GRASP: GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURES

MAURICIO G. C. RESENDE AND RICARDO M. A. SILVA

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 review the basic building blocks of GRASP. We cover solution construction schemes, local search methods, and the use of path-relinking as a memory mechanism in GRASP.

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 $f : 2^E \to \mathbb{R}$, all three defined for each specific problem. In this chapter, we consider the minimization version of the problem, where we seek an optimal solution $S^* \in F$ such that $f(S^*) \leq f(S),\ \forall S \in F$. Combinatorial optimization finds applications in many settings, including routing, scheduling, inventory and production planning, and facility location.

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. Among these, we find genetic algorithms, simulated annealing, tabu search, variable neighborhood search, scatter search, path-relinking, iterated local search, ant colony optimization, swarm optimization, and greedy randomized adaptive search procedures (GRASP).

In this chapter, we review the basic building blocks of GRASP, including solution construction schemes, local search methods, and use of path-relinking as a memory mechanism in GRASP. The chapter is organized as follows. In Section 1, we introduce a basic local search scheme. In Section 2 we examine the relationship between the metaheuristic GRASP and local search. Section 3 covers construction schemes while In Section 4, we introduce a memory mechanism in GRASP through path-relinking. Concluding remarks are made in Section 5.

1. Local search

Local search is a fundamental operator in GRASP heuristics. It is fully specified by a feasible starting solution $s_0 \in F$, an objective function $f(\cdot)$, for which we seek

Date: February 17, 2009. Revised September 9, 2009.

Key words and phrases. GRASP, metaheuristics, hybrid heuristics, path-relinking, local search.

AT&T Labs Research Technical Report. R.M.A. Silva was partially funded by the Brazilian Council for the Development of Science and Technology, CNPq.

$t \leftarrow s_0$;
while *there exists* $s \in N(t)$ *such that* $f(s) < f(t)$ **do**
  $t \leftarrow s$;
end
return t;

Algorithm 1: Basic local search algorithm.

a local optimum, and a local neighborhood structure $N(\cdot)$ that restricts feasible moves in the search space. Local search in GRASP is applied from many distinct feasible starting solutions. Each application of local search in a GRASP heuristic results in a locally optimal solution, the best of which is output as the GRASP solution.

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.

The basic local search schemes 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.

The Algorithm 1 shows pseudo-code for a basic local search algorithm. Local search 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* all neighboring solutions are evaluated and the local search moves to one of the best found. A compromise solution is used in *sampled local search* [48] where the neighborhood is sampled and the local search moves to the best of the sampled neighbors. One can think of this part of the search as *intensification*, since we concentrate on a small portion of the solution space.

Several approaches have been proposed to extend the basic local search scheme described above. These include methods such as variable neighborhood descent [5, 36, 47, 64, 65, 67], variable neighborhood search [9, 11, 17, 23, 36, 51], short-term memory tabu search [1, 15, 16, 29, 30, 40, 42, 43, 49, 55, 68, 69], simulated annealing [14, 39, 44], iterated local search [6, 7, 8, 38, 45, 46, 66], and very-large scale neighborhood search [2, 3, 28], that explore beyond the current solution's

```
x* ← ∞;
while stopping criterion not satisfied do
    x ← RandomizedGreedy(·);
    if x is not feasible then
        x ← repair(x);
    end
    x ← LocalSearch(x);
    if f(x) < f(x*) then
        x* ← x;
    end
end
return x*;
```

Algorithm 2: A basic GRASP in pseudo-code.

neighborhood by allowing cost-increasing moves, by exploring multiple neighborhoods, or by exploring very large neighborhoods.

2. Greedy Randomized Adaptive Search Procedures

An effective search method 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 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. The Algorithm 2 shows pseudo-code for a generic GRASP heuristic. 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 requires that a repair procedure is applied to restore feasibility. Examples of GRASP implementations where repair procedures are needed to restore feasibility of the constructed solution can be found in Duarte, Ribeiro, and Urrutia [18], Duarte et al. [19], Nascimento, Resende, and Toledo [50], and Mateus, Resende, and Silva [48].

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 [20]. See Feo and Resende [21], Pitsoulis and Resende [53], Resende and Ribeiro [58], and Resende [56] for surveys of GRASP and Festa and Resende [24, 25, 26] for annotated bibliographies.

```
S ← ∅; C ← E;
while |C| > 0 do
    For all c ∈ C compute greedy function value g(c);
    Define RCL(C) ← {c ∈ C | g(c) has a low value};
    Select at random c* ∈ RCL(C);
    Add c* to partial solution: S ← S ∪ {c*};
    Let C be the set of ground set elements that can
        be added to S;
end
return S;
```

Algorithm 3: Greedy randomized construction: Randomizing the greedy algorithm – semi-greedy algorithm.

3. 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 eight ways to do this. Solutions are built by adding one ground set element at a time to a partially constructed solution.

3.1. **Semi-greedy construction.** In the first construction scheme, called *semi-greedy algorithm* by Hart and Shogan [37], 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. Algorithm 3 shows pseudo-code for this construction procedure.

3.2. **Sample greedy construction.** In the second construction scheme, called *sample greedy* by Resende and Werneck [61], 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

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

Algorithm 4: GRASP greedy randomized construction: sample greedy.

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. Algorithm 4 shows pseudo-code for this construction procedure. Sample greedy 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.

3.3. **Random plus greedy construction.** 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. The third construction scheme, *random plus greedy*, was introduced in Resende and Werneck [61]. As is the case with the *sample greedy* construction scheme, the random plus greedy scheme also addresses 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.

3.4. **Proportional greedy construction.** The fourth construction scheme is the *proportional greedy* construction scheme. It was also introduced in Resende and Werneck [61]. 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 $g(c') - \min\{g(c)|c \in C\}$.

