METAHEURISTIC HYBRIDIZATION WITH GRASP

MAURICIO G. C. RESENDE

Abstract. GRASP, or greedy randomized adaptive search procedure, is a multi-start metaheuristic that repeatedly applies local search starting from solutions constructed by a randomized greedy algorithm. In this chapter we consider ways to hybridize GRASP to create new and more effective metaheuristics. We consider several types of hybridizations: constructive procedures, enhanced local search, memory structures, and cost reformulations.

1. Introduction

Combinatorial optimization 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 $f : 2^E \to \mathbb{R}$. Throughout this chapter, we consider the minimization version of the problem, where we search for an optimal solution $S^* \in F$ such that $f(S^*) \leq f(S),\ \forall S \in F$. The ground set E, the cost function f, and the set of feasible solutions F are defined for each specific problem. For instance, in the case of the traveling salesman problem, the ground set E is that of all edges connecting the cities to be visited, $f(S)$ is the sum of the costs of all edges $e \in S$, and F is formed by all edge subsets that determine a Hamiltonian cycle.

Combinatorial optimization finds applications in many settings, including routing, scheduling, inventory and production planning, and facility location. These problems arise in real-world situations such as in transportation (air, rail, trucking, shipping), energy (electrical power, petroleum, natural gas), and telecommunications (design, location, operation).

While much progress has been made in finding provably optimal solutions to combinatorial optimization problems employing techniques such as branch and bound, cutting planes, and dynamic programming, as well as provably near-optimal solutions using approximation algorithms, many combinatorial optimization problems arising in practice benefit from heuristic methods that quickly produce good-quality solutions. Many modern heuristics for combinatorial optimization are based on guidelines provided by metaheuristics.

Metaheuristics are high level procedures that coordinate simple heuristics, such as local search, to find solutions that are of better quality than those found by the simple heuristics alone. Many metaheuristics have been introduced in the last thirty years [40]. Among these, we find genetic algorithms, simulated annealing, tabu search, variable neighborhood search, scatter search, path-relinking, iterated

Date: April 14, 2008. Revised May 23, 2008.

Key words and phrases. GRASP, hybrid heuristics, metaheuristics, path-relinking, Lagrangian relaxation, variable neighborhood descent, tabu search, simulated annealing, iterated local search.

AT&T Labs Research Technical Report.

TRAVELING SALESMAN PROBLEM: Given a directed graph $G(V, E)$ with costs associated with its edges, and a starting vertex $v_0 \in V$, find a least-cost route that starts in v_0, visits each other vertex exactly once, and then returns to v_0.

local search, ant colony optimization, swarm optimization, and greedy randomized adaptive search procedures (GRASP).

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. These approaches are commonly referred to as *hybrid metaheuristics*. The main motivation to hybridize metaheuristics is to make up for the shortcomings of one metaheuristic with special characteristics of the other. As we will see later, pure GRASP lacks a memory mechanism that enables good solutions found in earlier iterations of the search to influence the search later. GRASP hybridized with path-relinking, however, overcomes this shortcoming by using a very effective memory mechanism to intensify the search near good-quality solutions. As a result, GRASP with path-relinking is more effective than simple GRASP or simple path-relinking. It is also more efficient. In this chapter we illustrate hybrid metaheuristics by looking at ways to hybridize GRASP with other metaheuristics and optimization strategies for finding optimal or near-optimal solutions of combinatorial optimization problems. Of course, hybridizations can be made with metaheuristics other than GRASP, as is discussed in Raidl [70], where a unified view of hybrid metaheuristics is presented.

The chapter is organized as follows. In Section 2 we review the metaheuristic GRASP. Hybrid construction schemes are considered in Section 3 and hybrid local search schemes in Section 4. Section 5 discusses hybridization of GRASP with path-relinking. Other types of hybridizations are considered in Section 6. Concluding remarks are made in Section 7.

2. A review of GRASP

A GRASP, or greedy randomized adaptive search procedure, repeatedly applies local search, starting from solutions that are constructed using a randomized greedy algorithm. The best local optimum found over all local searches is returned as the solution of the heuristic. An especially appealing characteristic of GRASP is the ease with which it can be implemented. Few parameters need to be set and tuned, and therefore development can focus on implementing algorithms and data structures to assure efficiency. As we will see later, basic implementations of GRASP rely exclusively on two parameters. The first controls the number of construction / local search iterations that will be applied and the second controls the blend of randomness and greediness in the solution construction procedure. In spite of its simplicity and ease of implementation, GRASP is a very effective metaheuristic and produces the best known solutions for many problems.

GRASP was first introduced by Feo and Resende [26]. See Feo and Resende [27], Pitsoulis and Resende [67], and Resende and Ribeiro [75] for surveys of GRASP and Festa and Resende [31, 32] for annotated bibliographies.

This section is organized as follows. We first discuss the concept of neighborhood solution space, needed to understand the basics of local search. This is followed

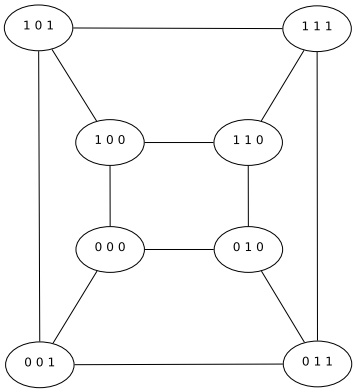


FIGURE 1. 1-flip neighborhood solution space graph $\mathcal{X}$ of dimension 3.

by an examination of local search and the tradeoffs observed by using different neighborhood structures in local search. Then we discuss how to add diversification into local search by using procedures that blend randomness and greediness to construct a diverse set of good starting solutions for local search.

2.1. **Neighborhood solution space.** Consider the *neighborhood solution space* graph $\mathcal{X} = (S, M)$, where the node set S represents all feasible solutions of a combinatorial optimization problem and the edge set M corresponds to *moves* connecting neighboring solutions. A solution $s \in S$ is in the neighborhood $N(t)$ of a solution $t \in S$ if s can be obtained from t by making a small predefined change in t. If a solution $s \in N(t)$, then $t \in N(s)$, $(s,t) \in M$, and $(t,s) \in M$. Different neighborhood structures can be defined for a given combinatorial optimization problem. For example, suppose a solution is represented as an indicator n-vector of zeroes and ones. Let the distance $d(s,t)$ between two solutions s and t be their Hamming distance, i.e. the number of components where the two solutions differ. The *1-flip* neighborhood of s is the set of all solutions t such that $d(s,t) = 1$. The size of the 1-flip neighborhood is n. Figure 1 shows the 1-flip neighborhood solution space graph for $[0,1]^3$. Neighborhood sizes vary from small, such as the 1-flip neighborhood, to neighborhood that are exponentially large. For example, the k-flip neighborhood of s, the set of all solutions t such that $d(s,t) = k$, has size $\binom{n}{k}$. Later in this chapter, we consider other very large neighborhoods that are incorporated into a local search scheme called *very large neighborhood search* [3].

```
begin LocalSearch(s_0 ∈ S)
1   t ← s_0;
2   while there exists s ∈ N(t) such that f(s) < f(t) do
3       t ← s;
4   end-while;
5   return t;
end
```

FIGURE 2. Standard local search algorithm.

```
begin RandomizedGreedy
1   S ← ∅;
2   C ← E = { ground set };
3   while |C| > 0 do
4       For all c ∈ C compute greedy function value g(c);
5       Define RCL(C) ← {c ∈ C | g(c) has a low value};
6       Select at random c* ∈ RCL(C);
7       Add c* to partial solution: S ← S ∪ {c*};
8       Let C be the set of ground set elements that can
        be added to S;
9   end-while;
end
```

FIGURE 3. GRASP greedy randomized construction: randomizing the greedy algorithm, or semi-greedy algorithm.

2.2. **Local search.** Local search (see Figure 2) starts at some node $s_0 \in S$ and makes it the current solution t, i.e. $t = s_0$. At each iteration, it seeks a neighboring solution having a better objective function value than the current solution, i.e. a solution $s \in N(t)$ such that $f(s) < f(t)$. If such a solution is found, it becomes the current solution, i.e. $t = s$. These iterations are repeated until there is no better solution in the neighborhood $N(t)$ of the current solution t. In this case, solution t is called a *local optimum*. Local search can be thought of as starting at a node $s \in S$ and examining adjacent nodes in graph $\mathcal{X}$ for an improving solution. In the *first-improving* variant, local search moves to the first improving solution that it finds, whereas in the *best-improving* version, it examines all of the neighbors and moves to the best one. Clearly, the complexity of carrying out one step of local search depends on the size of the neighborhood. Likewise, the likelihood that an improving solution will be found also depends on the size of the neighborhood. There is a tradeoff that needs to be explored. Small neighborhoods, such as the 1-flip, can be explored quickly but may not lead to optimal solutions, while large neighborhoods, such as k-flip, are more likely to contain an optimal solution, or lead to one, but are often exponentially large and expensive to explore. In this chapter we will examine this tradeoff. One can think of this part of the search as *intensification*, since we concentrate on a small portion of the solution space.

2.3. **Diversification of starting solutions for local search.** An effective search method also needs to enable *diversification*. In carrying out local search, a good strategy should not focus the search on one particular region of the solution space, for example around a solution built with a greedy algorithm. One alternative is to start local search from many randomly generated solutions with the expectation that there is a cost-improving path in $\mathcal{X}$ from one of those solutions to an optimal or near-optimal solution. A downside to starting at randomly generated solutions is the average low quality of such solutions and the large average length of cost-improving paths in $\mathcal{X}$ from them to optimal or near-optimal solutions. Long paths imply many steps and consequently long running times, as well as many places to make mistakes and take a detour to a bad local optimum.

```
begin SampleGreedy(p)
1   S ← ∅;
2   C ← E = { ground set };
3   while |C| > 0 do
4       Randomly sample min{p, |C|} elements from C
        and put them in RCL(C);
5       Select c* = argmin{g(c) | c ∈ RCL(C)};
6       Add c* to partial solution: S ← S ∪ {c*};
7       Let C be the set of ground set elements that can
        be added to S;
8   end-while;
end
```

FIGURE 4. GRASP greedy randomized construction: sample greedy.

