Greedy randomized adaptive search procedures: Advances, hybridizations, and applications

Mauricio G.C. Resende and Celso C. Ribeiro

Abstract GRASP is a multi-start metaheuristic for combinatorial optimization problems, in which each iteration consists basically of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result. In this chapter, we first describe the basic components of GRASP. Successful implementation techniques are discussed and illustrated by numerical results obtained for different applications. Enhanced or alternative solution construction mechanisms and techniques to speed up the search are also described: alternative randomized greedy construction schemes, Reactive GRASP, cost perturbations, bias functions, memory and learning, local search on partially constructed solutions, hashing, and filtering. We also discuss implementation strategies of memory-based intensification and post-optimization techniques using path-relinking. Hybridizations with other metaheuristics, parallelization strategies, and applications are also reviewed.

1 Introduction

We consider in this chapter a combinatorial optimization problem, defined by a finite ground set $E = \{1, \ldots, n\}$, a set of feasible solutions $F \subseteq 2^E$, and an objective function $f : 2^E \to \mathbb{R}$. In its minimization version, we search 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

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

Celso C. Ribeiro
Universidade Federal Fluminense, Niterói, RJ 22410-240 Brazil, e-mail: `celso@ic.uff.br`

connecting the cities to be visited, $f(S)$ is the sum of the costs of all edges in S, and F is formed by all edge subsets that determine a Hamiltonian cycle.

GRASP (Greedy Randomized Adaptive Search Procedure) [68, 69] is a multi-start or iterative metaheuristic, in which each iteration consists of two phases: construction and local search. The construction phase builds a solution. If this solution is not feasible, then it is necessary to apply a repair procedure to achieve feasibility. Once a feasible solution is obtained, its neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result. Extensive literature surveys are presented in [78, 79, 80, 156, 157, 160]. The pseudo-code in Figure 1 illustrates the main blocks of a GRASP procedure for minimization, in which `Max_Iterations` iterations are performed and `Seed` is used as the initial seed for the pseudo-random number generator.

```
procedure GRASP(Max_Iterations,Seed)
    Read_Input();
    for $k = 1,\ldots,$Max_Iterations do
        Solution ← Greedy_Randomized_Construction(Seed);
        if Solution is not feasible then
            Solution ← Repair(Solution);
        end;
        Solution ← Local_Search(Solution);
        Update_Solution(Solution,Best_Solution);
    end;
    return Best_Solution;
end GRASP.
```

Fig. 1 Pseudo-code of the GRASP metaheuristic.

Figure 2 illustrates the construction phase with its pseudo-code. At each iteration of this phase, let the set of candidate elements be formed by all elements of the ground set E that can be incorporated into the partial solution being built, without impeding the construction of a feasible solution with the remaining ground set elements. The selection of the next element for incorporation is determined by the evaluation of all candidate elements according to a greedy evaluation function. This greedy function usually represents the incremental increase in the cost function due to the incorporation of this element into the solution under construction. The evaluation of the elements by this function leads to the creation of a restricted candidate list (RCL) formed by the best elements, i.e. those whose incorporation to the current partial solution results in the smallest incremental costs (this is the greedy aspect of the algorithm). The element to be incorporated into the partial solution is randomly selected from those in the RCL (this is the probabilistic aspect of the heuristic). Once the selected element is incorporated into the partial solution, the candidate list is updated and the incremental costs are reevaluated (this is the adaptive aspect of the heuristic). The above steps are repeated while there exists at least one candidate element. This strategy is similar to the semi-greedy heuristic proposed by Hart and Shogan [97], which is also a multi-start approach based on greedy randomized constructions, but without local search.

```
procedure Greedy_Randomized_Construction(Seed)
1   Solution ← $\emptyset$;
2   Initialize the set of candidate elements;
3   Evaluate the incremental costs of the candidate elements;
4   while there exists at least one candidate element do
5       Build the restricted candidate list (RCL);
6       Select an element $s$ from the RCL at random;
7       Solution ← Solution $\cup \{s\}$;
8       Update the set of candidate elements;
9       Reevaluate the incremental costs;
10  end;
11  return Solution;
end Greedy_Randomized_Construction.
```

Fig. 2 Pseudo-code of the construction phase.

Not always is a randomized greedy construction procedure able to produce a feasible solution. In case this occurs, it may be necessary to apply a repair procedure to achieve feasibility. Examples of repair procedures can be found in [60, 61, 129].

