Hybrid GRASP Heuristics

Paola Festa and Mauricio G.C. Resende

Abstract. Experience has shown that a crafted combination of concepts of different metaheuristics can result in robust combinatorial optimization schemes and produce higher solution quality than the individual metaheuristics themselves, especially when solving difficult real-world combinatorial optimization problems. This chapter gives an overview of different ways to hybridize GRASP (Greedy Randomized Adaptive Search Procedures) to create new and more effective metaheuristics. Several types of hybridizations are considered, involving different constructive procedures, enhanced local search algorithms, and memory structures.

1 Introduction

Combinatorial optimization problems involve a finite number of alternatives: given a finite solution set X and a real-valued objective function $f : X \rightarrow R$, one seeks a solution $x^* \in X$ with $f(x^*) \leq f(x)$, $\forall\ x \in X$. Several combinatorial optimization problems can be solved in polynomial time, but many of them are computationally intractable in the sense that no polynomial time algorithm exists for solving it unless P = NP [27]. Due to the computational complexity of hard combinatorial problems, there has been an extensive research effort devoted to the development of approximation and heuristic algorithms, especially because many combinatorial optimization problems, including routing, scheduling, inventory and production planning, and

Paola Festa
Dept. of Mathematics and Applications, University of Napoli FEDERICO II, Naples, Italy
e-mail: paola.festa@unina.it

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

A. Abraham et al. (Eds.): Foundations of Comput. Intel. Vol. 3, SCI 203, pp. 75–100.
springerlink.com

facility location, arise in real-world situations such as in transportation (air, rail, trucking, shipping), energy (electrical power, petroleum, natural gas), and telecommunications (design, location).

To deal with hard combinatorial problems, heuristic methods are usually employed to find good, but not necessarily guaranteed optimal solutions. The effectiveness of these methods depends upon their ability to adapt to a particular realization, avoid entrapment at local optima, and exploit the basic structure of the problem. Building on these notions, various heuristic search techniques have been developed that have demonstrably improved our ability to obtain good solutions to difficult combinatorial optimization problems. One of the most promising of such techniques are usually called *metaheuristics* and include, but are not restricted to, simulated annealing [43], tabu search [28, 29, 32], evolutionary algorithms like genetic algorithms [36], ant colony optimization [19], scatter search [35, 45, 47], path-relinking [30, 31, 33, 34], iterated local search [8, 49], variable neighborhood search [37], and GRASP (Greedy Randomized Adaptive Search Procedures) [21, 22].

Metaheuristics are a class of methods commonly applied to suboptimally solve computationally intractable combinatorial optimization problems. The term metaheuristic derives from the composition of two Greek words: *meta* and *heuriskein*. The suffix 'meta' means 'beyond', 'in an upper level', while 'heuriskein' means 'to find'. In fact, metaheuristics are a family of algorithms that try to combine basic heuristic methods in higher level frameworks aimed at efficiently exploring the set of feasible solution of a given combinatorial problem. In [72] the following definition has been given:

> "A metaheuristic is an *iterative master process* that guides and modifies the operations of *subordinate heuristics* to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method."

Osman and Laporte [52] in their metaheuristics bibliography define a metaheuristics as follows:

> "A metaheuristic is formally defined as an *iterative generation process* which guides a *subordinate heuristic* by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions. "

In the last few years, many heuristics that do not follow the concepts of a single metaheuristic have been proposed. These heuristics combine one or more algorithmic ideas from different metaheuristics and sometimes even from outside the traditional field of metaheuristics. Experience has shown that a crafted combination of concepts of different metaheuristics can result in robust combinatorial optimization schemes and produce higher

solution quality than the individual metaheuristics themselves. These approaches combining different metaheuristics are commonly referred to as *hybrid metaheuristics*.

This chapter gives an overview of different ways to hybridize GRASP to create new and more effective metaheuristics. Several types of hybridizations are considered, involving different constructive procedures, enhanced local search algorithms, and memory structures.

In Section 2 the basic GRASP components are briefly reviewed. Hybrid construction schemes and hybridization with path-relinking are considered in Sections 3 and 4, respectively.

Hybridization schemes of GRASP with other metaheuristics are explained in Section 5. Concluding remarks are given in the last section.

2 A Basic GRASP

A basic GRASP metaheuristic [21, 22] is a multi-start or iterative method. Given a finite solution set X and a real-valued objective function $f : X \to R$ to be minimized, each GRASP iteration is usually made up of a construction phase, where a feasible solution is constructed, and a local search phase which starts at the constructed solution and applies iterative improvement until a locally optimal solution is found. Repeated applications of the construction procedure yields diverse starting solutions for the local search and the best overall solution is kept as the result.

The construction phase builds a solution x. If x is not feasible, a repair procedure is invoked to obtain feasibility. Once a feasible solution x is obtained, its neighborhood is investigated by the local search until a local minimum is found. The best overall solution is kept as the result. An extensive

```
procedure GRASP(f(·), g(·), MaxIterations, Seed)
1    x_best:=∅; f(x_best):=+∞;
2    for k = 1, 2, ...,MaxIterations→
3        x:=ConstructGreedyRandomizedSolution(Seed, g(·));
4        if (x not feasible) then
5            x:=repair(x);
6        endif
7        x:=LocalSearch(x, f(·));
8        if (f(x) < f(x_best)) then
9            x_best:=x;
10       endif
11   endfor;
12   return(x_best);
end GRASP
```

Fig. 1 Pseudo-code of a basic GRASP for a minimization problem

```
procedure ConstructGreedyRandomizedSolution(Seed, g(·))
1    x:=∅;
2    Sort the candidate elements i according to their incremental
     costs g(i);
3    while (x is not a complete solution)→
4        RCL:=MakeRCL();
5        v:=SelectIndex(RCL, Seed);
6        x := x ∪ {v};
7        Resort remaining candidate elements j according to their
         incremental costs g(j);
8    endwhile;
9    return(x);
end ConstructGreedyRandomizedSolution;
```

Fig. 2 Basic GRASP construction phase pseudo-code

survey of the literature is given in [26]. The pseudo-code in Figure 1 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 pseudorandom number generator.