2.4. **Greedy randomized construction.** The greedy randomized construction methods of GRASP seek to produce a diverse set of good-quality starting solutions from which to start local search. This is achieved by adding randomization to the greedy algorithm. We illustrate here two ways to do this. Solutions are built by adding one ground set element at a time to a partially constructed solution. In the first construction scheme, called *semi-greedy algorithm* by Hart and Shogan [47], at each step let the candidate set C denote all of the remaining ground set elements that can be added to the partial solution and let $RCL(C)$ be a *restricted candidate list* made up of high-quality candidate elements. The quality of a candidate element is determined by its contribution, at that point, to the cost of the solution being constructed. A *greedy function* $g(c)$ measures this contribution for each candidate $c \in C$. Membership can be determined by rank or by quality relative to other candidates. Membership by rank, also called *cardinality based*, is achieved if the candidate is one of the q candidates with smallest greedy function value, where q is an input parameter that determines how greedy or random the construction will be. Membership by *quality* relative to other candidates determines a greedy function cutoff value and only considers candidates with a greedy value no greater than the cutoff. To implement this, one usually makes use of a real-valued RCL parameter $\alpha \in [0, 1]$. Let $\acute{g} = \max\{g(c) \mid c \in C\}$ and $\grave{g} = \min\{g(c) \mid c \in C\}$. A candidate $c \in C$ is placed in the RCL only if $\grave{g} \leq g(c) \leq \grave{g} + \alpha \cdot (\acute{g} - \grave{g})$. Input parameter α determines how greedy or random the construction will be. Of these restricted candidates, one is selected at random and added to the partial solution. The construction is repeated until there are no further candidates. Figure 3 shows pseudo-code for this construction procedure.

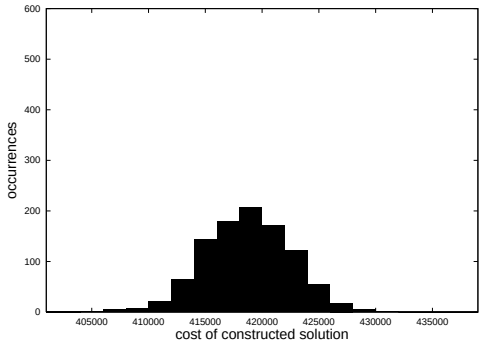
In the second construction scheme, called *sample greedy* by Resende and Werneck [77], instead of randomizing the greedy algorithm, a greedy algorithm is applied to each solution in a random sample of candidates. At each step a fixed-size subset of the candidates in C is sampled and the incremental contribution 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. Figure 4 shows pseudo-code for this construction procedure.

Calling `RandomizedGreedy` the chosen randomized greedy construction procedure, Figure 8 shows pseudo-code for a generic GRASP. At the completion of the randomized greedy phase the solution on hand is, for many problem instances, feasible. However, for some problem instances, it may be infeasible and require that a repair procedure be applied to restore feasibility. Examples of GRASP implementations where repair procedures were needed to restore feasibility of the constructed solution can be found in Duarte, Ribeiro, and Urrutia [24], Duarte et al. [25], and Nascimento, Resende, and Toledo [63].

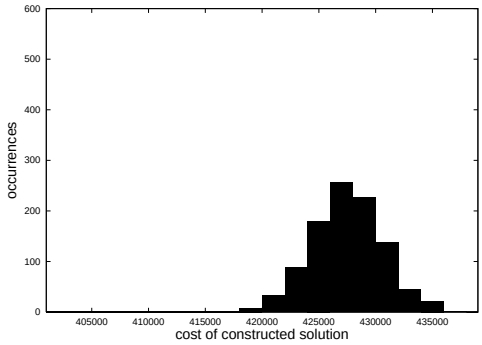
GRASP may be viewed as a repetitive sampling technique. Each iteration produces a sample solution from an unknown distribution, whose mean and variance are functions of the restrictive nature of the RCL. For example, if the RCL is restricted to a single element, then the same solution will be produced at all iterations. The variance of the distribution will be zero and the mean will be equal to the value of the greedy solution. If the RCL is allowed to have more elements, then many different solutions will be produced, implying a larger variance. Since greediness plays a smaller role in this case, the mean solution value should be worse. However, the value of the best solution found outperforms the mean value and very often is optimal. The histograms in Figure 5 illustrate this situation on an instance of MAXSAT with 100 variables and 850 clauses, depicting results obtained with 1000 independent constructions using the first phase of the GRASP described in [73, 74]. Since this is a maximization problem, the purely greedy construction corresponds to $\alpha = 1$, whereas the random construction occurs with $\alpha = 0$. We notice that when the value of α increases from 0 to 1, the mean solution value increases towards the purely greedy solution value, while the variance approaches zero.

For each value of α, we present in Figure 6 histograms with the results obtained by applying local search to each of the 1000 constructed solutions. Figure 7 summarizes the values observed for the total processing time and the local search time. We notice that both time measures considerably decrease as α tends to 1, approaching the purely greedy choice. In particular, we observe that the average local search time taken by $\alpha = 0$ (purely random) is approximately 2.5 times that taken in the case $\alpha = 0.9$ (almost greedy). In this example, two to three greedily constructed solutions can be investigated in the same time needed to apply local search to one single randomly constructed solution. The appropriate choice of the value of the RCL parameter α is clearly critical and relevant to achieve a good balance between computation time and solution quality. It is unlikely that GRASP will find an optimal solution if the average solution value is low, even if there is a large variance in the overall solution values, such as is the case for $\alpha = 0$. On the other hand, if there is little variance in the overall solution values, it is also unlikely that GRASP will find an optimal solution, even if the average solution is high, as is the case for $\alpha = 1$. What often leads to good solutions are relatively high average solution values in the presence of a relatively large variance, such as is the case for $\alpha = 0.8$. Very often, many GRASP solutions are generated in the same amount of time required for the local optimization procedure to converge from a single random start.

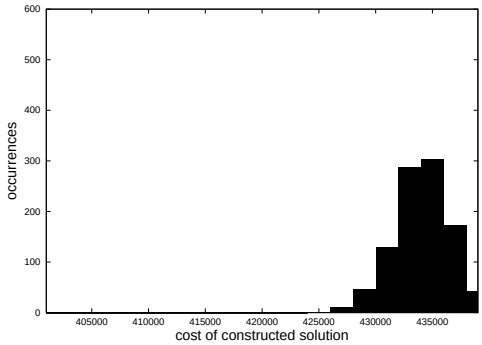
To show some solutions produced by GRASP, consider the maximum independent set problem. Feo, Resende, and Smith [28, 71] describe a GRASP for this problem where the independent set is built, one vertex at a time until a maximal independent set is produced. At each step of the construction, the set of candidate



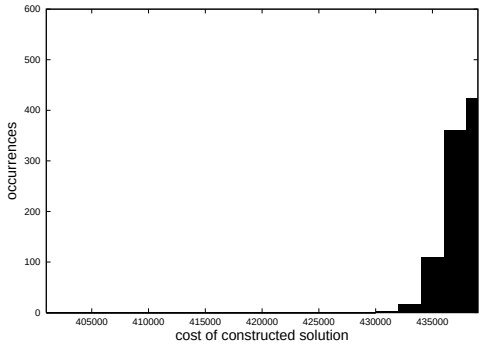
(a) RCL parameter alpha = 0.0 (random construction)



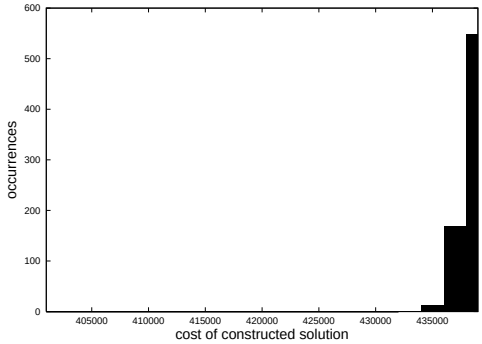
(b) RCL parameter alpha = 0.2



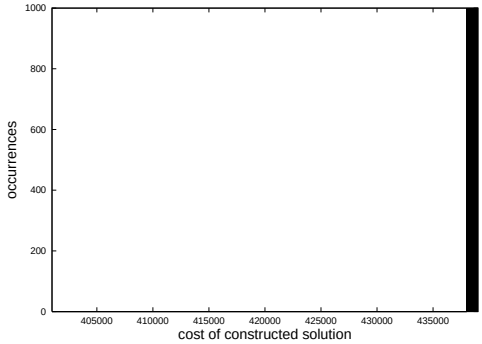
(c) RCL parameter alpha = 0.4



(d) RCL parameter alpha = 0.6

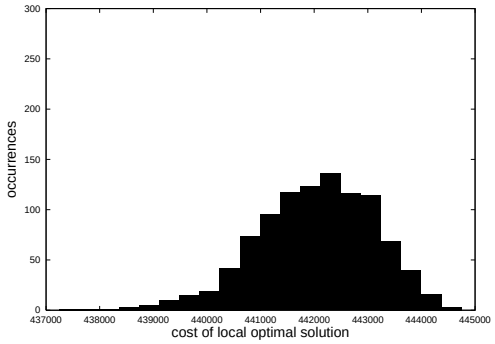


(e) RCL parameter alpha = 0.8

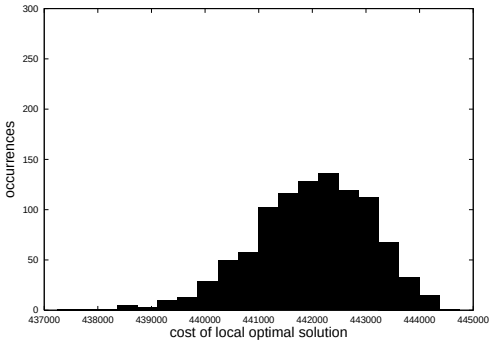


(f) RCL parameter alpha = 1.0 (greedy construction)

