mathscr{E}$ if it is different from all elite set solutions. In the case that the elite set is full, an elite solution T is selected in line 13 and path-relinking is applied to the pair S,T in line 14, and finally, in line 15, the elite set $\mathscr{E}$ is updated, i.e. solution S is considered for inclusion in $\mathscr{E}$ and if accepted, it will replace some existing solution in $\mathscr{E}$. In line 18, the GRASP with path-relinking procedure returns the best-quality solution S^* among all elite solutions.

GRASP with path-relinking maintains a elite set of diverse good-quality solutions found during the search. Periodically evolutionary path-relinking can be applied to the elite set with the objective of improving the quality of some of the elite set solutions. The pseudo-code in Figure 5.8 shows how to modify GRASP with path-relinking in order to obtain GRASP with evolutionary path-relinking. If a criterion for evolutionary path-relinking is triggered (line 3) then evolutionary path-relinking is applied to the current elite set in line 4. This criterion is usually a number of iterations since the last call to evolutionary path-relinking. Since the same pair of elite solutions may be relinked several times (in different calls to evolutionary path-relinking), evolutionary path-relinking is usually implemented in the inner loop (line 4) using the greedy randomized adaptive path-relinking operator. That way if a pair is relinked more than once, a different solution can result from the path-relinking operator. Finally, at the conclusion of the GRASP iterations, evolutionary path-relinking is applied a final time in line 20 to possibly improve

Fig. 5.7 Pseudo-code of a GRASP with path-relinking

```
begin GRASP+PR
1   ℰ ← ∅;
2   while stopping criterion not satisfied do
3       S ← RandomizedGreedy(·);
4       if S is not feasible then
5           S ← Repair(S);
6       end-if
7       S ← LocalSearch(S);
8       if ℰ is not full then
9           if Δ(S,T) ≠ 0, for all T ∈ ℰ then
10              ℰ ← ℰ ∪ {S};
11          end-if
12      else
13          Select T ∈ ℰ;
14          S ← PathRelinking(S,T);
15          ℰ ← UpdateElite(ℰ,S);
16      end-if
17  end-while
18  return S* = argmin{f(S) | S ∈ ℰ};
end
```

the elite set and allow the algorithm to output a potentially better solution S^* in line 21.

In a paper on GRASP with path-relinking for the three-index assignment problem, Aiex et al. [2] applied path-relinking between all pairs of the elite set as search intensification and as post-processing. Resende and Werneck [61, 62] applied evolutionary path-relinking in a post-processing phase in GRASP with path-relinking heuristics for the p-median and uncapacitated facility location problems. Andrade and Resende [2] applied evolutionary path-relinking between the two best elite solutions and all other elite solutions as a search intensification in a GRASP with path-relinking for a network migration problem. Resende et al. [56] showed through experimental results that a GRASP with evolutionary path-relinking for a max-min diversity problem could outperform heuristics based on pure GRASP with path-relinking, simulated annealing, and tabu search.

5.5 Hybrid GRASP Lagrangean Heuristic

Pessoa et al. [51, 52] proposed LAGRASP, a hybrid heuristic combining GRASP with path-relinking and subgradient optimization to solve the set k-covering problem . Their algorithm extends the Lagrangean heuristic for set covering of Beasley [11] to the case of set k-covering. In addition, instead of following Beasley and using a simple greedy heuristic as the primal heuristic, Pessoa et al. use a GRASP with path-relinking heuristic in which Lagrangean reduced costs are used in place of the original costs.

```
begin GRASP+evPR
1  ℰ ← ∅;
2  while stopping criterion not satisfied do
3      if evPR criterion triggered then
4          ℰ ← evPathRelinking(ℰ);
5      S ← RandomizedGreedy(·);
6      if S is not feasible then
7          S ← Repair(S);
8      end-if
9      S ← LocalSearch(S);
10     if ℰ is not full then
11         if Δ(S,T) ≠ 0, for all T ∈ ℰ then
12             ℰ ← ℰ ∪ {S};
13         end-if
14     else
15         Select T ∈ ℰ;
16         S ← PathRelinking(S,T);
17         ℰ ← UpdateElite(ℰ,S);
18     end-if
19 end-while
20 ℰ ← evPathRelinking(ℰ);
21 return S* = argmin{f(S) | S ∈ ℰ};
end
```