Starting from an empty solution, in the construction phase, a complete solution is iteratively constructed, one element at a time (see Figure 2). The basic GRASP construction phase is similar to the semi-greedy heuristic proposed independently by [39]. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list C with respect to a greedy function $g : C \rightarrow R$. This function measures the (myopic) benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the *restricted candidate list* (RCL). In other words, the RCL is made up of elements $i \in C$ with the best (i.e., the smallest) incremental costs $g(i)$. There are two main mechanisms to build this list: a *cardinality-based* (CB) and a *value-based* (VB) mechanism. In the CB case, the RCL is made up of the k elements with the best incremental costs, where k is a parameter. In the VB case, the RCL is associated with a parameter $\alpha \in [0, 1]$ and a threshold value $\mu = g_{min} + \alpha(g_{max} - g_{min})$, where g_{min} and g_{max} are the smallest and the largest incremental costs, respectively, i.e.

$$g_{min} = \min_{i \in C} g(i), \qquad g_{max} = \max_{i \in C} g(i). \tag{1}$$

```
procedure LocalSearch(x, f(·))
1    Let N(x) be the neighborhood of x;
2    H:={y ∈ N(x) | f(y) < f(x)};
3    while (|H| > 0)→
4        x:=Select(H);
5        H:={y ∈ N(x) | f(y) < f(x)};
6    endwhile
7    return(x);
end LocalSearch
```

Fig. 3 Pseudo-code of a generic local search procedure

Then, all candidate elements i whose incremental cost $g(i)$ is no greater than the threshold value are inserted into the RCL, i.e. $g(i) \in [g_{min}, \mu]$. Note that, the case $\alpha = 0$ corresponds to a pure greedy algorithm, while $\alpha = 1$ is equivalent to a random construction.

Solutions generated by a GRASP construction are not guaranteed to be locally optimal with respect to simple neighborhood definitions. Hence, it is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm iteratively replaces the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The *neighborhood structure* N for a problem relates a solution s of the problem to a subset of solutions $N(s)$. A solution s is said to be *locally optimal* if in $N(s)$ there is no better solution in terms of objective function value. The key to success for a local search algorithm consists of the suitable choice of a neighborhood structure, efficient neighborhood search techniques, and the starting solution. Figure 3 illustrates the pseudo-code of a generic local search procedure for a minimization problem.

It is difficult to formally analyze the quality of solution values found by using the GRASP methodology. However, there is an intuitive justification that views GRASP as a repetitive sampling technique. Each GRASP iteration produces a sample solution from an unknown distribution of all obtainable results. The mean and variance of the distribution are functions of the restrictive nature of the candidate list, as experimentally shown by Resende and Ribeiro in [56].

An especially appealing characteristic of GRASP is the ease with which it can be implemented either sequentially or in parallel, where only a single global variable is required to store the best solution found over all processors. Moreover, few parameters need to be set and tuned, and therefore development can focus on implementing efficient data structures to assure quick GRASP iterations.

3 Hybrid Construction Mechanisms

In this Section, we briefly describe enhancements and alternative techniques for the construction phase of GRASP.

Reactive GRASP

Reactive GRASP is the first enhancement that incorporate a learning mechanism in the memoryless construction phase of the basic GRASP.

The value of the RCL parameter α is selected at each iteration from a discrete set of possible values with a probability that depends on the solution values found along the previous iterations. One way to accomplish this is to use the rule proposed in [53]. Let $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ be the set of possible values for α. At the first GRASP iteration, all m values have the same probability to be selected, i.e.

$$p_i = \frac{1}{m}, \qquad i = 1, 2, \ldots, m. \tag{2}$$

At any subsequent iteration, let $\hat{z}$ be 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 reevaluated as follows:

$$p_i = \frac{q_i}{\sum_{j=1}^{m} q_j}, \tag{3}$$

where $q_i = \frac{\hat{z}}{A_i}$, $i = 1, \ldots, m$.

Reactive GRASP has been successfully applied in solving several combinatorial optimization problems arising in real-world applications [11, 18].

Cost perturbations

Another step toward an improved and alternative solution construction mechanism is to allow *cost perturbations*. The idea to introduce some "noise" in the original costs in a fashion resembles the noising method of Charon and Hudry [15, 16] and can be usefully applied in all cases when the construction algorithm is not very sensitive to randomization or for the problem to be solved there is available no greedy algorithm for randomization.

Experimental results in the literature have shown that embedding a strategy of costs perturbation into a GRASP framework improves the best overall results. The hybrid GRASP with path-relinking proposed for the Steiner problem in graphs by Ribeiro et al. in [62] uses this cost perturbation strategy and is among the most effective heuristics currently available. Path-relinking will be in detail described in Section 4.

Bias functions

Another construction mechanism has been proposed by Bresina [12]. Once the RCL is built, instead of choosing with equal probability one candidate among

the RCL elements, Bresina introduced a family of probability distributions to bias the selection toward some particular candidates. A bias function is based on a rank $r(x)$ assigned to each candidate x according to its greedy function value and is evaluated only for the elements in RCL. Several different bias functions have been introduced:

i. random bias: $\text{bias}(r(x)) = 1$;
ii. linear bias: $\text{bias}(r(x)) = \frac{1}{r(x)}$;
iii. log bias: $\text{bias}(r(x)) = \log^{-1}[r(x) + 1]$;
iv. exponential bias: $\text{bias}(r(x)) = e^{-r}$;
v. polynomial bias of order n: $\text{bias}(r(x)) = r^{-n}$.

Let $\text{bias}(r(x))$ be one of the bias function defined above. Once these values have been evaluated for all elements of the RCL, the probability p_x of selecting element x is

$$p_x = \frac{\text{bias}(r(x))}{\sum_{y \in RCL} \text{bias}(r(y))}. \tag{4}$$

A successful application of Bresina's bias function can be found in [10], where experimental results show that the evaluation of bias functions may be restricted only to the elements of the RCL.

Other hybrid construction proposals

Resende and Werneck [57] proposed the following further construction methods:

i. *Sample greedy construction.*
Instead of randomizing the greedy algorithm, a greedy algorithm is applied to each solution in a random sample of candidates. At each step, a fixed-size subset of the candidates is sampled and the incremental contribution to the cost of the partial solution is computed for each sampled element. An element with the best incremental contribution is selected and added to the partial solution. This process is repeated until, as before, the construction terminates when no further candidate exists. Resende and Werneck in [57] proposed for the p-median problem a sample greedy construction scheme, whose general framework for a minimization problem is shown in Figure 4.
ii. *Random plus greedy construction.* A partial random solution is built and a greedy algorithm is then applied to complete the construction. The size k of the randomly built portion determines how greedy or random the construction will be. The pseudo-code is reported in Figure 5.
iii. *Proportional greedy construction.*
In each iteration of proportional greedy, the incremental cost $g(c)$ for every candidate element $c \in C$ is computed and then a candidate is picked at random, but in a biased way. In fact, the probability of a given candidate $v \in C$ being selected is inversely proportional to $g(v) - \min\{g(c) \mid c \in C\}$.

```
procedure ConstructSampleGreedySolution(Seed, g(·), k)
Let C be the set of candidate elements.
1   x:=∅;
2   while (x is not a complete solution)→
3     RCL:=select-randomly(Seed,k,C);   /*k candidates at random*/
4     Evaluate incremental costs of candidates in RCL;
5     v :=argmin{g(i) | i ∈ RCL};
6     x := x ∪ {v};
7     C := C \ {v};
8   endwhile;
9   return(x);
end ConstructSampleGreedySolution;
```

Fig. 4 Sample greedy GRASP construction phase pseudo-code

```
procedure ConstructRand+GreedySolution(Seed, g(·), k)
Let C be the set of candidate elements.
1   x:=select-randomly(Seed,k,C);   /*k candidates at random*/
2   C := C \ x;
3   while (x is not a complete solution)→
4     Evaluate incremental costs of candidates in C;
5     v :=argmin{g(i) | i ∈ C};
6     x := x ∪ {v};
7     C := C \ {v};
8   endwhile;
9   return(x);
end ConstructRand+GreedySolution;
```

Fig. 5 Random plus greedy GRASP construction phase pseudo-code

4 GRASP and Path-Relinking

Path-relinking is a heuristic proposed in 1996 by Glover [30] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [31, 33, 34]. It can be traced back to the pioneering work of Kernighan and Lin [42].