The solutions generated by a greedy randomized construction are not necessarily optimal, even with respect to simple neighborhoods. The local search phase usually improves the constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in its neighborhood. It terminates when no better solution is found in the neighborhood. The pseudo-code of a basic local search algorithm starting from the solution `Solution` constructed in the first phase (and possibly made feasible by the repair heuristic) and using a neighborhood N is given in Figure 3.

```
procedure Local_Search(Solution)
1   while Solution is not locally optimal do
2       Find $s' \in N(\texttt{Solution})$ with $f(s') < f(\texttt{Solution})$;
3       Solution ← $s'$;
4   end;
5   return Solution;
end Local_Search.
```

Fig. 3 Pseudo-code of the local search phase.

The speed and the effectiveness of a local search procedure depend on several aspects, such as the neighborhood structure, the neighborhood search technique, the strategy used for the evaluation of the cost function value at the neighbors, and the starting solution itself. The construction phase plays a very important role with respect to this last aspect, building high-quality starting solutions for the local search. Simple neighborhoods are usually used. The neighborhood search may be implemented using either a *best-improving* or a *first-improving* strategy. In the case of the best-improving strategy, all neighbors are investigated and the current solution is replaced by the best neighbor. In the case of a first-improving strategy, the current solution moves to the first neighbor whose cost function value is smaller than that of the current solution. In practice, we observed on many applications that quite often both strategies lead to the same final solution, but in smaller computation times when the first-improving strategy is used. We also observed that premature convergence to a bad local minimum is more likely to occur with a best-improving strategy.

2 Construction of the restricted candidate list

An especially appealing characteristic of GRASP is the ease with which it can be implemented. Few parameters need to be set and tuned. Therefore, development can focus on implementing appropriate data structures for efficient construction and local search algorithms. GRASP has two main parameters: one related to the stopping criterion and the other to the quality of the elements in the restricted candidate list.

The stopping criterion used in the pseudo-code described in Figure 1 is determined by the number `Max_Iterations` of iterations. Although the probability of finding a new solution improving the incumbent (current best solution) decreases with the number of iterations, the quality of the incumbent may only improve with the latter. Since the computation time does not vary much from iteration to iteration, the total computation time is predictable and increases linearly with the number of iterations. Consequently, the larger the number of iterations, the larger will be the computation time and the better will be the solution found.

For the construction of the RCL used in the first phase we consider, without loss of generality, a minimization problem as the one formulated in Section 1. We denote by $c(e)$ the incremental cost associated with the incorporation of element $e \in E$ into the solution under construction. At any GRASP iteration, let c^{min} and c^{max} be, respectively, the smallest and the largest incremental costs.

The restricted candidate list RCL is made up of the elements $e \in E$ with the best (i.e., the smallest) incremental costs $c(e)$. This list can be limited either by the number of elements (cardinality-based) or by their quality (value-based). In the first case, it is made up of the p elements with the best incremental costs, where p is a parameter. In this chapter, the RCL is associated with a threshold parameter $\alpha \in [0,1]$. The restricted candidate list is formed by all elements $e \in E$ which can be inserted into the partial solution under construction without destroying feasibility and whose quality is superior to the threshold value, i.e., $c(e) \in [c^{min}, c^{min} + \alpha(c^{max} - c^{min})]$. The case $\alpha = 0$ corresponds to a pure greedy algorithm, while $\alpha = 1$ is equivalent to a random construction. The pseudo code in Figure 4 is a refinement of the greedy randomized construction pseudo-code shown in Figure 2. It shows that the parameter α controls the amounts of greediness and randomness in the algorithm.

```
procedure Greedy_Randomized_Construction($\alpha$, Seed)
1   Solution $\leftarrow \emptyset$;
2   Initialize the candidate set: $C \leftarrow E$;
3   Evaluate the incremental cost $c(e)$ for all $e \in C$;
4   while $C \neq \emptyset$ do
5       $c^{min} \leftarrow \min\{c(e) \mid e \in C\}$;
6       $c^{max} \leftarrow \max\{c(e) \mid e \in C\}$;
7       RCL $\leftarrow \{e \in C \mid c(e) \leq c^{min} + \alpha(c^{max} - c^{min})\}$;
8       Select an element $s$ from the RCL at random;
9       Solution $\leftarrow$ Solution $\cup \{s\}$;
10      Update the candidate set $C$;
11      Reevaluate the incremental cost $c(e)$ for all $e \in C$;
12  end;
13  return Solution;
end Greedy_Randomized_Construction.
```

Fig. 4 Refined pseudo-code of the construction phase.