Fig. 5.8 Pseudo-code of a GRASP with evolutionary path-relinking

The comparison of LAGRASP with pure GRASP with path-relinking showed that LAGRASP was able to find much better quality solutions than the pure GRASP with path-relinking. Furthermore, the comparison of different variants of LAGRASP showed that, by properly tuning its parameters, it is possible to obtain a good trade-off between solution quality and running time. Extensive experiments on 135 instances showed that LAGRASP can take advantage of randomization to make better use of dual information provided by subgradient optimization than Beasley's algorithm. As a consequence, LAGRASP is able to discover better solutions and to escape from locally optimal solutions after the stabilization of the lower bounds, whereas the greedy Lagrangean heuristic of Beasley [11] fails to find new improving solutions.

5.6 Parallel GRASP with Path-Relinking

Multiple-walk independent-thread parallel implementations distribute the GRASP with path-relinking iterations over the processors. Each thread performs $i_{\max}/p$ iterations, where $i_{\max}$ is the total number of iterations and p is the number of processors. As opposed to pure GRASP, were linear speedup is usually observed, multiple-walk independent-thread parallel implementations of GRASP with path-relinking have had mixed results. For example, Aiex et al. [2] showed linear speedups for the 3-index assignment problem whereas for the job-shop scheduling problem, Aiex et al. [1] showed sublinear speedups.

In this section, we focus on multiple-walk cooperative-thread schemes for implementing GRASP with path-relinking in parallel. In multiple-walk cooperative-thread schemes superlinear speedups have been observed (see, e.g. [1, 2, 3]). Two basic mechanisms have be used to implement multiple-walk cooperative-thread GRASP with path-relinking heuristics.

In *distributed strategies* [1, 3], each thread maintains its own pool of elite solutions. Each iteration of each thread consists initially of a GRASP construction, followed by local search. Then, the local optimum is combined with a randomly selected element of the thread's pool using path-relinking. The output of path-relinking is then tested for insertion into the pool. If accepted, the solution is sent to the other threads, where it is tested for insertion into the other pools. Collaboration takes place at this point. Though there may be some communication overhead in the early iterations, this tends to ease up as pool insertions become less frequent.

The second mechanism is the one used in *centralized strategies* [45, 64, 65], in which a single pool of elite solution is used. As before, each GRASP iteration performed at each thread starts by the construction and local search phases. Next, an elite solution is requested and received from the centralized pool. Once path-relinking is performed, the solution obtained as the output is sent to the pool and tested for insertion. Collaboration takes place when elite solutions are sent from the pool to other processors different from the one that originally computed it.

In both the distributed and the centralized strategies each processor has a copy of the sequential algorithm and a copy of the data. One processor acts as the master, reading and distributing the problem data, generating the seeds which will be used by the pseudo-random number generators at each processor, distributing the iterations, and collecting the best solution found by each processor. In the case of a distributed strategy, each processor has its own pool of elite solutions and all available processors perform GRASP iterations. In the case of a centralized strategy, one processor does not perform GRASP iterations and is used exclusively to store the pool and to handle all operations involving communication requests between the pool and the slaves.

5.7 Concluding Remarks

This chapter reviewed the hybridization of greedy randomized adaptive search procedures (GRASP) and path-relinking. As originally proposed in Feo and Resende [25, 26], GRASP does not make use of any memory structures. The hybridization of path-relinking with GRASP, proposed in Laguna and Martí [41], introduced memory structures in GRASP. Though path-relinking adds extra work to each iteration of GRASP (maintenance of the elite set and the path-relinking operation itself), the total number of iterations required to find a solution of a given quality more than compensates for this additional work, resulting in a higher probability that a target solution will be found in a given amount of search time. Figure 5.9 shows runtime distributions (time to target plots [4]) comparing implementations of pure GRASP