The result of the combination of the basic GRASP with path-relinking is a hybrid technique, leading to significant improvements in solution quality. The first proposal of a hybrid GRASP with path-relinking is in 1999 due to Laguna and Martí [46]. It was followed by several extensions, improvements, and successful applications [5, 13, 24, 25].

Starting from one or more elite solutions, paths in the solution space leading towards other guiding elite solutions are generated and explored in the search for better solutions. This is accomplished by selecting moves that

introduce attributes contained in the guiding solutions. At each iteration, all moves that incorporate attributes of the guiding solution are analyzed and the move that best improves (or least deteriorates) the initial solution is chosen.

Path-relinking is applied to a pair of solutions $\mathbf{x}, \mathbf{y}$, where one can be the solution obtained from the current GRASP iteration, and the other is a solution from an elite set of solutions. $\mathbf{x}$ is called the *initial solution* and $\mathbf{y}$ the *guiding solution*. The set $\mathcal{E}$ of elite solutions has usually a fixed size that does not exceed `MaxElite`. Given the pair $\mathbf{x}, \mathbf{y}$, their common elements are kept constant, and the space of solutions spanned by these elements is searched with the objective of finding a better solution. The size of the solution space grows exponentially with the the distance between the *initial* and *guiding* solutions and therefore only a small part of the space is explored by path-relinking. The procedure starts by computing the symmetric difference $\Delta(\mathbf{x}, \mathbf{y})$ between the two solutions, i.e. the set of moves needed to reach $\mathbf{y}$ (target solution) from $\mathbf{x}$ (initial solution). A path of solutions is generated linking $\mathbf{x}$ and $\mathbf{y}$. The best solution x^* in this path is returned by the algorithm.

Let us denote the set of solutions spanned by the common elements of the n-vectors $\mathbf{x}$ and $\mathbf{y}$ as

$$S(\mathbf{x}, \mathbf{y}) := \{w \text{ feasible} \mid w_i = x_i = y_i, i \notin \Delta(\mathbf{x}, \mathbf{y})\} \setminus \{\mathbf{x}, \mathbf{y}\}. \tag{5}$$

Clearly, $|S(\mathbf{x}, \mathbf{y})| = 2^{n-d(\mathbf{x},\mathbf{y})} - 2$, where $d(\mathbf{x}, \mathbf{y}) = |\Delta(\mathbf{x}, \mathbf{y})|$. The underlying assumption of path-relinking is that there exist good-quality solutions in $S(\mathbf{x}, \mathbf{y})$, since this space consists of all solutions which contain the common elements of two good solutions $\mathbf{x}$ and $\mathbf{y}$. Since the size of this space is exponentially large, a greedy search is usually performed where a path of solutions

$$\mathbf{x} = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{d(\mathbf{x},\mathbf{y})}, \mathbf{x}_{d(\mathbf{x},\mathbf{y})+1} = \mathbf{y}, \tag{6}$$

is built, such that $d(\mathbf{x}_i, \mathbf{x}_{i+1}) = 1, \quad i = 0, \ldots, d(\mathbf{x}, \mathbf{y})$, and the best solution from this path is chosen. Note that, since both $\mathbf{x}$ and $\mathbf{y}$ are, by construction, local optima in some neighborhood $N(\cdot)$[1], in order for $S(\mathbf{x}, \mathbf{y})$ to contain solutions which are not contained in the neighborhoods of $\mathbf{x}$ or $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{y}$ must be sufficiently distant.

Figure 6 illustrates the pseudo-code of the path-relinking procedure applied to the pair of solutions $\mathbf{x}$ (starting solution) and $\mathbf{y}$ (target solution). In line 1, an initial solution $\mathbf{x}$ is selected at random among the elite set elements and it usually differs sufficiently from the guiding solution $\mathbf{y}$. The loop in lines 6 through 14 computes a path of solutions $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{d(\mathbf{x},\mathbf{y})-2}$, and the solution x^* with the best objective function value is returned in line 15. This is achieved by advancing one solution at a time in a greedy manner. At each iteration, the procedure examines all moves $m \in \Delta(x, \mathbf{y})$ from the current

[1] The same metric $d(\mathbf{x}, \mathbf{y})$ is usually used.

```
procedure Path-relinking(f(·), x, E)
1     Choose, at random, a pool solution y ∈ E to relink with x;
2     Compute symmetric difference Δ(x, y);
3     f* := min{f(x), f(y)};
4     x* := arg min{f(x), f(y)};
5     x := x;
6     while (Δ(x, y) ≠ ∅) →
7           m* := arg min{f(x ⊕ m) | m ∈ Δ(x, y)};
8           Δ(x ⊕ m*, y) := Δ(x, y) \ {m*};
9           x := x ⊕ m*;
10          if (f(x) < f*) then
11                f* := f(x);
12                x* := x;
13          endif;
14    endwhile;
15    x* := LocalSearch(x*, f(·));
16    return (x*);
end Path-relinking;
```

Fig. 6 Pseudo-code of a generic path-relinking for a minimization problem

solution x and selects the one which results in the least cost solution (line 7), i.e. the one which minimizes $f(x \oplus m)$, where $x \oplus m$ is the solution resulting from applying move m to solution x. The best move m^* is made, producing solution $x \oplus m^*$ (line 9). The set of available moves is updated (line 8). If necessary, the best solution x^* is updated (lines 10–13). $\Delta(x, \mathbf{y}) = \emptyset$. Since x^* is not guaranteed to be locally optimal, a local search is usually applied and the locally optimal solution is returned.

We now describe a possible way to hybridize the basic GRASP described in Section 2 with path-relinking. The integration of the path-relinking procedure with the basic GRASP is shown in Figure 7. The pool $\mathcal{E}$ of elite solutions is initially empty, and until it reaches its maximum size no path relinking takes place. After a solution $\mathbf{x}$ is found by GRASP, it is passed to the path-relinking procedure to generate another solution. The procedure `AddToElite`$(\mathcal{E}, x_p)$ attempts to add to the elite set of solutions the currently found solution. Since we wish to maintain a pool of good but diverse solutions, each solution obtained by path-relinking 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 replace the latter. Another strategy, which tends to increase the diversity of the pool, is to replace the pool element most similar to the candidate among all pool elements with cost worse than the candidate's.