3.5. **Reactive GRASP construction.** The fifth construction scheme is *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 [54] 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 as a guide for the selection process.

Prais and Ribeiro [54] define $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ to be the set of possible values for α. The 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.

3.6. **Long-term memory in construction.** The sixth construction scheme introduces long-term memory structures. Though reactive GRASP uses long-term memory (information gathered in previous iterations) to adjust the balance of greediness and randomness, Fleurent and Glover [27] 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 [29, 30] 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 [27] 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.

3.7. **Biased sampling construction.** The seventh construction scheme was introduced by Bresina [10] as *heuristic-biased stochastic sampling*. It is another way to depart from the uniform selection of candidate elements in the construction of a greedy randomized solution. 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 [10], 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 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 RCL $= C$.

3.8. **Construction with cost perturbation.** In the eighth construction scheme, called *construction with cost perturbation*, random perturbations are introduced in the original problem as a way of achieving randomness. The idea of introducing noise into the original costs to randomize a solution construction procedure is similar to that in the so-called *noising method* of Charon and Hudry [12, 13].

Construction with cost perturbation 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 [65] showed that this is the case for the shortest-path heuristic of Takahashi and Matsuyama [70], 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 [11] for the prize-collecting Steiner tree problem, which makes use of the primal-dual approximation algorithm of Goemans and Williamson [35] to build initial solutions using perturbed costs.

4. Adding memory to GRASP through path-relinking

Path-relinking was originally proposed by Glover [31] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [32, 33, 34]. 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.

```
Compute symmetric differences Δ(x_s, x_t) and Δ(x_t, x_s);
f* ← min{f(x_s), f(x_t)};
x* ← argmin{f(x_s), f(x_t)};
x ← x_s; y ← x_t;
while |Δ(x, y)| > 1 do
    m* ← argmin{f(x ⊕ m) : m ∈ Δ(x, y)};
    x ← x ⊕ m*;
    Update Δ(x, y) and Δ(y, x);
    if f(x) < f* then
        f* ← f(x);
        x* ← x;
    end
    t ← y; y ← x; x ← t;
end
x* ← LocalSearch(x*);
return x*;
```

Algorithm 5: Mixed path-relinking procedure between solutions x_s and x_t.

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 Algorithm 5 illustrates the mixed path-relinking procedure applied to a pair of solutions x_s and x_t. Mixed path-relinking [63] 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 use of path-relinking in GRASP was first proposed by Laguna and Martí [41]. It was followed by several extensions, improvements, and successful applications [4, 11, 22, 52, 57, 58, 61, 62, 65]. A survey of GRASP with path-relinking is presented in Resende and Ribeiro [59]. 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

```
P ← ∅;
f* ← ∞;
for i = 1, ..., i_max do
    x ← GreedyRandomizedConstruction();
    if x is not feasible then
        x ← repair(x);
    end
    x ← LocalSearch(x);
    if i ≥ 2 then
        Randomly choose pool solutions 𝒴 ⊆ P to relink with x;
        for y ∈ 𝒴 do
            x_p ← PathRelinking(x, y);
            Update the elite set P with x_p;
            if f(x_p) < f* then
                f* ← f(x_p);
                x* ← x_p;
            end
        end
    end
end
P ← PostOptimize{P};
x* ← argmin{f(x), x ∈ P};
return x*;
```

Algorithm 6: A basic GRASP with path-relinking heuristic.

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 [61].

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 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 Algorithm 6 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.

For some combinatorial optimization problems, a move m^* guided by the target solution is not guaranteed to result in a feasible solution $x \oplus m^*$ and the above described scheme fails. Mateus, Resende, and Silva [48] proposed a new variant of GRASP with path-relinking suitable for such problems.

5. Concluding remarks

This chapter considered basic building blocks needed to engineer heuristics based on the guiding principles of GRASP. These include randomized solution construction schemes, local search procedures, and the introduction of memory structures by way of path-relinking. One important topic that was left out of this chapter is that of parallel GRASP. The interested reader is directed to the survey of Resende and Ribeiro [60].

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.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.

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

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

[7] 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.

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

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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`.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP, Part I: Algorithms. *International Transactions in Operational Research*, 16:1–24, 2009.

[26] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP, Part II: Applications. *International Transactions in Operational Research*, 16:131–172, 2009.

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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.

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

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

[35] 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.

[36] 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.

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

[38] 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.

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

[40] 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.

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

[42] 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.

[43] 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.

[44] 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.

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

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

[47] 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.

[48] G.R. Mateus, M.G.C. Resende, and R.M.A. Silva. GRASP with path-relinking for the generalized quadratic assignment problem. Technical report, AT&T Labs Research, Shannon Laboratory, Florham Park, New Jersey, January 2009.

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

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] M.G.C. Resende. Metaheuristic hybridization with Greedy Randomized Adaptive Search Procedures. In Zhi-Long Chen and S. Raghavan, editors, *TutORials in Operations Research*, pages 295–319. INFORMS, 2008.

[57] 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.

[58] 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.

[59] 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.

[60] M.G.C. Resende and C.C. Ribeiro. Parallel Greedy Randomized Adaptive Search Procedures. In E. Alba, editor, *Parallel Metaheuristics: A new class of algorithms*, pages 315–346. John Wiley and Sons, 2005.

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

[62] 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.

[63] 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.

[64] 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.

[65] 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.

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

[67] 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.

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

[69] 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.

[70] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.

(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

(Ricardo M. A. Silva) Computational Intelligence and Optimization Group, Department of Computer Science, Federal University of Lavras, C.P. 3037, 37200-000 Lavras, MG Brazil.
E-mail address: rmas@dcc.ufla.br