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 average solution value should be worse than that of the greedy solution. However, the value of the best solution found outperforms the average value and very often is optimal. It is unlikely that GRASP will find an optimal solution if the average solution value is high, even if there is a large variance in the overall solution values. 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 low. What often leads to good solutions are relatively low average solution values in the presence of a relatively large variance, such as is the case for $\alpha = 0.2$.

Another interesting observation is that the distances between the solutions obtained at each iteration and the best solution found increase as the construction phase moves from more greedy to more random. This causes the average time taken by the local search to increase. Very often, many GRASP solutions may be generated in the same amount of time required for the local search procedure to converge from a single random start. In these cases, the time saved by starting the local search from good initial solutions can be used to improve solution quality by performing more GRASP iterations.

These results are illustrated in Table 1 and Figure 5, for an instance of the MAXSAT problem where 1000 iterations were run. For each value of α ranging from 0 (purely random construction for maximization problems) to 1 (purely greedy construction for maximization problems), we give in Table 1 the average Hamming distance between each solution built during the construction phase and the corresponding local optimum obtained after local search, the average number of moves from the first to the latter, the local search time in seconds, and the total processing time in seconds. Figure 5 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.

Table 1 Average number of moves and local search time as a function of the RCL parameter α for a maximization problem.

α	avg. distance	avg. moves	local search time (s)	total time (s)
0.0	12.487	12.373	18.083	23.378
0.1	10.787	10.709	15.842	20.801
0.2	10.242	10.166	15.127	19.830
0.3	9.777	9.721	14.511	18.806
0.4	9.003	8.957	13.489	17.139
0.5	8.241	8.189	12.494	15.375
0.6	7.389	7.341	11.338	13.482
0.7	6.452	6.436	10.098	11.720
0.8	5.667	5.643	9.094	10.441
0.9	4.697	4.691	7.753	8.941
1.0	2.733	2.733	5.118	6.235

Prais and Ribeiro [142] have shown that using a single fixed value for the value of the RCL parameter α very often hinders finding a high-quality solution, which 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 self-tuned and its value is periodically modified according with the quality of the solutions obtained along the search. In particular, computational experiments on the problem of traffic assignment in communication satellites [143] have shown that Reactive GRASP found better solutions than the basic algorithm for many test instances. These results motivated the study of the behavior of GRASP for different strategies for the variation of the value of the RCL parameter α:

(R) α self tuned according with the Reactive GRASP procedure;
(E) α randomly chosen from a uniform discrete probability distribution;

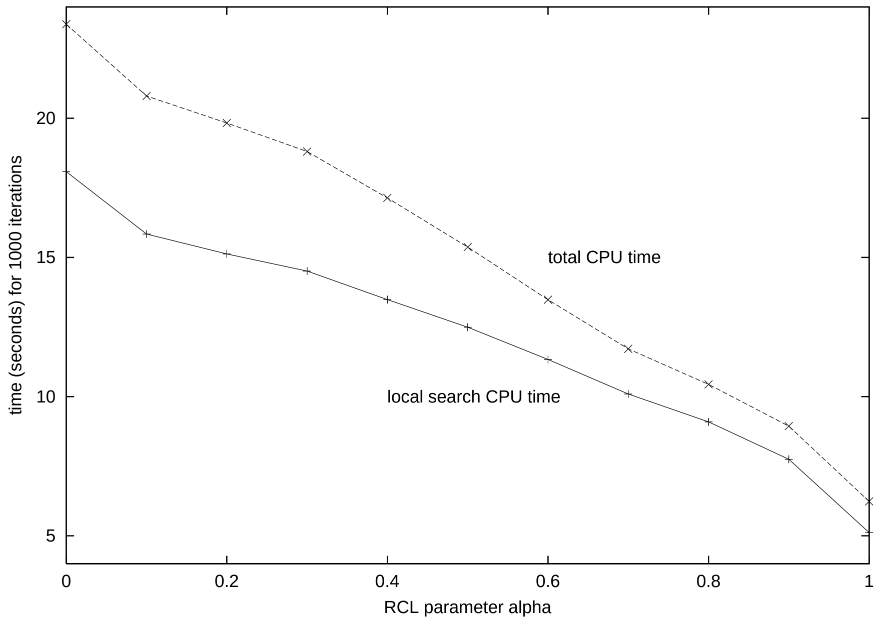


Fig. 5 Total CPU time and local search CPU time as a function of the RCL parameter α for a maximization problem (1000 repetitions for each value of α).