More formally, in several papers, a solution x_p is added to the elite set $\mathcal{E}$ if either one of the following conditions holds:

```
procedure GRASP+PR(f(·), g(·), MaxIterations, Seed, MaxElite)
1    x_best:=∅; f(x_best):=+∞; E := ∅
2    for k = 1, 2, ...,MaxIterations→
3        x:=ConstructGreedyRandomizedSolution(Seed, g(·));
4        if (x not feasible) then
5            x:=repair(x);
6        endif
7        x:=LocalSearch(x, f(·));
8        if (k ≤MaxElite) then
9            E := E ∪ {x};
10           if (f(x) < f(x_best)) then
11               x_best:=x;
12           endif
13       else
14           x_p:=Path-relinking(f(·), x, E);
15           AddToElite(E, x_p);
16           if (f(x_p) < f(x_best)) then
17               x_best:=x_p;
18           endif
19       endif
20   endfor;
21   return(x_best);
end GRASP+PR
```

Fig. 7 Pseudo-code of a basic GRASP with path-relinking heuristic for a minimization problem

1. $f(x_p) < \min\{f(\mathbf{w}) : \mathbf{w} \in \mathcal{E}\}$,
2. $\min\{f(\mathbf{w}) : \mathbf{w} \in \mathcal{E}\} \leq f(x_p) < \max\{f(\mathbf{w}) : \mathbf{w} \in \mathcal{E}\}$ and $d(x_p, \mathbf{w}) > \beta n, \quad \forall \mathbf{w} \in \mathcal{E}$, where β is a parameter between 0 and 1 and n is the number of decision variables.

If x_p satisfies either of the above, it then replaces an elite solution $\mathbf{z}$ no better than x_p and most similar to x_p, i.e. $\mathbf{z} = \text{argmin}\{d(x_p, \mathbf{w}) : \mathbf{w} \in \mathcal{E}$ such that $f(\mathbf{w}) \geq f(x_p)\}$.

Figure 7 shows the simplest way to combine GRASP with path-relinking, which is applied as an intensification strategy to each local optimum obtained after the GRASP local search phase.

In general, two basic strategies can be used:

i. path-relinking is applied as a post-optimization step to all pairs of elite solutions;
ii. path-relinking is applied as an intensification strategy to each local optimum obtained after the local search phase.

Applying path-relinking as an intensification strategy to each local optimum (strategy ii.) seems to be more effective than simply using it as a post-optimization step [58].

Several further alternatives have been recently considered and combined, all involving the trade-offs between computation time and solution quality. They include:

a. do not apply path-relinking at every GRASP iteration, but only periodically;
b. explore only one path, starting from either $\mathbf{x}$ (*forward path-relinking*) or $\mathbf{y}$ (*backward path-relinking*);
c. explore two different paths, using first $\mathbf{x}$, then $\mathbf{y}$ as the initial solution (*forward and backward path-relinking*);
d. do not follow the full path, but instead only part of it (*truncated path-relinking*).

Ribeiro et al. [61] observed that exploring two different paths for each pair $(\mathbf{x}, \mathbf{y})$ 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 path is to be investigated, better solutions are found when path-relinking starts from the best among $\mathbf{x}$ and $\mathbf{y}$. 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 path before the latter is reached.

Resende and Ribeiro [55] performed extensive computational experiments, running implementations of GRASP with several different variants of path-relinking. They analyzed the results and illustrated the trade-offs between the different strategies.

5 GRASP and Other Metaheuristics

In this section, we describe and comment on some enhancements of the basic GRASP obtained by hybridization with other approaches and optimization strategies. We also report on experience showing that a crafted combination of concepts of different metaheuristics/techniques can result in robust combinatorial optimization schemes and produce higher solution quality than the individual metaheuristics themselves, especially when solving difficult real-world combinatorial optimization problems.

Most of the GRASP hybrid approaches involve other metaheuristics in the basic local search scheme described in Section 2. They include methods that explore beyond the current solution's neighborhood by allowing cost-increasing moves, by exploring multiple neighborhoods, and by exploring very large neighborhoods.

5.1 GRASP and Tabu Search

Tabu search (TS) is a metaheuristic strategy introduced by Glover [28, 29, 30, 32, 33] that makes use of memory structures to enable escape from local minima by allowing cost-increasing moves. During the search, short-term memory TS uses a special data structure called *tabu list* to store information about solutions generated in the last iterations[2]. The process starts from a given solution and, as any local search heuristic, it moves in iterations from the current solution s to some solution $t \in N(s)$. To avoid returning to a just-visited local minimum, reverse moves mov_t that lead back to that local minimum are forbidden, or made *tabu*, for a number of iterations that can be a priori fixed (fixed sized tabu list) or adaptively varying (variable sized tabu list).

```
procedure TS(x, f(·), k)
1    Let N(x) be the neighborhood of x;
2    s := x; T := ∅; x_b := x;
3    while (stopping criterion not satisfied)→
4        N̂(s) := N(s) \ T;
5        t :=argmin{f(w) | w ∈ N̂(s)};
6        if (|T| ≥ k) then
7            Remove from T the oldest entry;
8        endif
9        T := T ∪ {t};
10       if (f(t) < f(x_b)) then
11           x_b := t;
12       endif
13       s := t;
14   endwhile
15   return(x_b);
end TS
```

Fig. 8 Short memory TS pseudo-code for a minimization problem

Figure 8 shows pseudo-code for a short-term TS using a fixed k sized tabu list T, that, for ease of handling, stores the complete solutions t instead of the corresponding moves mov_t.

It is clear that TS can be used as a substitute for the standard local search in a GRASP. This type of search allows the exploration beyond the neighborhood of the greedy randomized solution. By using the number of cost-increasing moves as a stopping criterion one can balance the amount

[2] Usually, the tabu list stores all moves that reverse the effect of recent local search steps.

```
procedure simulated-annealing (x, f(·), T, Seed)
1    s := x; x_b := x;
2    while (T > 0 and stopping criterion not satisfied)→
3        t :=select-randomly(Seed, N(s));
4        if (f(t) − f(s) < 0) then
5            s := t;
6            if (f(t) < f(x_b)) then x_b := t;
7            endif
8        else s := t with probability e^{−(f(t)−f(s))/(K·T)};
9        endif
10       Decrement T according to a defined criterion;
11   endwhile
12   return (x_b);
end simulated-annealing
```

Fig. 9 SA pseudo-code for a minimization problem

of time that GRASP allocates to constructing a greedy randomized solution and exploring around that solution with tabu search.

Examples of GRASP with tabu search include [18] for the single source capacitated plant location problem, [1] for multi-floor facility layout, [71] for the capacitated minimum spanning tree problem, [48] for the m-VRP with time windows, and [20] for the maximum diversity problem.