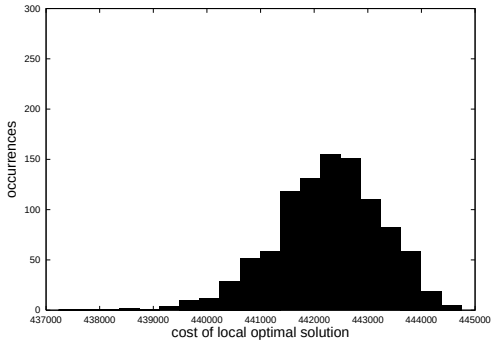
FIGURE 5. Distribution of construction phase solutions as a function of the RCL parameter α (1000 repetitions were recorded for each value of α).



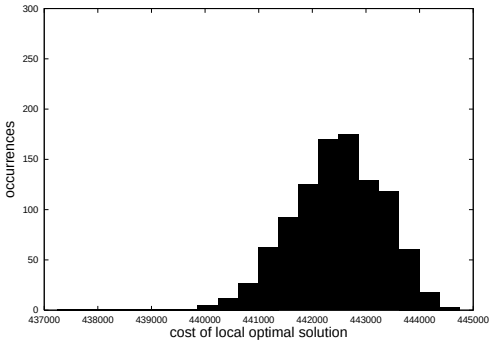
(a) RCL parameter alpha = 0.0 (random)



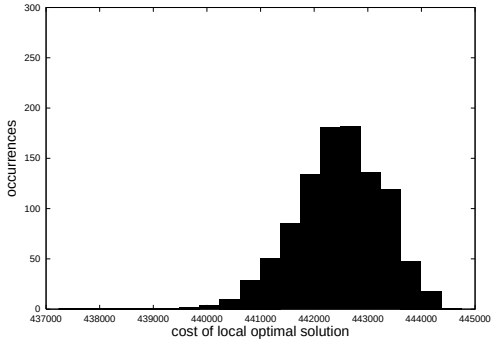
(b) RCL parameter alpha = 0.2



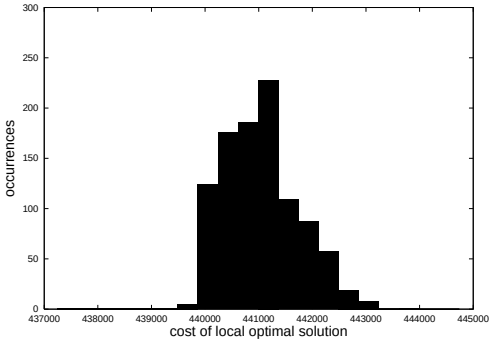
(c) RCL parameter alpha = 0.4



(d) RCL parameter alpha = 0.6



(e) RCL parameter alpha = 0.8



(f) RCL parameter alpha = 1.0 (greedy)

FIGURE 6. Distribution of local search phase solutions as a function of the RCL parameter α (1000 repetitions for each value of α).

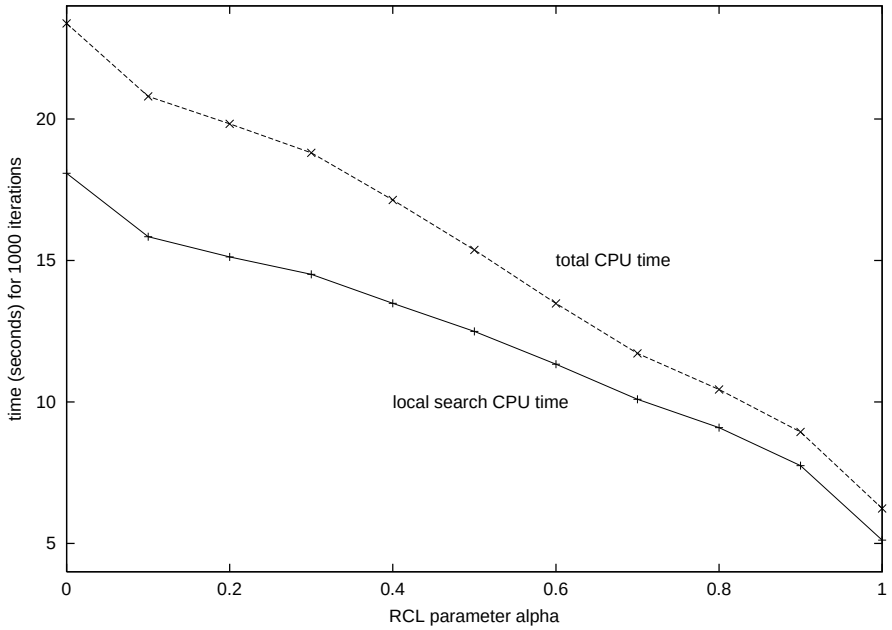


FIGURE 7. Total CPU time and local search CPU time as a function of the RCL parameter α (1000 repetitions for each value of α) for a maximization problem.

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

FIGURE 8. A basic GRASP in pseudo-code.

MAXIMUM INDEPENDENT SET PROBLEM: Given an undirected graph, find the largest subset of mutually nonadjacent vertices.

vertices to be added to the partial solution is made up of vertices that are nonadjacent to all vertices in the partial solution and the greedy function is the degree of the candidate vertex with respect to the other candidate vertices. The local search

is a (k, l)-exchange where k vertices are removed from the independent set and $l > k$ vertices are added to it. Figure 9 shows solutions produced by the construction and local search phases of GRASP on a ten-node instance of the maximum independent set problem where the local search used is a $(2, 3)$-exchange.

3. Hybrid construction schemes

As discussed in Section 2, the central idea behind GRASP is the repeated application of local search starting from a diverse set of good-quality starting solutions. We illustrated the starting solution generation with two schemes for generating such starting solutions: randomized greedy and sample greedy. In this section, we consider a few other schemes that can be hybridized within a GRASP.

3.1. **More randomized greedy schemes.** A possible shortcoming of the greedy randomized construction based on restricted candidate list is its complexity. At each step of the construction, each yet unselected candidate element has to be evaluated by the greedy function. In cases where the difference between the number of elements in the ground set and the number of elements that appear in a solution is large, this may not be very efficient. Resende and Werneck [77] introduced the *random plus greedy* and the *sample greedy* construction schemes to address this shortcoming.

In random plus greedy, a portion of the solution is constructed by randomly choosing p candidate elements and the remaining solution is completed in a greedy fashion. The resulting solution is randomized greedy. The value of p determines how greedy or random the construction will be. Small values of p lead to more greedy solutions where large values lead to ones that are more random. Sample greedy was presented in Subsection 2.4. It also uses a parameter p to control the balance between greediness and randomness in the construction. Small values of p lead to more random solutions, while large values lead to more greedy solutions.

Resende and Werneck [77] also introduced the *proportional greedy* construction scheme. In each iteration of proportional greedy, we compute the greedy function $g(c)$ for every candidate element $c \in C$ and then pick a candidate at random, but in a biased way: the probability of a given candidate $c' \in C$ being selected is inversely proportional to $g(c') - \min\{g(c)|c \in C\}$.

3.2. **Reactive GRASP.** In randomized greedy construction procedures, the algorithm designer must decide how to balance greediness and randomness. A simple approach is to balance at random. For example, in the semi-greedy construction with membership in the RCL by quality relative to the other candidates, the parameter α can be selected uniformly at random from the interval $[0, 1]$ so that each GRASP iteration uses a different α value and therefore has a different balance between greediness and randomness. Prais and Ribeiro [68] showed that using a single fixed value for the value of RCL parameter α in the membership by quality relative to other candidates of the semi-greedy algorithm often hinders finding a high-quality solution, which eventually could be found if another value was used. They proposed an extension of the basic GRASP procedure, which they call *Reactive GRASP*, in which the parameter α is not fixed, but instead is selected at each iteration from a discrete set of possible values. The solution values found along the previous iterations serve a guide for the selection process. Prais and Ribeiro [68] define $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ to be the set of possible values for α. The