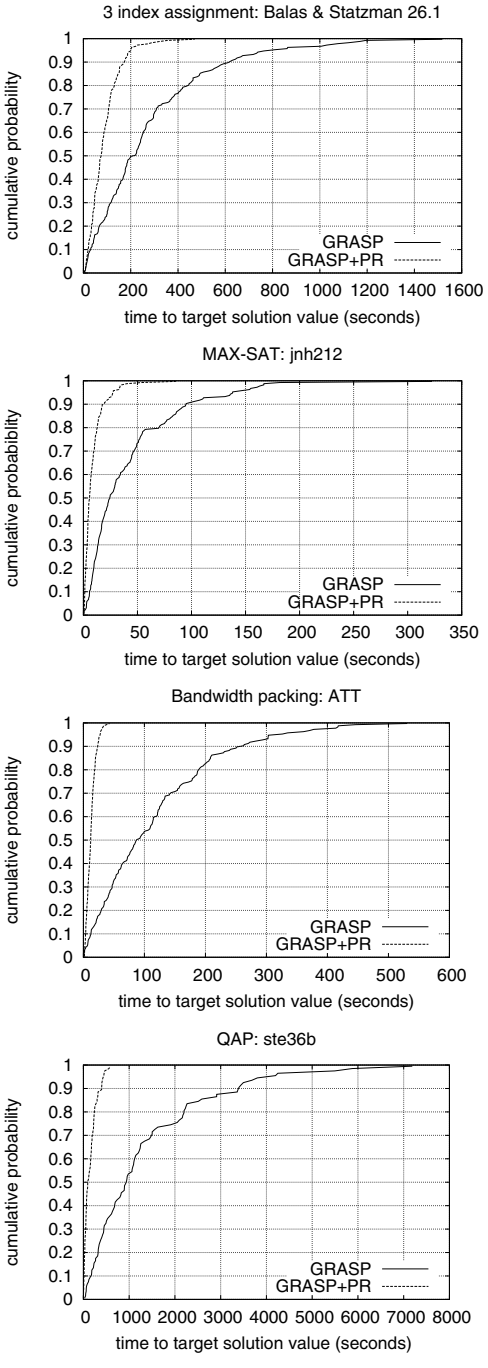

Fig. 5.9 Time to target plots comparing running times of pure GRASP and GRASP with path-relinking on four instances of distinct problem types: three index assignment [2], maximum satisfiability [27], bandwidth packing [57], and quadratic assignment [48].

and GRASP with path-relinking on four instances of distinct problem types: three index assignment [2], maximum satisfiability [27], bandwidth packing [57], and quadratic assignment [48]. The four plots are typical in the comparison of GRASP and GRASP with path-relinking in that:

- For a fixed running time, the probability that GRASP with path-relinking finds a solution at least as good as the target value is greater than the probability that pure GRASP will;
- For a fixed probability, the running time for GRASP with path-relinking to find a solution at least as good as the target value with that probability is smaller than the running time need for pure GRASP to find such a solution with the same probability.

Hybridization with path-relinking is now the standard approach to implementing GRASP.

We conclude this chapter with a list of applications of GRASP with path-relinking (which we do not intend to be exhaustive):

- Graph drawing [41];
- Job-shop scheduling [1], PBX migration scheduling [6], broadcast scheduling [17], network migration scheduling [7], machine scheduling [37], flowshop scheduling [70];
- Two-path network design [64], rural road network design [71], capacitated minimum spanning tree [73];
- Bandwidth packing [57], matrix bandwidth minimization [53], antibandwidth [22];
- Quadratic assignment [48], generalized quadratic assignment [46], three-index assignment [2], SONET ring assignment [10];
- Max-SAT [28], max-cut [29];
- p-median [61], uncapacitated facility location [62], health care facility location [50], capacitated clustering [20];
- Capacitated arc routing with time windows [55], traveling salesman problem [43];
- Production-distribution planning [14], assembly line sequencing [5], capacitated lot sizing [47];
- Maximum diversity [8], max-min diversity [56].

References

1. Aiex, R.M., Binato, S., Resende, M.G.C.: Parallel GRASP with path-relinking for job shop scheduling. Parallel Computing 29, 393–430 (2003)
2. Aiex, R.M., Pardalos, P.M., Resende, M.G.C., Toraldo, G.: GRASP with path-relinking for three-index assignment. INFORMS J. on Computing 17, 224–247 (2005)
3. Aiex, R.M., Resende, M.G.C.: Parallel strategies for GRASP with path-relinking. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solvers, pp. 301–331. Springer (2005)