5.2 GRASP and Simulated Annealing

Simulated annealing (SA) [43] is based on principles of mechanical statistics and on the idea of simulating the annealing process of a mechanical system.

It offers a further possibility to enhance the basic GRASP local search phase and pseudo-code in Figure 9 shows how SA can be used as a substitute for the standard local search in a GRASP.

As any stochastic local search procedure, SA is also given a starting solution x which is used to initialize the current solution s. At each iteration, it randomly selects a trial solution $t \in N(s)$. In perfect correspondence of mechanical systems state change rules, if t is an improving solution, then t is made the current solution. Otherwise, t is made the current solution with probability given by

$$e^{-\frac{f(t)-f(s)}{K \cdot T}}, \tag{7}$$

where $f(x)$ is interpreted as the energy of the system in state x, K is the Boltzmann constant, and T a control parameter called the *temperature*.

There are many ways to implement SA, depending on the adopted stopping criterion and on the rule (*cooling schedule*) applied to decrement the temperature parameter T (line 10). Note that, the higher is the temperature T the higher is the probability of moving on a not improving solution t.

Usually, starting from a high initial temperature T_0, at iteration k the cooling schedule changes the temperature by setting $T_{k+1} := T_k \cdot \gamma$, where $0 < \gamma < 1$.

Therefore, initial iterations can be thought of as a diversification phase, where a large part of the solution space is explored. As the temperature cools, fewer non-improving solutions are accepted and those cycles can be thought of as intensification cycles.

To make use of SA as a substitute for the standard local search in GRASP, one should limit the search to the intensification part, since the diversification is already guaranteed by the randomness of the GRASP construction phase. Limitation to only intensification part can be done by starting already with a cool temperature T_0.

Examples of hybrid GRASP with SA include [70] for a simplified fleet assignment problem and [17] for the rural postman problem.

5.3 GRASP, Genetic Algorithms, and Population-Based Heuristics

Evolutionary metaheuristics such as genetic algorithms (GA) [36], ant colony optimization [19], scatter search [35, 45, 47], and evolutionary path-relinking [57] require the generation of an initial population of solutions.

Rooted in the mechanisms of evolution and natural genetics and therefore derived from the principles of natural selection and Darwin's evolutionary theory, the study of heuristic search algorithms with underpinnings in natural and physical processes began as early as the 1970s, when Holland [40] first proposed genetic algorithms. This type of evolutionary technique has been theoretically and empirically proven to be a robust search method [36] having a high probability of locating the global solution optimally in a multimodal search landscape.

In nature, competition among individuals results in the fittest individuals surviving and reproducing. This is a natural phenomenon called *the survival of the fittest*: the genes of the fittest survive, while the genes of weaker individuals die out. The reproduction process generates diversity in the gene pool. Evolution is initiated when the genetic material (chromosomes) from two parents recombines during reproduction. The exchange of genetic material among chromosomes is called *crossover* and can generate good combination of genes for better individuals. Another natural phenomenon called *mutation* causes regenerating lost genetic material. Repeated selection, mutation, and crossover cause the continuous evolution of the gene pool and the generation of individuals that survive better in a competitive environment.

In complete analogy with nature, once encoded each possible point in the search space of the problem into a suitable representation, a GA transforms a population of individual solutions, each with an associated *fitness* (or objective function value), into a new generation of the population. By applying genetic operators, such as crossover and mutation [44], a GA

```
procedure GA(f(·))
1    Let N(x) be the neighborhood of a solution x;
2    k := 0;
3    Initialize population P(0); x_b :=argmin{f(x) | x ∈ P(0)};
4    while (stopping criterion not satisfied)→
5        k := k + 1;
6        Select P(k) from P(k − 1);
7        t :=argmin{f(x) | x ∈ P(k)};
8        if (f(t) < f(x_b)) then
9          x_b := t;
10       endif
11       Alter P(k);
12   endwhile
13   return(x_b);
end GA
```

Fig. 10 Pseudo-code of a generic GA for a minimization problem

successively produces better approximations to the solution. At each iteration, a new generation of approximations is created by the process of selection and reproduction. In Figure 10 a simple genetic algorithm is described by the pseudo-code, where $P(k)$ is the population at iteration k.

In solving a given optimization problem $\mathcal{P}$, a GA consists of the following basic steps.

1. Randomly create an initial population $P(0)$ of individuals, i.e. solutions for $\mathcal{P}$.
2. Iteratively perform the following substeps on the current generation of the population until the termination criterion has been satisfied.
   a. Assign fitness value to each individual using the fitness function.
   b. Select parents to mate.
   c. Create children from selected parents by crossover and mutation.
   d. Identify the best-so-far individual for this iteration of the GA.

Scatter Search (SS) operates on a *reference set* of solutions, that are combined to create new ones. One way to obtain a new solution is to linearly combine two reference set solutions. Unlike a GA, the reference set of solutions is relatively small, usually consisting of less than 20 solutions. At the beginning, a starting set of solutions is generated to guarantee a critical level of diversity and some local search procedure is applied to attempt to improve them. Then, a subset of the best solutions is selected as reference set, where the quality of a solution is evaluated both in terms of objective function and diversity with other reference set candidates. At each iteration, new solutions are generated by combining reference set solutions. One criterion used to

select reference solutions for combination takes into account the convex regions spanned by the reference solutions.

Evolutionary path-relinking (EvPR) has been introduced by Resende and Werneck [57] and applied as a post-processing phase for GRASP with PR. In EvPR, the solutions in the pool are evolved as a series of populations $P(1), P(2), \ldots$ of equal size. The initial population $P(0)$ is the pool of elite solutions produced by GRASP with PR. In iteration k, PR is applied between a set of pairs of solutions in population $P(k)$ and, with the same rules used to test for membership in the pool of elite solutions, each resulting solution is tested for membership in population $P(k+1)$. This evolutionary process is repeated until no improvement is seen from one population to the next.

As just described, all above techniques are evolutionary metaheuristics requiring the generation of an initial population of solutions. Usually, these initial solutions are randomly generated, but another way to generate them is to use a GRASP.

Ahuja et al. [4] used a GRASP to generate the initial population of a GA for the quadratic assignment problem. Alvarez et al. [6] proposed a GRASP embedded scatter search for the multicommodity capacitated network design problem. Very recently, Contreras and Díaz used GRASP to initialize the reference set of scatter search for the single source capacitated facility location problem. GRASP with EvPR has been recently used in [59] for the uncapacitated facility location problem and in [54] for the max-min diversity problem.

5.4 GRASP and Variable Neighborhood Search

Almost all randomization effort in implementations of the basic GRASP involves the construction phase. On the other hand, strategies such as Variable Neighborhood Search (VNS) and Variable Neighborhood Descent (VND) [38, 51] rely almost entirely on the randomization of the local search to escape from local optima. With respect to this issue, probabilistic strategies such as GRASP and VNS may be considered as complementary and potentially capable of leading to effective hybrid methods.