(H) α randomly chosen from a decreasing non-uniform discrete probability distribution; and
(F) fixed value of α, close to the purely greedy choice.

We summarize the results obtained by the experiments reported in [141, 142]. These four strategies were incorporated into the GRASP procedures developed for four different optimization problems: (P-1) matrix decomposition for traffic assignment in communication satellite [143], (P-2) set covering [68], (P-3) weighted MAX-SAT [153, 154], and (P-4) graph planarization [155, 161]. Let $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ be the set of possible values for the parameter α for the first three strategies. The strategy for choosing and self-tuning the value of α in the case of the Reactive GRASP procedure (R) is described later in Section 3. In the case of the strategy (E) based on using the discrete uniform distribution, all choice probabilities are equal to $1/m$. The third case corresponds to the a hybrid strategy (H), in which the authors considered $p(\alpha = 0.1) = 0.5$, $p(\alpha = 0.2) = 0.25$, $p(\alpha = 0.3) = 0.125$, $p(\alpha = 0.4) = 0.03$, $p(\alpha = 0.5) = 0.03$, $p(\alpha = 0.6) = 0.03$, $p(\alpha = 0.7) = 0.01$, $p(\alpha = 0.8) = 0.01$, $p(\alpha = 0.9) = 0.01$, and $p(\alpha = 1.0) = 0.005$. Finally, in the last strategy (F), the value of α is fixed as recommended in the original references of problems P-1 to P-4 cited above, where this parameter was tuned for each problem. A subset of the literature instances was considered for each class of test problems. The results reported in [142] are summarized in Table 2. For each problem, we first list the number of instances considered. Next, for each strategy, we give the number of times it found the best solution (hits), as well as the average CPU time (in seconds) on an IBM 9672 model R34. The number of iterations was fixed at 10,000.

Table 2 Computational results for different strategies for the variation of parameter α.

		R		*E*		*H*		*F*	
Problem	Instances	hits	time	hits	time	hits	time	hits	time
P-1	36	34	579.0	35	358.2	32	612.6	24	642.8
P-2	7	7	1346.8	6	1352.0	6	668.2	5	500.7
P-3	44	22	2463.7	23	2492.6	16	1740.9	11	1625.2
P-4	37	28	6363.1	21	7292.9	24	6326.5	19	5972.0
Total	124	91		85		78		59	

Strategy (F) presented the shortest average computation times for three out the four problem types. It was also the one with the least variability in the constructed solutions and, in consequence, found the best solution the fewest times. The reactive strategy (R) is the one which most often found the best solutions, however, at the cost of computation times that are longer than those of some of the other strategies. The high number of hits observed by strategy (E) also illustrates the effectiveness of strategies based on the variation of the RCL parameter.

3 Alternative construction mechanisms

A possible shortcoming of the standard GRASP framework is the independence of its iterations, i.e., the fact that it does not learn from the search history or from solutions found in previous iterations. This is so because the basic algorithm discards information about any solution previously encountered that does not improve the incumbent. Information gathered from good solutions can be used to implement memory-based procedures to influence the construction phase, by modifying the selection probabilities associated with each element of the RCL or by enforcing specific choices. Another possible shortcoming of the greedy randomized construction 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 large, this may not be very efficient.

In this section, we consider enhancements and alternative techniques for the construction phase of GRASP. They include random plus greedy, sampled greedy, Reactive GRASP, cost perturbations, bias functions, memory and learning, and local search on partially constructed solutions.

3.1 Random plus greedy and sampled greedy construction

In Section 2, we described the semi-greedy construction scheme used to build randomized greedy solutions that serve as starting points for local search. Two other randomized greedy approaches were proposed in [158], with smaller worst-case complexities than the semi-greedy algorithm.

Instead of combining greediness and randomness at each step of the construction procedure, the *random plus greedy* scheme applies randomness during the first p construction steps to produce a random partial solution. Next, the algorithm completes the solution with one or more pure greedy construction steps. The resulting solution is randomized greedy. One can control the balance between greediness and randomness in the construction by changing the value of the parameter p. Larger values of p are associated with solutions that are more random, while smaller values result in greedier solutions.