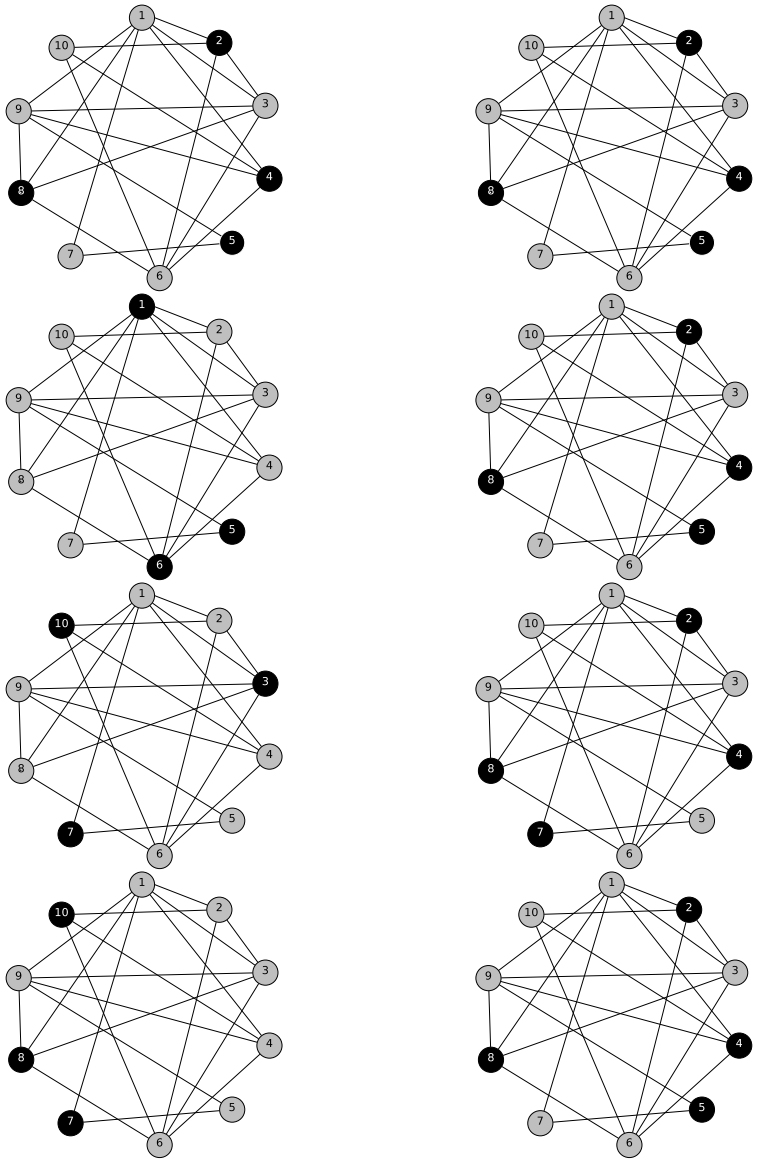


FIGURE 9. Four iterations of GRASP for maximum independent set. The solutions on the left are maximal and were constructed with a randomized greedy algorithm. The corresponding solutions on the right are produced by applying a $(2,3)$-exchange local search on the constructed solution. Since the size of the maximum independent set for this instance is four, all locally optimal solutions found are also globally optimal.

probabilities associated with the choice of each value are all initially made equal to $p_i = 1/m,\ i = 1, \ldots, m$. Furthermore, let 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 by taking $p_i = q_i / \sum_{j=1}^{m} q_j$, with $q_i = z^*/A_i$ for $i = 1, \ldots, m$. The value of q_i will be larger for values of $\alpha = \alpha_i$ leading to the best solutions on average. 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.

3.3. **Long-term memory in construction.** Though reactive GRASP uses long-term memory (information gathered in previous iterations) to adjust the balance of greediness and randomness, Fleurent and Glover [33] observed that the basic GRASP does not and proposed a long-term memory scheme to address this issue in multi-start heuristics. Long-term memory is one of the fundamentals on which tabu search [36, 37] relies. Their scheme maintains a pool of elite solutions to be used in the construction phase. Later in this chapter we discuss pools of elite solutions in detail. For now, all we need to know is that to become an elite solution, a solution must be either better than the best member of the pool, or better than its worst member and sufficiently different from the other solutions in the pool. Fleurent and Glover [33] define a *strongly determined variable* to be one that cannot be changed without eroding the objective or changing significantly other variables and a *consistent variable* to be one that receives a particular value in a large portion of the elite solution set. Let $I(e)$ be a measure of the strongly determined and consistent features of solution element e from the ground set E. $I(e)$ becomes larger as e appears more often in the pool of elite solutions. It is used in the construction phase as follows. Recall that $g(e)$ is the greedy function value for candidate $e \in C$, i.e. the incremental cost associated with the incorporation of element $e \in C$ into the solution under construction. Let $K(e) = F(g(e), I(e))$ be a function of the greedy and the intensification functions. For example, $K(e) = \lambda g(e) + I(e)$. The intensification scheme biases selection from the set C of candidate solutions to those elements $e \in C$ with a high value of $K(e)$ by setting their selection probability to be $p(e) = K(e) / \sum_{s \in \mathrm{RCL}} K(s)$. The function $K(e)$ can vary with time by changing the value of λ, e.g. initially λ may be set to a large value that is decreased when diversification is called for.

3.4. **Biased sampling.** Another way to depart from the uniform selection of candidate elements in the construction of a greedy randomized solution is to follow the strategy proposed in the *heuristic-biased stochastic sampling* scheme of Bresina [14]. Instead of choosing the next candidate element to add to the partial solution uniformly at random, heuristic-biased stochastic sampling suggests that any probability distribution can be used to bias the selection toward some particular candidates. In the construction mechanism proposed by Bresina [14], a family of such probability distributions is introduced. They are based on the rank $r[\sigma]$ assigned to each candidate element σ, according to its greedy function value. The element with the smallest greedy function value has rank 1, the second smallest has rank 2, and so on. Several bias functions $b(\cdot)$ are introduced by Bresina, such as random bias where $b(r) = 1$, linear bias where $b(r) = 1/r$, log bias where $b(r) = \log^{-1}(r+1)$, exponential bias where $b(r) = e^{-r}$, and polynomial bias of order n where $b(r) = r^{-n}$. Once all elements of the RCL have been ranked, the probability $\pi(\sigma)$ of selecting

STEINER PROBLEM IN GRAPHS: Given an undirected graph $G(V, E)$ with costs associated with its edges and a subset $S \subseteq V$ of the vertices, find a tree of minimum weight which spans all vertices in S, where the weight of the tree is the sum of the costs of the edges in the tree.

PRIZE-COLLECTING STEINER PROBLEM IN GRAPHS: Given an undirected graph with penalties associated with its vertices and costs associated with its edges, find a tree of minimum weight, where the weight of the tree is the sum of the costs of its edges plus the sum of the penalties of the vertices not spanned by the tree.

element $\sigma \in \text{RCL}$ can be computed as $\pi(\sigma) = b(r[\sigma])/(\sum_{\sigma' \in \text{RCL}} b(r[\sigma']))$. In this scheme one can restrict candidates to be selected from the RCL or from the entire set of candidates, i.e. make $\text{RCL} = C$.

3.5. **Cost perturbation.** The idea of introducing noise into the original costs as a way of randomizing a solution construction procedure is similar to that in the so-called *noising method* of Charon and Hudry [16, 17]. It can be more effective than the greedy randomized construction of the basic GRASP procedure in circumstances where the construction algorithms are insensitive to standard randomization strategies, such as selecting an element at random from a restricted candidate list. Ribeiro, Uchoa, and Werneck [81] showed that this is the case for the shortest-path heuristic of Takahashi and Matsuyama [88], used as one of the main building blocks of the construction phase of a hybrid GRASP they proposed for the Steiner problem in graphs. Another situation where cost perturbations can be effective arises when no greedy algorithm is available for straightforward randomization as was the case of the hybrid GRASP developed by Canuto, Resende, and Ribeiro [15] for the prize-collecting Steiner tree problem, which makes use of the primal-dual approximation algorithm of Goemans and Williamson [43] to build initial solutions using perturbed costs.

The cost perturbation methods used in the GRASP for the minimum Steiner tree problem incorporate learning mechanisms associated with intensification and diversification strategies, originally proposed in the context of tabu search. Let w_e denote the weight of edge e. Three distinct weight randomization methods (D, I, U) are applied. At a given GRASP iteration i, the modified weight w_e^i of each edge e is randomly selected from a uniform distribution between w_e and $r_i(e) \cdot w_e$, where the coefficient $r_i(e)$ depends on the selected weight randomization method applied at iteration i. Let $t_{i-1}(e)$ be the number of locally optimal solutions in which edge e appeared, after $i-1$ iterations of the hybrid GRASP procedure have been performed. Clearly, $0 \leq t_{i-1}(e) \leq i-1$.

In method D, values of the coefficients $r_i(e) = 1.25 + 0.75 \cdot t_{i-1}(e)/(i-1)$ are larger for edges which appeared more frequently in previously found local optima. This scheme leads to a diversification strategy, since more frequently used edges are likely to be penalized with stronger augmentations. Contrarily, method I is an intensification strategy penalizing less frequent edges with larger coefficients $r_i(e) = 2 - 0.75 \cdot t_{i-1}(e)/(i-1)$. Finally, the third randomization method U uses a uniform penalization strategy, independent of frequency information, i.e. it uses

$r_i(e) = 2$. The original weights without any penalization are used in the first three iterations, combined with three different construction heuristics. The weight randomization methods are then cyclically applied, one at each of the remaining iterations, starting with method I, next D, then U, then I again, and so on. The alternation between diversifying (method D) and intensifying (method I) iterations characterizes a strategic oscillation approach [39].

In the case of the GRASP for the prize-collecting Steiner tree problem, a new solution is built at each iteration using node penalties updated by a perturbation function, according to the structure of the current solution. Two penalty perturbation schemes are used. *Perturbation by eliminations* enforces search diversification by forcing the approximation algorithm to build a new solution without some of the nodes appearing in the solution constructed in the previous iteration. This is done by changing to zero the penalties of some persistent nodes, which appeared in the last solution built and remained at the end of the local search. A parameter ρ controls the fraction of the persistent nodes whose penalties are temporarily set to zero. *Perturbation by penalty changes* forces the approximation algorithm to build different, but still good, solutions by introducing noise into the node penalties in a way similar to what is proposed in Charon and Hudry [16, 17]. For each node i, a perturbation factor $\beta(i)$ is randomly generated in the interval $[1-a, 1+a]$, where a is an implementation parameter. The penalty associated with node i is temporarily changed to $\pi'(i) = \pi(i) \cdot \beta(i)$, where $\pi(i)$ is its original penalty.