The variable neighborhood search (VNS) metaheuristic, proposed by Hansen and Mladenović [38], is based on the exploration of a dynamic neighborhood model. Contrary to other metaheuristics based on local search methods, VNS allows changes of the neighborhood structure along the search.

VNS explores increasingly distant neighborhoods of the current best found solution. Each step has three major phases: neighbor generation, local search, and jump. Let N_k, $k = 1, \ldots, k_{max}$ be a set of pre-defined neighborhood structures and let $N_k(x)$ be the set of solutions in the kth-order neighborhood of a solution x. In the first phase, a neighbor $x' \in N_k(x)$ of the current solution is applied. Next, a solution x'' is obtained by applying local search to x'. Finally, the current solution jumps from x to x'' in case the latter

```
procedure VNS(x, f(·), k_max, Seed)
1    x_b := x; k := 1;
2    while (k ≤ k_max)→
3        x' :=select-randomly(Seed, N_k(x));
4        x'' :=LocalSearch(x', f(·));
5        if (f(x'') < f(x')) then
6            x := x''; k := 1;
7            if (f(x'') < f(x_b)) then x_b := x'';
8            endif
9        else k := k + 1;
10       endif
11   endwhile
12   return(x_b);
end VNS
```

Fig. 11 Pseudo-code of a generic VNS for a minimization problem

improved the former. Otherwise, the order of the neighborhood is increased by one and the above steps are repeated until some stopping condition is satisfied.

Usually, until a stopping criterion is met, VNS generates at each iteration a solution x at random. In hybrid GRASP with VNS, where VNS is applied as local search, the starting solution is the output x of the GRASP construction procedure and the pseudo-code of a generic VNS local search is illustrated in Figure 11.

Examples of GRASP with VNS include [14] for the prize-collecting Steiner tree problem in graphs, [25] for the MAX-CUT problem, and [9] for the strip packing problem.

VND allows the systematic exploration of multiple neighborhoods and is based on the facts that a local minimum with respect to one neighborhood is not necessarily a local minimum with respect to another and that a global minimum is a local minimum with respect to all neighborhoods. VND also is based on the empirical observation that, for many problems, local minima with respect to one or more neighborhoods are relatively close to each other. Since a global minimum is a local minimum with respect to all neighborhoods, it should be easier to find a global minimum if more neighborhoods are explored.

Let $N_k(x)$, for $k = 1, \ldots, k_{max}$, be k_{max} neighborhood structures of solution x. The search begins with a given starting solution x which is made the current solution s. Each major iteration (lines 2–11) searches for an improving solution t in up to k_{max} neighborhoods of s. If no improving solution is found in any of the neighborhoods, the search ends. Otherwise, t is made the current solution s and the search is applied starting from s.

```
procedure VND(x, f(·), k_max)
1    x_b := x; s := x; flag:=true;
2    while (flag)→
3       flag:=false;
4       for k = 1, ..., k_max →
5          if (∃t ∈ N_k(s) | f(t) < f(s)) then
6             if (f(t) < f(x_b)) then x_b := t;
7             endif
8             s := t; flag:=true; break;
9          endif
10      endfor
11   endwhile
12   return(x_b);
end VND
```

Fig. 12 Pseudo-code of a generic VND for a minimization problem

In hybrid GRASP with VND, where VND is applied as local search, the starting solution is the output x of the GRASP construction procedure and the pseudo-code of a generic VND local search is illustrated in Figure 12. A first attempt in the direction of hybridizing GRASP with VNS has been done by Martins et al. [50]. The construction phase of their hybrid heuristic for the Steiner problem in graphs follows the greedy randomized strategy of GRASP, while the local search phase makes use of two different neighborhood structures as a VND strategy. Their heuristic was later improved by Ribeiro, Uchoa, and Werneck [61], one of the key components of the new algorithm being another strategy for the exploration of different neighborhoods. Ribeiro and Souza [60] also combined GRASP with VND in a hybrid heuristic for the degree-constrained minimum spanning tree problem. In the more recent literature, Ribeiro and Vianna [64] and Andrade and Resende [7] proposed a hybrid GRASP with VND for the phylogeny problem and for PBX telephone migration scheduling problem, respectively.

5.5 GRASP and Iterated Local Search

Iterated Local Search (ILS) [49] is a multistart heuristic that at each iteration k finds a locally optimal solution searched in the neighborhood of an initial solution obtained by perturbation of the local optimum found by local search at previous iteration $k-1$.

The efficiency of ILS strongly depends on the perturbation (line 3) and acceptance criterion (line 5) rules. A "light" perturbation may cause local search to lead back to the starting solution t, while a "strong" perturbation may cause the search to resemble random multi-start. Usually, the acceptance criterion

```
procedure ils (x, f(·), history)
1    t :=LocalSearch(x, f(·)); x_b := t;
2    while (stopping criterion not satisfied)→
3        s :=perturbation(t, history);
4        ŝ :=LocalSearch(s, f(·));
5        t :=AcceptanceCriterion(t, ŝ, history);
6        if (f(t) < f(x_b)) then x_b := t;
7        endif
8    endwhile
9    return (x_b);
end ils
```

Fig. 13 ILS pseudo-code for a minimization problem

resembles SA, i.e. $\hat{s}$ is accepted if it is an improving solution; otherwise, it is accepted with some positive probability.

ILS can be applied to enhance the basic GRASP local search phase and pseudo-code in Figure 13 shows how it can be used as a substitute for the standard local search in a GRASP. The procedure `LocalSearch` can also be the basic GRASP local search as defined in Figure 3.

Ribeiro and Urrutia [63] designed a hybrid GRASP with ILS for the mirrored traveling tournament problem, where the acceptance rule makes use of a threshold parameter β, initialized to 0.001. Then, each time the best solution changes (line 6), it is reinitialized to the same value, while it is doubled if the current solution does not chance after a fixed number of iterations. Finally, a solution $\hat{s}$ is accepted if $f(\hat{s}) < (1+\beta) \cdot f(t)$ and the adopted stopping criterion has been to allow at most 50 cost-deteriorating moves without improvement in the current best solution.

5.6 GRASP and Very-Large Scale Neighborhood Search

As for any local search procedure, to efficiently search in the neighborhood of a solution, it is required that the neighborhood have a small size. Nevertheless, the larger the neighborhood, the better the quality of the locally optimal solution. Neighborhoods whose sizes grow exponentially as a function of problem dimension are called *very large scale neighborhoods* and they necessarily require efficient search techniques to be explored.

Ahuja et al. [2] presented a survey of methods called very-large scale neighborhood (VLSN) search. The following three classes of VLSN methods are described:

1. variable-depth methods where exponentially large neighborhoods are searched with heuristics;

2. a VLSN method that uses network flow techniques to identify improving neighborhood solutions;
3. a VLSN method that explores neighborhoods for NP-hard problems induced by restrictions of the problems that are solved in polynomial time.