Similar to the random plus greedy procedure, the *sampled greedy* construction also combines randomness and greediness but in a different way. This procedure is also controlled by a parameter p. At each step of the construction process, the procedure builds a restricted candidate list by sampling $\min\{p, |C|\}$ elements of the candidate set C. Each element of the RCL is evaluated by the greedy function. The element with the smallest greedy function value is added to the partial solution. This two-step process is repeated until there are no more candidate elements. The resulting solution is also randomized greedy. The balance between greediness and randomness can be controlled by changing the value of the parameter p, i.e. the number of candidate elements that are sampled. Small sample sizes lead to more random solutions, while large sample sizes lead to greedier solutions.

3.2 Reactive GRASP

The first strategy to incorporate a learning mechanism in the memoryless construction phase of the basic GRASP was the Reactive GRASP procedure introduced in Section 2. In this case, the value of the RCL parameter α is not fixed, but instead is randomly selected at each iteration from a discrete set of possible values. This selection is guided by the solution values found along the previous iterations. One way to accomplish this is to use the rule proposed in [143]. Let $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ be a set of possible values for α. The probabilities associated with the choice of each value are all initially made equal to $p_i = 1/m$, for $i = 1, \ldots, m$. Furthermore, let z^* be the incumbent solution and let A_i be the average value of all solutions found using $\alpha = \alpha_i$, for $i = 1, \ldots, m$. 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 the more appropriate values will then increase when they are reevaluated.

The reactive approach leads to improvements over the basic GRASP in terms of robustness and solution quality, due to greater diversification and less reliance on parameter tuning. In addition to the applications in [141, 142, 143], this approach has been used in power system transmission network planning [41], job shop scheduling [40], channel assignment in mobile phone networks [93], rural road network development [171], capacitated location [57], strip-packing [11], and a combined production-distribution problem [43].

3.3 Cost perturbations

The idea of introducing some noise into the original costs is similar to that in the so-called "noising" method of Charon and Hudry [48, 49]. It adds more flexibility into algorithm design and may be even more effective than the greedy randomized construction of the basic GRASP procedure in circumstances where the construction algorithms are not very sensitive to randomization. This is indeed the case for the shortest-path heuristic of Takahashi and Matsuyama [175], used as one of the main building blocks of the construction phase of the hybrid GRASP procedure proposed by Ribeiro et al. [165] for the Steiner problem in graphs. Another situation where cost perturbations can be very effective appears when no greedy algorithm is available for straightforward randomization. This happens to be the case of the hybrid GRASP developed by Canuto et al. [46] for the prize-collecting Steiner tree problem, which makes use of the primal-dual algorithm of Goemans and Williamson [92] to build initial solutions using perturbed costs.

In the case of the GRASP for the prize-collecting Steiner tree problem described in [46], a new solution is built at each iteration using node prizes updated by a perturbation function, according to the structure of the current solution. Two different prize perturbation schemes were used. In *perturbation by eliminations*, the primal-dual algorithm used in the construction phase is driven to build a new solution without some of the nodes that appeared in the solution constructed in the previous iteration. In *perturbation by prize changes*, some noise is introduced into the node prizes to change the objective function, similarly to what is proposed in [48, 49].

The cost perturbation methods used in the GRASP for the minimum Steiner tree problem described in [165] incorporate learning mechanisms associated with intensification and diversification strategies. Three distinct weight randomization methods were applied. At a given GRASP iteration, the modified weight of each edge is randomly selected from a uniform distribution from an interval which depends on the selected weight randomization method applied at that iteration. The different weight randomization methods use frequency information and may be used to enforce intensification and diversification strategies. The experimental results reported in [165] show that the strategy combining these three perturbation methods is more robust than any of them used in isolation, leading to the best overall results on a quite broad mix of test instances with different characteristics. The GRASP heuristic using this cost perturbation strategy is among the most effective heuristics currently available for the Steiner problem in graphs.