4. Hybrid local search schemes

The basic local search schemes described in Section 2 take an initial solution, make it the current solution, and search the current solution's neighborhood for an improving solution. If one is found, it is made the current solution and local search is recursively applied to the current solution. The procedures terminate when no better solution exists in the current solution's neighborhood. Such local search schemes require that the size of the neighborhood be such that its exploration can be done efficiently.

Several approaches have been proposed to extend the above local search scheme. These 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.

4.1. **Variable neighborhood descent.**

Variable neighborhood descent (VND) allows the systematic exploration of multiple neighborhoods [45]. As opposed to variable neighborhood search, which we examine later in this section and for which the neighborhoods are parameterized, the neighborhoods in VND need not be related. VND 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 [46]. 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 $N_k(t)$, for $k = 1, \ldots, K$ be K neighborhood structures of solution $t \in S$. The pseudo-code in Figure 10 illustrates this scheme. As usual, the search begins with a given starting solution $s_0 \in S$ which is made the current solution t. Each major

```
begin VND(s_0 ∈ S)
1   t ← s_0; flag ← true;
2   while flag do
3       flag ← false;
4       for k = 1, ..., K do
5           if there exists s ∈ N_k(t) such that f(s) < f(t) then
6               t ← s; flag ← true; break;
7           end-if
8       end-for
9   end-while;
10  return t;
end
```

FIGURE 10. Variable neighborhood descent (VND) on K neighborhood structures $N_1, N_2, \ldots, N_k$.

iteration (lines 2 to 9) searches for an improving solution s in up to K neighborhoods of t. If no improving solution is found in any of the K neighborhoods, the search ends. Otherwise, s is made the current solution and the search is applied starting from the current solution.

GRASP with VND is the result of replacing the standard local search in a GRASP by a VND with two or more neighborhood structures. Applications of VND within GRASP include Martins et al. [61], Ribeiro and Souza [80], Ribeiro, Uchoa, and Werneck [81], Ribeiro and Vianna [83], and Andrade and Resende [7].

4.2. **Variable neighborhood search.** Another strategy based on searching multiple neighborhoods is *variable neighborhood search*, or VNS [45]. Whereas in variable neighborhood descent, the neighborhoods may be unrelated, in VNS the K neighborhoods are parameterized such that $|N_1(s)| \leq |N_2(s)| \leq \cdots \leq |N_K(s)|$, for all $s \in S$. The 1-flip, 2-flip, ..., K-flip neighborhoods constitute an example of K such neighborhoods.

The standard VNS combines deterministic and stochastic elements to search the K neighborhoods. The pseudo-code in Figure 11 illustrates this method. In the k-th step of the inner loop of VNS, a solution s' is chosen at random from the $N_k(t)$ neighborhood of the incumbent $t \in S$. This operation is often referred to as *shaking*. The standard local search is done on the $N_1(s')$ neighborhood of s' in search for a solution that is better than the incumbent solution t. If an improving solution is found, it is made the incumbent and the process restarts with $k = 1$. Otherwise, k is incremented and a new step begins. The process stops when all K neighborhoods have been searched without producing a new incumbent solution.

Whereas in VND we require that the neighborhoods be small enough so they can be searched efficiently, in VNS only neighborhood N_1 should be small since it is the only neighborhood that is searched. The only practical requirement is that it be easy to randomly select a solution from each neighborhood $N_2, N_3, \ldots, N_K$.

Examples of GRASP with VNS include Canuto, Resende, and Ribeiro [15], Ochi, Silva, Drummond [64], Drummond et al. [23], Festa et al. [30], and Beltrán et al. [13].

```
begin VNS(s_0 ∈ S)
1   t ← s_0; flag ← true;
2   while flag do
3       flag ← false;
4       for k = 1, ..., K do
5           s' ← a randomly selected solution ∈ N_k(t);
6           if there exists s ∈ N_1(s') such that f(s) < f(t) then
7               t ← s; flag ← true; break;
8           end-if
9       end-for
10  end-while;
11  return t;
end
```

FIGURE 11. Variable neighborhood search (VNS) on K neighborhood structures $N_1, N_2, \ldots, N_k$.

```
begin TS(s_0 ∈ S)
1   t ← s_0; f* ← ∞; T ← ∅;
2   while stopping criterion not satisfied do
3       Ñ(t) ← N(t) \ {s ∈ N(t) | s can be reached from t by move in tabu list T};
4       s ← a least-cost solution ∈ Ñ(t);
5       Add to tabu list T the reverse move from s to t;
6       Remove from tabu list T the oldest entry if needed;
7       if f(s) < f* then
8           f* ← f(s); t* ← s;
9       end-if
10      t ← s;
11  end-while;
12  return t*;
end
```

FIGURE 12. Short memory tabu search (TS).

4.3. **Short-term memory tabu search.** Tabu search [36, 37] is a search strategy that makes use of memory structures to enable escape from local minima by allowing cost-increasing moves. Short-term memory tabu search makes use of the short-term memory structure tabu list. The process starts from a given solution and moves in steps from the current solution $s \in S$ to some solution $t \in N(s)$. To avoid returning to a just-visited local minimum, reverse moves that lead back to that local minimum are forbidden, or made tabu, for a number of steps. This is usually implemented by making tabu all moves that reverse the effect of recent moves [34].

Figure 12 shows pseudo-code for a short-term tabu search procedure that 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

```
begin SA(s_0 ∈ S)
1   t ← s_0; f* ← ∞; k ← 1; T_1 ← T_0;nf ← 0;
2   while nf < F̄ do
3       nc ← 0; nt ← 0;
4       while nc < C̄ and nt < T̄ do
5           nt ← nt + 1;
6           Select s at random from N(t);
7           if f(s) ≤ f(t) then
8               t ← s; nc ← nc + 1;
9           else
10              With probability e^{-(f(s)-f(t))/T_k}:
11                  t ← s; nc ← nc + 1;
12          end-if
13          if f(t) ≤ f* then
14              t* ← t; f* ← f(t);
15          end-if
16      end-while;
17      T_{k+1} ← T_k · γ;
18      k ← k + 1;
19      if nc < C̲ then nf ← nf + 1;
20  end-while;
21  return t*;
end
```

FIGURE 13. Simulated annealing (SA).

can balance the amount 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 Laguna and González-Velarde [53], Delmaire et al. [21], Abdinnour-Helm and Hadley [1], Serra and Colomé [86], Souza, Maculan, and Ochi [87], Souza, Duhamel, and Ribeiro [20], Li et al. [55], Lim and Wang [56], Moura and Oliveira [62], and Pu et al. [69].

4.4. **Simulated annealing.** Simulated annealing [52] can also be used to extend the reach of the standard local search. Simulated annealing is given a starting solution $s_0 \in S$ which is used to initialize the current solution $t \leftarrow s_0$. At each step, it randomly selects a *trial* solution $s \in N(t)$. If $f(s) \leq f(t)$, then s is made the current solution, i.e. $t \leftarrow s$. On the other hand, if $f(s) > f(t)$, then with probability $e^{-(f(s)-f(t))/T_k}$ it is made the current solution, where T_k is a control parameter called the *temperature*. There are many ways to implement simulated annealing. Johnson et al. [49, 50] propose running simulated annealing in cycles. In each cycle the temperature remains constant. A *cooling schedule*, such as $T_{k+1} \leftarrow T_k \cdot \gamma$, where $0 < \gamma < 1$, adjusts the temperature from one cycle to the next. In each cycle, at most $\bar{T}$ trial solutions are sampled. A *change* takes place if a trial solution is accepted. At most $\bar{C}$ changes are allowed for a fixed temperature. The algorithm terminates when $\bar{F}$ temperature cycles go by with less than $\underline{C}$ changes in the current solution t. The pseudo-code in Figure 13 shows this implementation of simulated annealing.

```
begin ILS(s_0 ∈ S)
1   t ← LocalSearch(s_0);
2   while stopping criterion not satisfied do
3       t̂ ← Perturbation(t, history);
4       t̂* ← LocalSearch(t̂);
5       t ← AcceptanceCriterion(t, t̂*, history);
6   end-while;
7   return t;
end
```

FIGURE 14. Iterated local search (ILS).

Simulated annealing usually starts with a high temperature, i.e. one that makes it accept, in the initial cycles, most non-improving solutions. The initial cycles constitute 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 considered intensification cycles. To make use of simulated annealing as a substitute for the standard local search in GRASP, one should limit the search to the intensification part. This can be done by starting with a cool temperature. Load balancing with the construction procedure can be achieved by adjusting the simulated annealing parameters T_0, $\bar{F}$, $\bar{C}$, $\underline{C}$, and $\bar{T}$.

Examples of GRASP with simulated annealing include Liu et al. [57] and de la Peña [19].

4.5. **Iterated local search.** Iterated local search (ILS) iteratively builds a sequence of solutions generated by the repeated application of local search and perturbation of the local optima found by local search [11]. Lourenço, Martin, and Stützle [58] point out that ILS has been rediscovered many times and is also known as iterated descent [9, 10], large step Markov chains [60], iterated Lin-Kernighan [48], and chained local optimization [59].