In particular, with respect to class 2, they define special neighborhood structures called multi-exchange neighborhoods. The search is based on the cyclic transfer neighborhood structure that transforms a cost-reducing exchange into a negative cost subset-disjoint cycle in an *improving graph* and then a modified shortest path label-correcting algorithm is used to identify these negative cycles.

Ahuja et al. in [3] present two generalizations of the best known neighborhood structures for the capacitated minimum spanning tree problem. The new neighborhood structures defined allow cyclic exchanges of nodes among multiple subtrees simultaneously. To judge the efficacy of the neighborhoods, local improvement and tabu search algorithms have been developed. Local improvement uses a GRASP construction mechanism to generate repeated starting solutions for local improvement.

5.7 Other Hybridizations

In the previous sections of this chapter, we have reviewed some important hybridizations of GRASP, mostly involving the GRASP local search phase. More recently, several further hybridizations have been proposed. They include the use of GRASP in Branch & Bound framework and the combination of GRASP with data mining techniques.

GRASP and branch & bound

In 2004, Rocha et al. [66] proposed a hybridization of GRASP as an upper bound for a branch and bound (B&B) procedure to solve a scheduling problem with non-related parallel machines and sequence-dependent setup times. In 2007, Fernandes and Lourenço [23] proposed a hybrid GRASP and B&B for the job-shop scheduling problem. The B&B method is used within GRASP to solve subproblems of one machine scheduling subproblem obtained from the incumbent solution.

GRASP and data mining

In 2006, Jourdan et al. [41] presented a short survey enumerating opportunities to combine metaheuristics and data mining (DM) techniques. By using methods and theoretical results from statistics, machine learning, and pattern recognition, DM automatically explores large volumes of data (instances described according to several attributes) with the objective of discovering patterns. In fact, DM is also known as Knowledge-Discovery in Databases.

In GRASP with data mining (DM-GRASP), after executing a significant number of GRASP iterations, the data mining process extracts patterns from an elite set of solutions which will guide the GRASP construction procedure in the subsequent iterations. In fact, instead of building the randomized greedy solution from scratch, the construction procedure starts from a solution pattern (a partial solution) that was previously mined. Computational experiments have shown that the hybridization has benefited in both running time and quality of the solutions found.

DM-GRASP has been introduced in 2005 by Santos et al [69] for the maximum diversity problem. In 2006, Ribeiro et al. [65] also proposed a hybrid GRASP with DM and tested it the set packing problem as a case study and Santos et al. [68] solved a real world problem, called server replication for reliable multicast.

Very recently, s survey of applications of DM-GRASP has been published by Santos et al. [67].

6 Concluding Remarks

Simulated annealing, tabu search, ant colony, genetic algorithms, scatter search, path-relinking, GRASP, iterated local search, and variable neighborhood search are often listed as examples of "classical" metaheuristics. In the last few years, several different algorithms have been designed and proposed in the literature that do not purely apply the basic ideas of one single "classical" metaheuristic, but they combine various algorithmic ideas of different metaheuristic frameworks. The design and implementation of hybrid metaheuristics are emerging as one of the most exciting field.

In this chapter, we have surveyed hybridizations of GRASP and other metaheuristics. Among these, we highlight: path-relinking, tabu search, simulated annealing, genetic algorithms and population-based heuristics, variable neighborhood search and variable neighborhood descent, iterated local search, very large scale neighborhood local search, and very recent hybrids, such as GRASP with data mining and GRASP with branch and bound.

References

1. Abdinnour-Helm, S., Hadley, S.W.: Tabu search based heuristics for multi-floor facility layout. International J. of Production Research 38, 365–383 (2000)
2. Ahuja, R.K., Ergun, O., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Appl. Math. 123, 75–102 (2002)
3. Ahuja, R.K., Orlin, J.B., Sharma, D.: Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. Mathematical Programming 91, 71–97 (2001)
4. Ahuja, R.K., Orlin, J.B., Tiwari, A.: A greedy genetic algorithm for the quadratic assignment problem. Computers and Operations Research 27, 917–934 (2000)

5. Aiex, R., Resende, M.G.C., Pardalos, P.M., Toraldo, G.: GRASP with path relinking for three-index assignment. INFORMS J. on Computing 17(2), 224–247 (2005)
6. Alvarez, A.M., Gonzalez-Velarde, J.L., De Alba, K.: GRASP embedded scatter search for the multicommodity capacitated network design problem. J. of Heuristics 11, 233–257 (2005)
7. Andrade, D.V., Resende, M.G.C.: A GRASP for PBX telephone migration scheduling. In: Eighth INFORMS Telecommunication Conference (April 2006)
8. Baum, E.B.: Iterated descent: A better algorithm for local search in combinatorial optimization problems. Technical report, California Institute of Technology (1986)
9. Beltrán, J.D., Calderón, J.E., Cabrera, R.J., Pérez, J.A.M., Moreno-Vega, J.M.: GRASP/VNS hybrid for the strip packing problem. In: Proceedings of Hybrid Metaheuristics (HM 2004), pp. 79–90 (2004)
10. Binato, S., Hery, W.J., Loewenstern, D., Resende, M.G.C.: A greedy randomized adaptive search procedure for job shop scheduling. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and surveys on metaheuristics, pp. 58–79. Kluwer Academic Publishers, Dordrecht (2002)
11. Binato, S., Oliveira, G.C.: A Reactive GRASP for transmission network expansion planning. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and surveys on metaheuristics, pp. 81–100. Kluwer Academic Publishers, Dordrecht (2002)
12. Bresina, J.L.: Heuristic-biased stochastic sampling. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI 1996), pp. 271–278 (1996)
13. 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)
14. 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)
15. Charon, I., Hudry, O.: The noising method: A new method for combinatorial optimization. Operations Research Letters 14, 133–137 (1993)
16. Charon, I., Hudry, O.: The noising methods: A survey. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and surveys on metaheuristics, pp. 245–261. Kluwer Academic Publishers, Dordrecht (2002)
17. 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)
18. 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)
19. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
20. Duarte, A., Martí, R.: Tabu search and GRASP for the maximum diversity problem. European J. of Operational Research 178(1), 71–84 (2007)
21. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8, 67–71 (1989)
22. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. of Global Optimization 6, 109–133 (1995)