3.4 Bias functions

In the construction procedure of the basic GRASP, the next element to be introduced in the solution is chosen at random from the candidates in the RCL. The elements of the RCL are assigned equal probabilities of being chosen. However, any probability distribution can be used to bias the selection toward some particular candidates. Another construction mechanism was proposed by Bresina [44], where 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. Several bias functions were proposed, such as:

- random bias: $\texttt{bias}(r) = 1$;
- linear bias: $\texttt{bias}(r) = 1/r$;
- log bias: $\texttt{bias}(r) = \log^{-1}(r+1)$;
- exponential bias: $\texttt{bias}(r) = e^{-r}$; and
- polynomial bias of order n: $\texttt{bias}(r) = r^{-n}$.

Let $r(\sigma)$ denote the rank of element σ and let $\texttt{bias}(r(\sigma))$ be one of the bias functions defined above. Once these values have been evaluated for all elements in the candidate set C, the probability $\pi(\sigma)$ of selecting element σ is

$$\pi(\sigma) = \frac{\texttt{bias}(r(\sigma))}{\sum_{\sigma' \in C} \texttt{bias}(r(\sigma'))}. \tag{1}$$

The evaluation of these bias functions may be restricted to the elements of the RCL. Bresina's selection procedure restricted to elements of the RCL was used in [40]. The standard GRASP uses a random bias function.

3.5 Intelligent construction: memory and learning

Fleurent and Glover [82] observed that the basic GRASP does not use long-term memory (information gathered in previous iterations) 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 relies.

Their scheme maintains a pool of elite solutions to be used in the construction phase. 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. For example, one can count identical solution vector components and set a threshold for rejection.

A *strongly determined variable* is one that cannot be changed without eroding the objective or changing significantly other variables. A *consistent variable* is one that receives a particular value in a large portion of the elite solution set. Let $I(e)$ be a measure of the strong determination and consistency features of a solution element $e \in E$. Then, $I(e)$ becomes larger as e appears more often in the pool of elite solutions. The intensity function $I(e)$ is used in the construction phase as follows. Recall that $c(e)$ is the greedy function, i.e. the incremental cost associated with the incorporation of element $e \in E$ into the solution under construction. Let $K(e) = F(c(e), I(e))$ be a function of the greedy and the intensification functions. For example, $K(e) = \lambda c(e) + I(e)$. The intensification scheme biases selection from the RCL to those elements $e \in E$ with a high value of $K(e)$ by setting its 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. For example, λ may be set to a large value that is decreased when diversification is called for. Procedures for changing the value of λ are given by Fleurent and Glover [82] and Binato et al. [40].

3.6 POP in construction

The Proximate Optimality Principle (POP) is based on the idea that "good solutions at one level are likely to be found 'close to' good solutions at an adjacent level" [90]. Fleurent and Glover [82] provided a GRASP interpretation of this principle. They suggested that imperfections introduced during steps of the GRASP construction can be "ironed-out" by applying local search during (and not only at the end of) the GRASP construction phase.

Because of efficiency considerations, a practical implementation of POP to GRASP consists in applying local search a few times during the construction phase, but not at every construction iteration. Local search was applied by Binato et al. [40] after 40% and 80% of the construction moves have been taken, as well as at the end of the construction phase.

4 Path-relinking

Path-relinking is another enhancement to the basic GRASP procedure, leading to significant improvements in both solution quality and running times. This technique was originally proposed by Glover [88] as an intensification strategy to explore trajectories connecting elite solutions obtained by tabu search or scatter search [89, 90, 91].

We consider the undirected graph associated with the solution space $G = (S, M)$, where the nodes in S correspond to feasible solutions and the edges in M correspond to moves in the neighborhood structure, i.e. $(i, j) \in M$ if and only if $i \in S$,

$j \in S$, $j \in N(i)$, and $i \in N(j)$, where $N(s)$ denotes the neighborhood of a node $s \in S$. Path-relinking is usually carried out between two solutions: one is called the *initial solution*, while the other is the *guiding solution*. One or more paths in the solution space graph connecting these solutions are explored in the search for better solutions. Local search is applied to the best solution in each of these paths, since there is no guarantee that the latter is locally optimal.

Let $s \in S$ be a node on the path between an initial solution and a guiding solution $g \in S$. Not all solutions in the neighborhood $N(s)$ are allowed to be the next on the path from s to g. We restrict the choice only to those solutions that are more similar to g than s. This is accomplished by selecting moves from s that introduce attributes contained in the guiding solution g. Therefore, path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions (i.e. the guiding elite solutions), by favoring these attributes in the selected moves.