Figure 14 shows pseudo-code for ILS. Applying local search from the starting solution $s_0 \in S$, results in a local optimum t. The loop from line 2 to line 6 is repeated until some stopping criterion is satisfied. The current solution t is perturbed, becoming $\hat{t}$, which is used as the starting solution for local search. Perturbation needs to be done with some care. A perturbation that is too small may cause local search to lead back to the starting solution $\hat{t}$. On the other hand, a perturbation that is too large may cause the search to resemble random multi-start. The local optimum resulting from local search is $\hat{t}^*$. An acceptance rule determines if $\hat{t}^*$ is accepted to be the new current solution. Such acceptance criterion resemble simulated annealing, i.e. $\hat{t}^*$ is accepted if $\hat{t}^* < t$. Otherwise, it is accepted with some positive probability.

ILS can be hybridized with GRASP by replacing the standard local search. The GRASP construction produces the solution $s_0 \in S$ which is passed to the ILS procedure. Ribeiro and Urrutia [82] present a GRASP with ILS for the mirrored traveling tournament problem. Perturbation is achieved by randomly generating a solution in the *game rotation ejection chain* neighborhood. The acceptance rule makes use of a threshold parameter β. In the beginning of each GRASP iteration β is initialized to 0.001 and it is reinitialized to the same value each time the best

MIRRORED TRAVELING TOURNAMENT PROBLEM: Given an even number n of teams, each with a home venue, and the cost of travel between venues, a double-round robin tournament is organized in rounds, where in each round there are $n/2$ games and each team plays exactly once. The tournament is made up of $n-1$ rounds, where each team plays each other team exactly once, followed by another $n-1$ rounds with the same games as in the first $n-1$ rounds, but with reversed venues. Each team starts at its home venue before the tournament and finishes there at the end of the tournament. Find a tournament schedule having the least travel cost.

CAPACITATED MINIMUM SPANNING TREE PROBLEM: Given an undirected graph with demands associated with its vertices, edge costs, a fixed capacity, equal for all edges, and a source vertex, find a minimum-cost spanning tree rooted at the source vertex such that all the demand can be routed from the source to the demand vertices without violating any of the edge capacity constraints.

solution changes in the ILS. Solution $\hat{t}^*$ is accepted if its cost is less than $(1+\beta)$ times the cost of the current solution t. If the current solution does not change after a fixed number of iterations, then the value of β is doubled. The stopping criterion used was to allow at most 50 cost-deteriorating moves without improvement in the current best solution.

4.6. **Very-large scale neighborhood search.** The standard local search procedure described in Section 2 searches for an improving solution in the neighborhood of the current solution. In the first-improving variant of local search, solutions in the neighborhood are scanned until an improving solution is found. In the best-improving variant, the entire neighborhood is scanned to determine the best neighbor solution. To efficiently apply such local search requires that neighborhoods be small, i.e. that their growth be bounded by a polynomial function in the dimension of the problem. Many neighborhoods are of this type. The 1-flip neighborhood of a solution represented as a binary n-vector has size $O(n)$. The swap neighborhood of a solution represented as a permutation vector of size n has size $O(n^2)$.

It is easy to design neighborhoods that are much larger. Neighborhoods whose sizes grow exponentially as a function of problem size are called *very large scale neighborhoods*. Even neighborhoods with $O(n^4)$ solutions can be too large to search efficiently even for small values of n. These neighborhoods require efficient search techniques to be explored. Ahuja et al. [3] present a survey of methods called *very-large scale neighborhood* (VLSN) search. Three classes of VLSN methods are considered. The first are variable-depth methods where exponentially large neighborhoods are searched with heuristics. The second uses network flow techniques to identify improving neighborhood solutions. The third class consist of neighborhoods for NP-hard problems induced by restrictions of the problems that are solved in polynomial time.

We illustrate two examples of GRASP with VLSN search. In both instances VLSN search was used in a GRASP in place of the standard local search. Ahuja, Orlin, and Sharma [2] examine the capacitated minimum spanning tree problem.

CAPACITATED WAREHOUSE ROUTING PROBLEM: Consider a warehouse with a depot and n locations where from each a good is to be picked up by a vehicle. The vehicle is limited to traveling on a given road network in the warehouse and has a given capacity C so it may not be able to pick up all of the goods alone. Suppose we are provided K vehicles, a number sufficient to pickup all of the goods. Find K tours starting and ending at the depot, where in each tour no more than C goods are picked up, such that the sum of the K tour lengths in minimized and all goods are picked up.

They use multi-exchange neighborhood structures. Even though the sizes of these neighborhoods grow exponentially with problem size, the effort to find an improving solution in them grows insignificantly. Their search is based on the cyclic transfer neighborhood structure [89, 90] that transforms a cost-reducing exchange into a negative cost subset-disjoint cycle in an *improving* graph. A modified shortest path label-correcting algorithm is used to identify these cycles. Geng, Li, and Lim [35] consider the capacitated warehouse routing problem. They consider two neighborhoods, path-exchange and cyclic exchange and use an improvement graph to identify cost-improving cyclic- and path-exchanges.

5. Hybridization with path-relinking

Path-relinking was originally proposed by Glover [38] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [39, 41, 42]. Starting from one or more elite solutions, paths in the solution space graph leading toward other elite solutions are generated and explored in the search for better solutions. To generate paths, moves are selected to introduce attributes in the current solution that are present in the elite guiding solution. Path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions, by favoring these attributes in the selected moves.

The pseudo-code in Figure 15 illustrates the mixed path-relinking procedure applied to a pair of solutions x_s and x_t. Mixed path-relinking [79] interchanges the roles of starting and guiding solutions after each move. The procedure starts by computing the symmetric differences $\Delta(x_s, x_t)$ and $\Delta(x_t, x_s)$ between the two solutions, i.e. the set of moves needed to reach x_t from x_s and vice-versa. Two paths of solutions are generated, one starting at x_s and the other at x_t. These paths grow out and meet to form a single path between x_t from x_s. Local search is applied to the best solution x^* in this path and the local minimum is returned by the algorithm. Initially, x and y are set to x_s and x_t, respectively. At each step, the procedure examines all moves $m \in \Delta(x, y)$ from the current solution x to solutions that that contain an additional attribute of y and selects the one which results in the least cost solution, 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. A best move m^* is made, producing solution $x \oplus m^*$. If necessary, the best solution x^* is updated. The sets of available moves are updated and the roles of x and y are interchanged. The procedure terminates when x and y are in each other's local neighborhood, i.e. when $|\Delta(x, y)| = 1$.

Path-relinking is a major enhancement to the basic GRASP procedure, leading to significant improvements in solution time and quality. The hybridization of path-relinking and GRASP was first proposed by Laguna and Martí [54]. It was followed by several extensions, improvements, and successful applications [15, 75, 77, 81, 65, 5, 78, 29, 72]. A survey of GRASP with path-relinking is presented in Resende and Ribeiro [76]. Two basic strategies are used. In one, path-relinking is applied to all pairs of elite solutions, either periodically during the GRASP iterations or after all GRASP iterations have been performed as a post-optimization step. In the other, 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 seems to be more effective than simply using it only as a post-optimization step. In general, combining intensification with post-optimization results in the best strategy. In the context of intensification, path-relinking is applied to pairs (x, y) of solutions, where x is a locally optimal solution produced by each GRASP iteration after local search and y is one of a few elite solutions randomly chosen from a pool with a limited number `Max_Elite` of elite solutions found along the search. Uniform random selection is a simple strategy to implement. Since the symmetric difference is a measure of the length of the path explored during relinking, a strategy biased toward pool elements y with large symmetric difference with respect to x is usually better than one using uniform random selection [77].

The pool is originally empty. Since we wish to maintain a pool of good but diverse solutions, each locally optimal solution obtained by local search is considered as a candidate to be inserted into the pool if it is sufficiently different from every other solution currently in the pool. If the pool already has `Max_Elite` solutions and the

```
begin PathRelinking(x_s ∈ S, x_t ∈ S)
1   Compute symmetric differences Δ(x_s, x_t) and Δ(x_t, x_s);
2   f* ← min{f(x_s), f(x_t)};
3   x* ← argmin{f(x_s), f(x_t)};
4   x ← x_s; y ← x_t;
5   while |Δ(x, y)| > 1 do
6       m* ← argmin{f(x ⊕ m) : m ∈ Δ(x, y)};
7       x ← x ⊕ m*;
8       Update Δ(x, y) and Δ(y, x);
9       if f(x) < f* then
10          f* ← f(x);
11          x* ← x;
12      end-if;
13      t ← y; y ← x; x ← t;
14  end-while;
15  x* ← LocalSearch(x*);
16  return x*;
end
```

FIGURE 15. Mixed path-relinking procedure between solutions x_s and x_t.

candidate is better than the worst of them, then a simple strategy is to have the former replaces 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. If the pool is not full, the candidate is simply inserted.

Post-optimization is done on a series of pools. The initial pool P_0 is the pool P obtained at the end of the GRASP iterations. The value of the best solution of P_0 is assigned to f_0^* and the pool counter is initialized $k = 0$. At the k-th iteration, all pairs of elements in pool P_k are combined using path-relinking. Each result of path-relinking is tested for membership in pool P_{k+1} following the same criteria used during the GRASP iterations. If a new best solution is produced, i.e. $f_{k+1}^* < f_k^*$, then $k \leftarrow k + 1$ and a new iteration of post-optimization is done. Otherwise, post-optimization halts with $x^* \in \text{argmin}\{f(x) \mid x \in P_{k+1}\}$ as the result.