4. Aiex, R.M., Resende, M.G.C., Ribeiro, C.C.: TTTPLOTS: A perl program to create time-to-target plots. Optimization Letters 1, 355–366 (2007)
5. Alpay, S.: GRASP with path relinking for a multiple objective sequencing problem for a mixed-model assembly line. International J. of Production Research 47, 6001–6017 (2009)
6. Andrade, D.V., Resende, M.G.C.: A GRASP for PBX telephone migration scheduling. In: Proceedings of The Eighth INFORMS Telecommunications Conference (2006)
7. Andrade, D.V., Resende, M.G.C.: GRASP with path-relinking for network migration scheduling. In: Proceedings of the International Network Optimization Conference (2007)
8. de Andrade, M.R.Q., de Andrade, P.M.F., Martins, S.L., Plastino, A.: GRASP with Path-Relinking for the Maximum Diversity Problem. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 558–569. Springer, Heidelberg (2005)
9. Bartkutė, V., Felinskas, G., Sakalauskas, L.: Optimality testing in stochastic and heuristic algorithms. Technological and Economic Development of Economy 12(1), 4–10 (2006)
10. Bastos, L.O., Ochi, L.S., Macambira, E.M.: GRASP with path-relinking for the SONET ring assignment problem. In: International Conference on Hybrid Intelligent Systems, Los Alamitos, CA, USA, pp. 239–244. IEEE Computer Society (2005)
11. Beasley, J.E.: A Lagrangian heuristic for set-covering problems. Naval Research Logistics 37, 151–164 (1990)
12. Binato, S., Faria Jr., H., Resende, M.G.C.: Greedy randomized adaptive path relinking. In: Sousa, J.P. (ed.) Proceedings of the IV Metaheuristics International Conference, pp. 393–397 (2001)
13. Boender, C.G.E., Rinnooy Kan, A.H.G.: Bayesian stopping rules for multistart global optimization methods. Mathematical Programming 37, 59–80 (1987)
14. Boudia, M., Louly, M.A.O., Prins, C.: A reactive GRASP and path relinking for a combined production-distribution problem. Computers and Operations Research 34, 3402–3419 (2007)
15. Canuto, S.A., Resende, M.G.C., Ribeiro, C.C.: Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks 38, 50–58 (2001)
16. Colomé, R., Serra, D.: Consumer choice in competitive location models: Formulations and heuristics. Papers in Regional Science 80, 439–464 (2001)
17. Commander, C.W., Butenko, S.I., Pardalos, P.M., Oliveira, C.A.S.: Reactive GRASP with path relinking for the broadcast scheduling problem. In: Proceedings of the 40th Annual International Telemetry Conference, pp. 792–800 (2004)
18. de la Peña, M.G.B.: Heuristics and metaheuristics approaches used to solve the rural postman problem: A comparative case study. In: Proceedings of the Fourth International ICSC Symposium on Engineering of Intelligent Systems, EIS 2004 (2004), http://www.x-cd.com/eis04/22.pdf
19. Delmaire, H., Díaz, J.A., Fernández, E., Ortega, M.: Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. INFOR 37, 194–225 (1999)
20. Deng, Y., Bard, J.F.: A reactive GRASP with path relinking for capacitated clustering. J. of Heuristics 17, 119–152 (2011)
21. Dorea, C.C.Y.: Stopping rules for a random optimization method. SIAM J. on Control and Optimization 28, 841 (1990)
22. Duarte, A., Martí, R., Resende, M.G.C., Silva, R.M.A.: GRASP with path relinking heuristics for the antibandwidth problem. Networks 58, 171–189 (2011)
23. Ribeiro, C.C., Vianna, D.S.: A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. International Transactions in Operational Research 12, 325–338 (2005)
24. Faria Jr., H., Binato, S., Resende, M.G.C., Falcão, D.J.: Transmission network design by a greedy randomized adaptive path relinking approach. IEEE Transactions on Power Systems 20, 43–49 (2005)

25. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8, 67–71 (1989)
26. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. of Global Optimization 6, 109–133 (1995)
27. Festa, P., Pardalos, P.M., Pitsoulis, L.S., Resende, M.G.C.: GRASP with path-relinking for the weighted MAXSAT problem. ACM J. of Experimental Algorithmics 11, 1–16 (2006)
28. Festa, P., Pardalos, P.M., Pitsoulis, L.S., Resende, M.G.C.: GRASP with path relinking for the weighted MAXSAT problem. J. of Experimental Algorithmics 11(2.4) (2007)
29. Festa, P., Pardalos, P.M., Resende, M.G.C., Ribeiro, C.C.: Randomized heuristics for the MAX-CUT problem. Optimization Methods and Software 7, 1033–1058 (2002)
30. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP, Part I: Algorithms. International Transactions in Operational Research 16, 1–24 (2009)
31. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP, Part II: Applications. International Transactions in Operational Research 16, 131–172 (2009)
32. Garey, M.R., Johnson, D.S.: Computers and intractability - A guide to the theory of NP-completeness. W.H. Freeman and Company (1979)
33. Gendreau, M., Potvin, J.-Y. (eds.): Handbook of Metaheuristics, 2nd edn. International Series in Operations Research & Management Science, vol. 146. Springer (2010)
34. Geng, Y., Li, Y., Lim, A.: A very large-scale neighborhood search approach to capacitated warehouse routing problem. In: 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2005), pp. 58–65 (2005)
35. Glover, F.: Tabu search and adaptive memory programming – Advances, applications and challenges. In: Barr, R.S., Helgason, R.V., Kennington, J.L. (eds.) Interfaces in Computer Science and Operations Research, pp. 1–75. Kluwer Academic Publishers (1996)
36. Glover, F., Kochenberger, G. (eds.): Handbook of Metaheuristics. Kluwer Academic Publishers (2003)
37. Gupta, S.R., Smith, J.S.: Algorithms for single machine total tardiness scheduling with sequence dependent setups. European J. of Operational Research 175, 722–739 (2006)
38. Hart, J.P., Shogan, A.W.: Semi-greedy heuristics: An empirical study. Operations Research Letters 6, 107–114 (1987)
39. Hart, W.E.: Sequential stopping rules for random optimization methods with applications to multistart local search. SIAM J. on Optimization, 270–290 (1998)
40. Laguna, M., González-Velarde, J.L.: A search heuristic for just-in-time scheduling in parallel machines. J. of Intelligent Manufacturing 2, 253–260 (1991)
41. Laguna, M., Martí, R.: GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS J. on Computing 11, 44–52 (1999)
42. Liu, X., Pardalos, P.M., Rajasekaran, S., Resende, M.G.C.: A GRASP for frequency assignment in mobile radio networks. In: Badrinath, B.R., Hsu, F., Pardalos, P.M., Rajasejaran, S. (eds.) Mobile Networks and Computing. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 52, pp. 195–201. American Mathematical Society (2000)
43. Marinakis, Y., Migdalas, A., Pardalos, P.M.: Multiple phase neighborhood search – GRASP based on Lagrangean relaxation, random backtracking Lin–Kernighan and path relinking for the TSP. J. of Combinatorial Optimization 17, 134–156 (2009)
44. Martins, S.L., Pardalos, P.M., Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures for the Steiner problem in graphs. In: Pardalos, P.M., Rajasejaran, S., Rolim, J. (eds.) Randomization Methods in Algorithmic Design. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 43, pp. 133–145. American Mathematical Society (1999)
45. Martins, S.L., Ribeiro, C.C., Rosseti, I.: Applications and Parallel Implementations of Metaheuristics in Network Design and Routing. In: Manandhar, S., Austin, J., Desai, U., Oyanagi, Y., Talukder, A.K. (eds.) AACC 2004. LNCS, vol. 3285, pp. 205–213. Springer, Heidelberg (2004)