The use of path-relinking within a GRASP procedure, as an intensification strategy applied to each locally optimal solution, was first proposed by Laguna and Martí [106]. It was followed by several extensions, improvements, and successful applications [6, 7, 18, 46, 75, 130, 146, 156, 158, 159, 163, 165, 171]. A survey of GRASP with path-relinking can be found in [157].

Enhancing GRASP with path-relinking almost always improves the performance of the heuristic. As an illustration, Figure 6 shows time-to-target plots for GRASP and GRASP with path-relinking implementations for four different applications. These time-to-target plots show the empirical cumulative probability distributions of the *time-to-target* random variable when using pure GRASP and GRASP with path-relinking, i.e., the time needed to find a solution at least as good as a prespecified target value. For all problems, the plots show that GRASP with path-relinking is able to find target solutions faster than GRASP.

GRASP with path-relinking makes use of an *elite set* to collect a diverse pool of high-quality solutions found during the search. This pool is limited in size, i.e. it can have at most `Max_Elite` solutions. Several schemes have been proposed for the implementation of path-relinking, which may be applied as:

- an intensification strategy, between each local optimum obtained after the local search phase and one or more elite solutions;
- a post-optimization step, between every pair of elite solutions;
- an intensification strategy, periodically (after a fixed number of GRASP iterations since the last intensification phase) submitting the pool of elite solutions to an evolutionary process (see Subsection 4.7);
- a post-optimization phase, submitting the pool of elite solutions to an evolutionary process; or
- any other combination of the above schemes.

The pool of elite solutions is initially empty. Each locally optimal solution obtained by local search and each solution resulting from path-relinking is considered as a candidate to be inserted into the pool. If the pool is not yet full, the candidate is simply added to the pool. Otherwise, if the candidate is better than the incumbent, it replaces an element of the pool. In case the candidate is better than the worst ele-

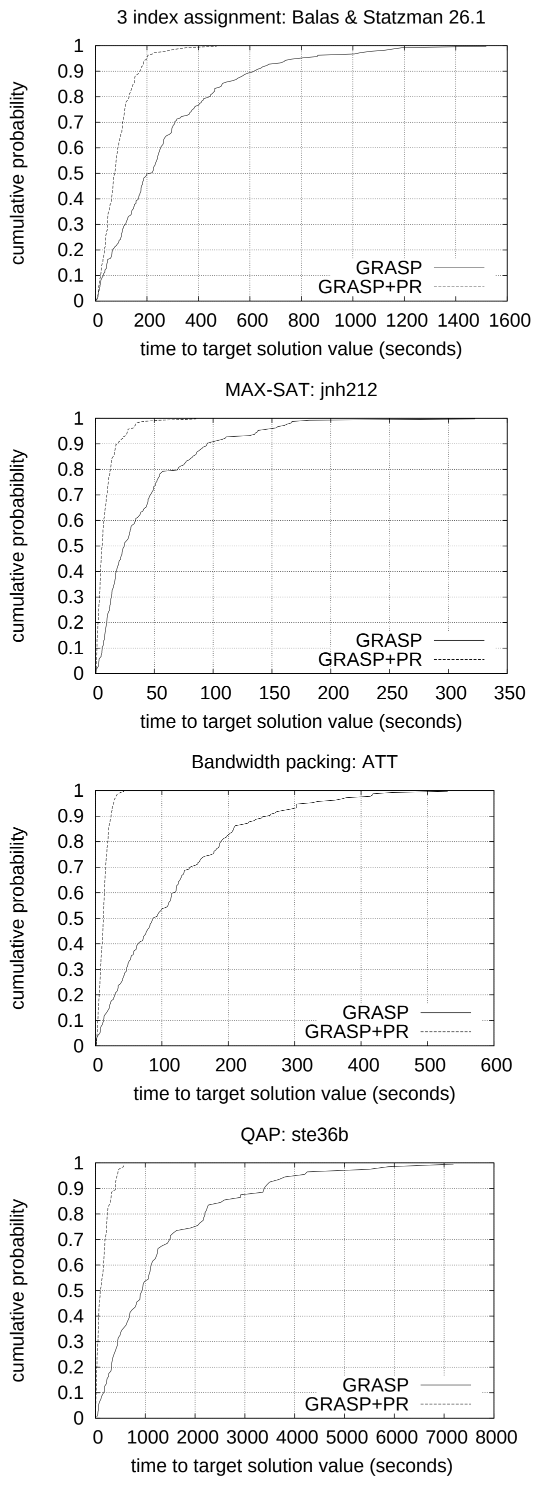


Fig. 6 Time to target plots comparing running times of pure GRASP and GRASP with path-relinking on four instances of distinct problem types: three index assignment, maximum satisfiability, bandwidth packing, and quadratic assignment.

ment of the pool but not better than the best element, then it replaces some element of the pool if it is sufficiently different from every other solution currently in the pool. To balance the impact on pool quality and diversity, the element selected to be replaced is the one that is most similar to the entering solution among those elite solutions of quality no better than the entering solution [158].