The pseudo-code in Figure 16 illustrates such a procedure. Each GRASP iteration has now three main steps. In the *construction phase* a greedy randomized construction procedure is used to build a feasible solution. In the *local search phase* the solution built in the first phase is progressively improved by a neighborhood search strategy, until a local minimum is found. In the *path-relinking phase* the path-relinking algorithm is applied to the solution obtained by local search and to a randomly selected solution from the pool. The best solution found along this trajectory is also considered as a candidate for insertion in the pool and the incumbent is updated. At the end of the GRASP iterations, a *post-optimization phase* combines the elite solutions in the pool in the search for better solutions.

6. Other hybridizations

In the previous sections of this chapter, we have reviewed some important hybridizations of GRASP. In this section, we briefly review other, more recent, hybridizations. These include the use of GRASP in Lagrangian heuristics, GRASP with data mining, and the use of GRASP to generate initial solutions for population-based heuristics, such as genetic algorithms and scatter search.

6.1. **GRASP in Lagrangian heuristics.** Pessôa, Resende, and Ribeiro [66] proposed a hybrid Lagrangian heuristic with GRASP and path-relinking for the k-set covering problem, a variant of the classical set covering problem. Pessôa, Resende, and Ribeiro [66] extend the Lagrangian relaxation scheme of Beasley [12] for set covering to solve k-set covering. Sub-gradient optimization is used to solve the so-called *Lagrangian Dual Problem* which starts with an initial set of Lagrange multipliers and iteratively generates further multipliers and corresponding lower bounds. Each step of the sub-gradient optimization produces a primal (usually infeasible) solution to the Lagrangian dual problem. This solution corresponds to a partial cover that, if necessary, is made feasible by a so-called *basic heuristic*. The objective function value of the feasible cover produced by the basic heuristic is given back to the Lagrangian heuristic as an upper bound and is used to adjust the step-size of the procedure that updates the multipliers.

Two basic heuristics were proposed by Pessôa, Resende, and Ribeiro [66]. The first is a greedy algorithm similar to the one proposed by Beasley [12] for the set covering problem. A local search procedure that removes redundant sets from the cover is applied after the greedy algorithm. The other basic heuristic is a GRASP

```
begin GRASP+PR()
1   P ← ∅;
2   f* ← ∞;
3   for i = 1, ..., i_max do
4       x ← GreedyRandomizedConstruction();
5       if x is not feasible then
6           x ← repair(x);
7       end-if
8       x ← LocalSearch(x);
9       if i ≥ 2 then
9           Choose, at random, pool solutions 𝒴 ⊆ P to relink with x;
10          for y ∈ 𝒴 do
11              x_p ← PathRelinking(x, y);
12              Update the elite set P with x_p;
13              if f(x_p) < f* then
14                  f* ← f(x_p);
15                  x* ← x_p;
16              end-if;
17          end-for;
18      end-if
19  end-for;
20  end-while;
21  P ← PostOptimize{P};
22  x* ← argmin{f(x), x ∈ P};
23  return x*;
end
```

FIGURE 16. A basic GRASP with path-relinking heuristic for minimization.

K-SET COVERING PROBLEM: Given n finite sets $P_1, P_2, \ldots, P_n$, let c_j be the cost associated with set P_j, for $j = 1, \ldots, n$. Denote sets $I = \bigcup_{j=1}^{n} P_j = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$. An element of set I is called an *object*. A subset $\hat{J} \subseteq J$ is called a *cover* if $\bigcup_{j \in \hat{J}} P_j = I$. The cost of cover $\hat{J}$ is $\sum_{j \in \hat{J}} c_j$. The *set covering problem* is to find a cover of minimum cost. The *k-set covering problem* is a generalization of the set covering problem where each object $i \in I$ must be covered by at least k sets.

with path-relinking which uses a randomized version of Beasley's greedy algorithm and the same local search. The path-relinking procedure is similar to the mixed scheme described in Section 5.

Computational results show that both Lagrangian heuristics find solutions that are much closer to the optimal than those found by the pure GRASP with path-relinking. Furthermore, they are much faster. Comparing the two basic heuristics shows that the one based on GRASP with path-relinking on average finds solutions within 1% of optimal while the one based on the greedy algorithm finds solutions between 1% and 2% of optimal. However, the Lagrangian heuristic based

QUADRATIC ASSIGNMENT PROBLEM: Given n facilities, represented by the set $F = \{w_1, \ldots, w_n\}$, and n locations, represented by the set $L = \{l_1, \ldots, l_n\}$, let $A^{n\times n} = (a_{ij})$ be a matrix where $a_{ij} \in \mathbb{R}^+$ is the required flow between facilities w_i and w_j and $B^{n\times n} = (b_{ij})$ be a matrix where $b_{ij} \in \mathbb{R}^+$ is the distance between locations l_i and l_j. Find a permutation $\pi : \{1 \ldots n\} \rightarrow \{1 \ldots n\}$, i.e. assign each facility to a different location, such that $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}$ is minimized.

on GRASP with path-relinking takes about twice as long as the one based on the greedy algorithm.

6.2. **GRASP with data mining.** Data mining techniques have recently begun to find their way into metaheuristics [51]. DM-GRASP, or GRASP with data mining, was introduced by Ribeiro, Plastino, and Martins [84]. This scheme uses data mining techniques to search for solution patterns that reoccur in high-quality solutions. In a initial phase, called the *elite set generation phase*, DM-GRASP builds a set of elite solutions as it would, for example, for path-relinking. Once a large enough pool of elite solutions is on hand, in the second phase of DM-GRASP data mining techniques are applied to mine the pool for frequently occurring solution patterns. This process is known in the data mining community as *frequent itemset mining*, or FIM [44], a subproblem of association rule mining.

In a third phase of DM-GRASP, called the *hybrid phase*, the mined patterns serve as a guide for GRASP construction. Instead of building the randomized greedy solution from scratch, the construction procedure starts from a solution pattern (a partial solution) that was mined in the second phase.

A survey of applications of DM-GRASP can be found in Santos, Martins, and Plastino [85].

6.3. **GRASP as initial solutions of population-based heuristics.** Population-based heuristics, such as genetic algorithms, scatter search, and evolutionary path-relinking require the generation of an initial population. Often, these initial populations are randomly generated. Another way to generate initial solutions is to use a GRASP.

Ahuja, Orlin, and Tiwari [4] used GRASP to generate the initial population of a genetic algorithm for the quadratic assignment problem. Alvarez, Gonzalez-Velarde, and Alba [6], Díaz and Fernández [22], and Contreras and Díaz [18] used GRASP to initialize the reference set of scatter search.

In Section 5 we discussed using path-relinking in a post-optimization phase. This process, call *Evolutionary Path-relinking* by Andrade and Resende [8] has been used in Resende and Werneck [77, 78] and Resende et al. [72].

7. Concluding remarks

In this chapter, we surveyed hybridizations of GRASP and other metaheuristics. The chapter considered a wide range of topics but by no means was it exhaustive. The field of hybrid metaheuristics is one that currently enjoys much activity. Many important contributions to this field have come about recently and the field promises important developments in the near future. Hybridizations involving GRASP will hopefully be part of these developments.

References

[1] S. Abdinnour-Helm and S.W. Hadley. Tabu search based heuristics for multi-floor facility layout. *International Journal of Production Research*, 38:365–383, 2000.

[2] R. K. Ahuja, J. B. Orlin, and D. Sharma. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming, Ser. A*, 91:71–97, 2001.

[3] R.K. Ahuja, Ö. Ergun, J.B. Orlin, and A.P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Appl. Math.*, 123:75–102, 2002.

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

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

[6] A.M. Alvarez, J.L. Gonzalez-Velarde, and K. De Alba. GRASP embedded scatter search for the multicommodity capacitated network design problem. *J. of Heuristics*, 11:233–257, 205.

[7] D.V. Andrade and M.G.C. Resende. A GRASP for PBX telephone migration scheduling. In *Proceedings of The Eighth INFORMS Telecommunications Conference*, 2006.

[8] D.V. Andrade and M.G.C. Resende. GRASP with evolutionary path-relinking. In *Proc. of Seventh Metaheuristics International Conference (MIC 2007)*, July 2007.

[9] E.B. Baum. Iterated descent: A better algorithm for local search in combinatorial optimization problems. Technical report, California Institute of Technology, 1986.

[10] E.B. Baum. Towards practical 'neural' computation for combinatorial optimization problems. In *AIP Conference Proceedings 151 on Neural Networks for Computing*, pages 53–58, Woodbury, NY, USA, 1987. American Institute of Physics Inc.

[11] J. Baxter. Local optima avoidance in depot location. *Journal of the Operational Research Society*, 32:815–819, 1981.

[12] J.E. Beasley. Lagrangean relaxation. In C.R. Reeves, editor, *Modern heuristic techniques for combinatorial problems*, pages 243–303. Blackwell Scientific Publications, Oxford, U.K. and Boston MA, 1993.

[13] J.D. Beltrán, J.E. Calderón, R.J. Cabrera, and J.A.M. Pérez and J.M. Moreno-Vega. GRASP/VNS hybrid for the strip packing problem. In *Proceedings of Hybrid Metaheuristics (HM2004)*, pages 79–90, 2004.

[14] J.L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 271–278, Portland, 1996.

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

[16] I. Charon and O. Hudry. The noising method: A new method for combinatorial optimization. *Operations Research Letters*, 14:133–137, 1993.

[17] I. Charon and O. Hudry. The noising methods: A survey. In C.C. Ribeiro and C.C. Ribeiro, editors, *Essays and Surveys in Metaheuristics*, pages 245–261. Kluwer Academic Publishers, 2002.

[18] I.A. Contreras and J.A. Díaz. Scatter search for the single source capacitated facility location problem. *Annals of Operations Research*, 157:73–89, 2008.

[19] M.G.B. de la Peña. 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`.