46. Mateus, G.R., Resende, M.G.C., Silva, R.M.A.: GRASP with path-relinking for the generalized quadratic assignment problem. J. of Heuristics 17, 527–565 (2011)
47. Nascimento, M.C.V., Resende, M.G.C., Toledo, F.M.B.: GRASP with path-relinking for the multi-plant capacitated plot sizing problem. In: European J. of Operational Research (2008) (to appear)
48. Oliveira, C.A.S., Pardalos, P.M., Resende, M.G.C.: GRASP with Path-Relinking for the Quadratic Assignment Problem. In: Ribeiro, C.C., Martins, S.L. (eds.) WEA 2004. LNCS, vol. 3059, pp. 356–368. Springer, Heidelberg (2004)
49. Orsenigo, C., Vercellis, C.: Bayesian stopping rules for greedy randomized procedures. J. of Global Optimization 36(3), 365–377 (2006)
50. Pacheco, J.A., Casado, S.: Solving two location models with few facilities by using a hybrid heuristic: A real health resources case. Computers and Operations Research 32, 3075–3091 (2005)
51. Pessoa, L.S., Resende, M.G.C., Ribeiro, C.C.: A hybrid Lagrangean heuristic with GRASP and path-relinking for set k-covering. Technical report. AT&T Labs Research, Shannon Laboratory, Florham Park, NJ 07932 (2010)
52. Pessoa, L.S., Resende, M.G.C., Ribeiro, C.C.: Experiments with LAGRASP heuristic for set *k*-covering. Optimization Letters 5, 407–419 (2011)
53. Pinana, E., Plana, I., Campos, V., Martí, R.: GRASP and path relinking for the matrix bandwidth minimization. European J. of Operational Research 153, 200–210 (2004)
54. Prais, M., Ribeiro, C.C.: Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. INFORMS J. on Computing 12, 164–176 (2000)
55. Reghioui, M., Prins, C., Labadi, N.: GRASP with Path Relinking for the Capacitated arc Routing Problem with Time Windows. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 722–731. Springer, Heidelberg (2007)
56. Resende, M.G.C., Martí, R., Gallego, M., Duarte, A.: GRASP and path relinking for the max-min diversity problem. Computers and Operations Research 37, 498–508 (2010)
57. Resende, M.G.C., Ribeiro, C.C.: A GRASP with path-relinking for private virtual circuit routing. Networks 41, 104–114 (2003)
58. Resende, M.G.C., Ribeiro, C.C.: GRASP with path-relinking: Recent advances and applications. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solvers, pp. 29–63. Springer (2005)
59. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures: Advances and applications. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics, 2nd edn. International Series in Operations Research & Management Science, vol. 146, pp. 281–317. Springer (2010)
60. Resende, M.G.C., Ribeiro, C.C., Glover, F., Martí, R.: Scatter search and path-relinking: Fundamentals, advances, and applications. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics, 2nd edn. International Series in Operations Research & Management Science, vol. 146, pp. 87–107. Springer (2010)
61. Resende, M.G.C., Werneck, R.F.: A hybrid heuristic for the *p*-median problem. J. of Heuristics 10, 59–88 (2004)
62. Resende, M.G.C., Werneck, R.F.: A hybrid multistart heuristic for the uncapacitated facility location problem. European J. of Operational Research 174, 54–68 (2006)
63. Ribeiro, C.C., Resende, M.G.C.: Path-relinking intensification methods for stochastic local search algorithms. J. of Heuristics (2011) (to appear)
64. Ribeiro, C.C., Rosseti, I.: A parallel GRASP for the 2-path network design problem CICLing 2001. LNCS, vol. 2004, pp. 922–926 (2002)
65. Ribeiro, C.C., Rosseti, I.: Efficient parallel cooperative implementations of GRASP heuristics. Parallel Computing 33, 21–35 (2007)
66. Ribeiro, C.C., Rosseti, I., Souza, R.C.: Effective Probabilistic Stopping Rules for Randomized Metaheuristics: GRASP Implementations. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 146–160. Springer, Heidelberg (2011)
67. Ribeiro, C.C., Souza, M.C.: Variable neighborhood search for the degree constrained minimum spanning tree problem. Discrete Applied Mathematics 118, 43–54 (2002)

68. Ribeiro, C.C., Uchoa, E., Werneck, R.F.: A hybrid GRASP with perturbations for the Steiner problem in graphs. INFORMS J. on Computing 14, 228–246 (2002)
69. Ribeiro, C.C., Urrutia, S.: Heuristics for the mirrored traveling tournament problem. European J. of Operational Research 179, 775–787 (2007)
70. Ronconi, D.P., Henriques, L.R.S.: Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. Omega 37, 272–281 (2009)
71. Scaparra, M., Church, R.: A GRASP and path relinking heuristic for rural road network development. J. of Heuristics 11, 89–108 (2005)
72. Schrijver, A.: Theory of linear and integer programming. John Wiley & Sons, Ltd., West Sussex (1996)
73. Souza, M.C., Duhamel, C., Ribeiro, C.C.: A GRASP heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy. In: Resende, M.G.C., de Sousa, J.P. (eds.) Metaheuristics: Computer Decision-Making, pp. 627–658. Kluwer Academic Publisher (2004)
74. Vazirani, V.V.: Approximation algorithms. Springer, Berlin (2001)
75. Williamson, D.P., Shmoys, D.B.: The design of approximation algorithms. Cambridge University Press, New York (2011)
76. Wolsey, L.A., Nemhauser, G.L.: Integer and combinatorial optimization. John Wiley & Sons, Inc., New York (1999)