23. Fernandes, S., Lourenço, H.R.: A GRASP and Branch-and-Bound Metaheuristic for the Job-Shop Scheduling. In: Cotta, C., van Hemert, J. (eds.) EvoCOP 2007. LNCS, vol. 4446, pp. 60–71. Springer, Heidelberg (2007)
24. 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)
25. 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)
26. Festa, P., Resende, M.G.C.: GRASP: An annotated bibliography. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and Surveys on Metaheuristics, pp. 325–367. Kluwer Academic Publishers, Dordrecht (2002)
27. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman and Company, New York (1979)
28. Glover, F.: Tabu search – Part I. ORSA J. on Computing 1, 190–206 (1989)
29. Glover, F.: Tabu search – Part II. ORSA J. on Computing 2, 4–32 (1990)
30. Glover, F.: Tabu search and adaptive memory programing – 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, Dordrecht (1996)
31. Glover, F.: Multi-start and strategic oscillation methods – Principles to exploit adaptive memory. In: Laguna, M., Gonzáles-Velarde, J.L. (eds.) Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, pp. 1–24. Kluwer, Dordrecht (2000)
32. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1997)
33. Glover, F., Laguna, M.: Tabu search. Kluwer Academic Publishers, Dordrecht (1997)
34. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics 39, 653–684 (2000)
35. Glover, F., Laguna, M., Martí, R.: Scatter Search. In: Ghosh, A., Tsutsui, S. (eds.) Advances in Evolutionary Computation: Theory and Applications, pp. 519–537. Kluwer Academic Publishers, Dordrecht (2003)
36. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading (1989)
37. Hansen, P., Mladenović, N.: An introduction to variable neighborhood search. In: Voss, S., Martello, S., Osman, I.H., Roucairol, C. (eds.) Meta-heuristics, Advances and trends in local search paradigms for optimization, pp. 433–458. Kluwer Academic Publishers, Dordrecht (1998)
38. Hansen, P., Mladenović, N.: Developments of variable neighborhood search. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and Surveys in Metaheuristics, pp. 415–439. Kluwer Academic Publishers, Dordrecht (2002)
39. Hart, J.P., Shogan, A.W.: Semi-greedy heuristics: An empirical study. Operations Research Letters 6, 107–114 (1987)
40. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. of Michigan Press, Ann Arbor (1975)
41. Jourdan, L., Dhaenens, C., Talbi, E.-G.: Using datamining techniques to help metaheuristics: A short survey. In: Almeida, F., Blesa Aguilera, M.J., Blum, C., Moreno Vega, J.M., Pérez Pérez, M., Roli, A., Sampels, M. (eds.) HM 2006. LNCS, vol. 4030, pp. 57–69. Springer, Heidelberg (2006)

42. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning problems. Bell System Technical Journal 49(2), 291–307 (1970)
43. Kirkpatrick, S.: Optimization by simulated annealing: Quantitative studies. J. of Statistical Physics 34, 975–986 (1984)
44. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A.: Genetic Programming III, Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, San Francisco (1999)
45. Laguna, M.: Scatter Search. In: Pardalos, P.M., Resende, M.G.C. (eds.) Handbook of Applied Optimization, pp. 183–193. Oxford University Press, Oxford (2002)
46. Laguna, M., Martí, R.: GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS J. on Computing 11, 44–52 (1999)
47. Laguna, M., Martí, R.: Scatter Search: Methodology and Implementations in C. Kluwer, Dordrecht (2003)
48. Lim, A., Wang, F.: A smoothed dynamic tabu search embedded GRASP for m-VRPTW. In: Proceedings of ICTAI 2004, pp. 704–708 (2004)
49. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 321–353. Kluwer Academic Publishers, Dordrecht (2003)
50. Martins, S.L., Resende, M.G.C., Ribeiro, C.C., Pardalos, P.: A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. Journal of Global Optimization 17, 267–283 (2000)
51. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research 24, 1097–1100 (1997)
52. Osman, I.H., Laporte, G.: Metaheuristics: A bibliography. Annals of Operations Research 63, 513 (1996)
53. 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)
54. Resende, M.G.C., Martí, R., Gallego, M., Duarte, A.: GRASP and path relinking for the max-min diversity problem. Technical report, AT&T Labs Research, Florham Park, NJ–USA (2008)
55. Resende, M.G.C., Ribeiro, C.C.: A GRASP with path-relinking for private virtual circuit routing. Networks 41, 104–114 (2003)
56. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 219–249. Kluwer Academic Publishers, Dordrecht (2003)
57. Resende, M.G.C., Ribeiro, C.C.: A hybrid heuristic for the p-median problem. J. of Heuristics 10, 59–88 (2004)
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, Heidelberg (2005)
59. 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)
60. Ribeiro, C.C., Souza, M.C.: Variable neighborhood search for the degree constrained minimum spanning tree problem. Discrete Applied Mathematics 118, 43–54 (2002)
61. Ribeiro, C.C., Uchoa, E., Werneck, R.F.: A hybrid GRASP with perturbations for the Steiner problem in graphs. INFORMS Journal on Computing 14, 228–246 (2002)

62. 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)
63. Ribeiro, C.C., Urrutia, S.: Heuristics for the mirrored traveling tournament problem. European J. of Operational Research 179, 775–787 (2007)
64. Ribeiro, C.C., Vianna, D.S.: A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. Intl. Trans. in Op. Res. 12, 325–338 (2005)
65. Ribeiro, M.H., Plastino, A., Martins, S.L.: Hybridization of GRASP metaheuristic with data mining techniques. J. of Mathematical Modelling and Algorithms 5, 23–41 (2006)
66. Rocha, P.L., Ravetti, M.G., Mateus, G.R.: The metaheuristic GRASP as an upper bound for a branch and bound algorithm in a scheduling problem with non-related parallel machines and sequence-dependent setup times. In: Proceedings of the 4th EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-Heuristics, vol. 1, pp. 62–67 (2004)
67. Santos, L.F., Martins, S.L., Plastino, A.: Applications of the DM-GRASP heuristic: A survey. In: International Transactions on Operational Research (2008)
68. Santos, L.F., Milagres, R., Albuquerque, C.V., Martins, S., Plastino, A.: A hybrid GRASP with data mining for efficient server replication for reliable multicast. In: 49th Annual IEEE GLOBECOM Technical Conference (2006)
69. Santos, L.F., Ribeiro, M.H., Plastino, A., Martins, S.L.: A hybrid GRASP with data mining for the maximum diversity problem. In: Blesa, M.J., Blum, C., Roli, A., Sampels, M. (eds.) HM 2005. LNCS, vol. 3636, pp. 116–127. Springer, Heidelberg (2005)
70. Sosnowska, D.: Optimization of a simplified fleet assignment problem with metaheuristics: Simulated annealing and GRASP. In: Pardalos, P.M. (ed.) Approximation and complexity in numerical optimization. Kluwer Academic Publishers, Dordrecht (2000)
71. 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., Sousa, J. (eds.) Metaheuristics: Computer Decision-Making, pp. 627–658. Kluwer Academic Publishers, Dordrecht (2004)
72. Voss, S., Martello, S., Osman, I.H., Roucairo, C. (eds.): Meta-heuristics: Andvances and trends in local search paradigms for optimization. Kluwer Academic Publishers, Dordrecht (1999)