[20] M.C. de Souza, C. Duhamel, and C.C. Ribeiro. A GRASP heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy. In M.G.C. Resende and J.P. de Sousa, editors, *Metaheuristics: Computer decision-making*, pages 627–658. Kluwer Academic Publishers, 2003.

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

[22] J.A. Díaz and E. Fernández. Hybrid scatter search and path relinking for the capacitated p-median problem. *European J. of Operational Research*, 169:570–585, 2006.

[23] L. Drummond, L.S. Vianna, M.B. Silva, and L.S. Ochi. Distributed parallel metaheuristics based on GRASP and VNS for Solving the traveling purchaser problem. In *Proceedings of*

the Ninth International Conference on Parallel and Distributed Systems (ICPADS02), pages 257–266, 2002.

[24] A. Duarte, C.C. Ribeiro, and S. Urrutia. A hybrid ILS heuristic to the referee assignment problem with an embedded MIP strategy. *Lecture Notes in Computer Science*, 4771:82–95, 2007.

[25] A.R. Duarte, C.C. Ribeiro, S. Urrutia, and E.H. Haeusler. Referee assignment in sports leagues. *Lecture Notes in Computer Science*, 3867:158–173, 2007.

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

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

[28] T.A. Feo, M.G.C. Resende, and S.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878, 1994.

[29] P. Festa, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. GRASP with path-relinking for the weighted MAXSAT problem. *ACM J. of Experimental Algorithmics*, 11, 2006. article 2.4: 1-16.

[30] P. Festa, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Randomized heuristics for the MAX-CUT problem. *Optimization Methods and Software*, 7:1033–1058, 2002.

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

[32] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP. Technical report, AT&T Labs Research, Florham Park, NJ 07932, 2008. To appear in *International Transactions in Operational Research*.

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

[34] M. Gendreau. An introduction to tabu search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 37–54. Kluwer Academic Publishers, 2003.

[35] Y. Geng, Y. Li, and A. Lim. A very large-scale neighborhood search approach to capacitated warehouse routing problem. In *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 05)*, pages 8 pp.–, 2005.

[36] F. Glover. Tabu search – Part I. *ORSA J. on Computing*, 1:190–206, 1989.

[37] F. Glover. Tabu search – Part II. *ORSA J. on Computing*, 2:4–32, 1990.

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

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

[40] F. Glover and G. Kochenberger, editors. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

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

[42] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

[43] M.X. Goemans and D.P. Williamson. The primal dual method for approximation algorithms and its application to network design problems. In D. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 144–191. PWS Publishing Co., 1996.

[44] B. Goethals and M.J. Zaki, editors. *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[45] P. Hansen and N. Mladenović. An introduction to variable neighbourhood search. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Metaheuristics: Advances and Trends in Local Search Procedures for Optimization*, pages 433–458. Kluwer, 1999.

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

[47] J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.

[48] D.S. Johnson. Local optimization and the traveling salesman problem. In *Proceedings of the 17th Colloquium on Automata*, volume 443 of *LNCS*, pages 446–461. Springer-Verlag, 1990.

[49] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation. Part I, graph partitioning. *Operations Research*, 37:865–892, 1989.

[50] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation. Part II, graph coloring and number partitioning. *Operations Research*, 39:378–406, 1991.

[51] L. Jourdan, C. Dhaenens, and E.-G. Talbi. Using datamining techniques to help metaheuristics: A short survey. In F. Almeida, M.J.B. Aguilera, C. Blum, J.M.M. Vega, M.P. Pérez, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics, Third InternationalWorkshop, HM 2006*, volume 4030 of *Lecture Notes in Computer Science*. Springer, 2006.

[52] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

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

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

[55] Z. Li, S. Guo, F. Wang, and A. Lim. Improved GRASP with tabu search for vehicle routing with both time window and limited number of vehicles. In B. Orchard, C. Yang, and M. Ali, editors, *Innovations in Applied Artificial Intelligence – Proceedings of the 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2004)*, volume 3029 of *Lecture Notes in Computer Science*, pages 552–561. Springer-Verlag, 2004.

[56] A. Lim and F. Wang. A smoothed dynamic tabu search embedded GRASP for m-VRPTW. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*, pages 704–708, 2004.

[57] X. Liu, P.M. Pardalos, S. Rajasekaran, and M.G.C. Resende. A GRASP for frequency assignment in mobile radio networks. In B.R. Badrinath, F. Hsu, P.M. Pardalos, and S. Rajasejaran, editors, *Mobile Networks and Computing*, volume 52 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 195–201. American Mathematical Society, 2000.

[58] H.R. Lourenço, O.C. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 321–353. Kluwer Academic Publishers, 2003.

[59] O. Martin and S.W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.

[60] O. Martin, S.W. Otto, and E.W. Felten. Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.

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

[62] A. Moura and J.F. Oliveira. A GRASP approach to the container-loading problem. *IEEE Intelligent Systems*, 20:50–57, 2005.

[63] M.C.V. Nascimento, M.G.C. Resende, and F.M.B. Toledo. GRASP with path-relinking for the multi-plant capacitated plot sizing problem. *European J. of Operational Research*, 2008. To appear.

[64] L.S. Ochi, M.B. Silva, and L. Drummond. GRASP and VNS for solving traveling purchaser problem. In *Proceedings of The Fourth Metaheuristics International Conference (MIC2001)*, pages 489–494, 2001.

[65] C.A. Oliveira, P.M. Pardalos, and M.G.C. Resende. GRASP with path-relinking for the quadratic assignment problem. In C.C. Ribeiro and S.L. Martins, editors, *Proceedings of III Workshop on Efficient and Experimental Algorithms (WEA2004)*, volume 3059, pages 356–368. Springer, 2004.

[66] L.S. Pessôa, M.G.C. Resende, and C.C. Ribeiro. A hybrid Lagrangian heuristic with GRASP and path-relinking for k-set cover. Technical report, AT&T Labs Research, Florham Park, NJ 07932 USA, 2008.

[67] L.S. Pitsoulis and M.G.C. Resende. Greedy randomized adaptive search procedures. In P.M. Pardalos and M.G.C. Resende, editors, *Handbook of Applied Optimization*, pages 168–183. Oxford University Press, 2002.

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

[69] G.G. Pu, Z. Chong, Z.Y. Qiu, Z.Q. Lin, and J.F. He. A hybrid heuristic algorithm for HW-SW partitioning within timed automata. In *Proceedings of Knowledge-based Intelligent Information and Engineering Systems*, volume 4251 of *Lecture Notes in Artificial Intelligence*, pages 459–466. Springer-Verlag, 2006.

[70] G.R. Raidl. A unified view on hybrid metaheuristics. In F. Almeida, M.J.B. Aguilera, C. Blum, J M.M. Vega, M.P. Pérez, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 4030 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2006.

[71] M.G.C. Resende, T.A. Feo, and S.H. Smith. Algorithm 787: Fortran subroutines for approximate solution of maximum independent set problems using GRASP. *ACM Trans. Math. Software*, 24:386–394, 1998.

[72] M.G.C. Resende, R. Martí, M. Gallego, and A. Duarte. GRASP and path relinking for the max-min diversity problem. *Computers and Operations Research*, 2008. To appear.

[73] M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Approximate solution of weighted MAX-SAT problems using GRASP. In J. Gu and P.M. Pardalos, editors, *Satisfiability Problems*, volume 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 393–405. American Mathematical Society, 1997.

[74] M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Fortran subroutines for computing approximate solutions of MAX-SAT problems using GRASP. *Discrete Applied Mathematics*, 100:95–113, 2000.

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

[76] M.G.C. Resende and C.C. Ribeiro. GRASP with path-relinking: Recent advances and applications. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: Progress as Real Problem Solvers*, pages 29–63. Springer, 2005.

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

[78] M.G.C. Resende and R.F. Werneck. A hybrid multistart heuristic for the uncapacitated facility location problem. *European J. of Operational Research*, 174:54–68, 2006.

[79] C.C. Ribeiro and I. Rosseti. A parallel GRASP for the 2-path network design problem. *Lecture Notes in Computer Science*, 2004:922–926, 2002.

[80] C.C. Ribeiro and M.C. Souza. Variable neighborhood search for the degree constrained minimum spanning tree problem. *Discrete Applied Mathematics*, 118:43–54, 2002.

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

[82] C.C. Ribeiro and S. Urrutia. Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, 127:775–787, 2007.

[83] C.C. Ribeiro and D.S. Vianna. A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. *International Transactions in Operational Research*, 12:325–338, 2005.

[84] M.H. Ribeiro, A. Plastino, and S.L. Martins. Hybridization of GRASP metaheuristic with data mining techniques. *J. of Mathematical Modelling and Algorithms*, 5:23–41, 2006.

[85] L.F. Santos, S.L. Martins, and A. Plastino. Applications of the DM-GRASP heuristic: A survey. *International Transactions in Operational Research*, 2008. To appear.

[86] D. Serra and R. Colomé. Consumer choice and optimal location models: Formulations and heuristics. *Papers in Regional Science*, 80:439–464, 2001.

[87] M.J.F. Souza, N. Maculan, and L.S. Ochi. A GRASP-tabu search algorithm to solve a school timetabling problem. In *Proceedings of The Fourth Metaheuristics International Conference (MIC2001)*, pages 53–58, 2001.

[88] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.
[89] P.M. Thompson and J.B. Orlin. The theory of cyclic transfers. Technical Report OR 200-89, Operations Research Center, MIT, Cambridge, MA, 1989.
[90] P.M. Thompson and H.N. Psaraftis. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research*, 41:935–946, 1993.

(Mauricio G. C. Resende) Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.
E-mail address: mgcr@research.att.com