Given a local optimum s_1 produced at the end of a GRASP iteration, we need to select at random from the pool a solution s_2 to be path-relinked with s_1. In principle, any pool solution could be selected. However, we may want to avoid pool solutions that are too similar to s_1, because relinking two solutions that are similar limits the scope of the path-relinking search. If the solutions are represented by $|E|-$dimensional incidence vectors, we should privilege pairs of solutions for which the Hamming distance (i.e., the number of components that take on different values in each solution) between them is high. A strategy introduced in [158] is to select a pool element s_2 at random with probability proportional to the Hamming distance between the pool element and the local optimum s_1. Since the number of paths between two solutions grows exponentially with their Hamming distance, this strategy favors pool elements that have a large number of paths connecting them to and from s_1.

After determining which solution (s_1 or s_2) will be designated the initial solution i and which will be the guiding solution g, the algorithm starts by computing the set $\Delta(i,g)$ of components in which i and g differ. This set corresponds to the moves which should be applied to i to reach g. Starting from the initial solution, the best move in $\Delta(i,g)$ still not performed is applied to the current solution, until the guiding solution is reached. By best move, we mean the one that results in the highest quality solution in the restricted neighborhood. The best solution found along this trajectory is submitted to local search and returned as the solution produced by the path-relinking algorithm.

Several alternatives have been considered and combined in recent implementations. These include forward, backward, back and forward, mixed, truncated, greedy randomized adaptive, and evolutionary path-relinking. All these alternatives involve trade-offs between computation time and solution quality.

4.1 Forward path-relinking

In *forward* path-relinking, the GRASP local optimum is designated as the initial solution and the pool solution is made the guiding solution. This is the original scheme proposed by Laguna and Martí [106].

4.2 Backward path-relinking

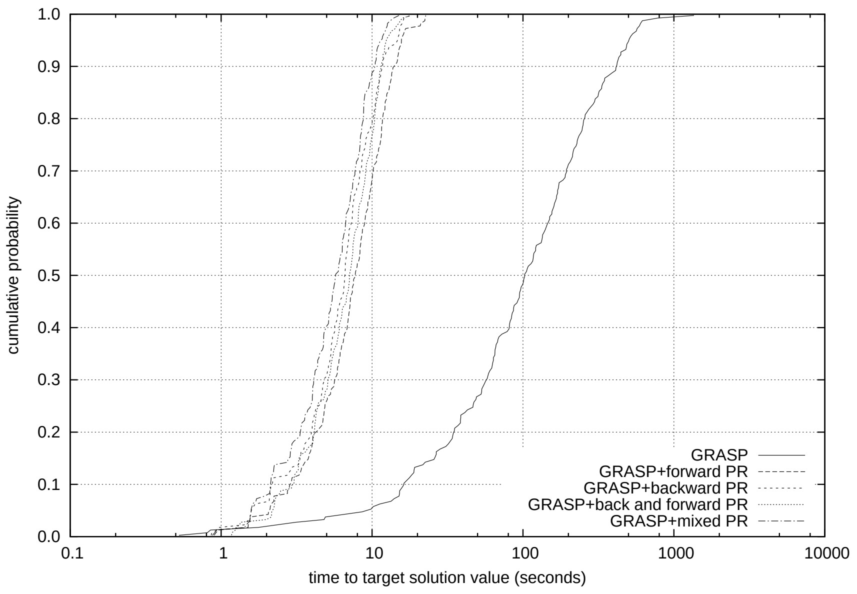
In *backward* path-relinking, the pool solution is designated as the initial solution and the GRASP local optimum is made the guiding one. This scheme was originally proposed in Aiex et al. [7] and Resende and Ribeiro [156]. The main advantage of this approach over forward path-relinking comes from the fact that, in general, there are more high-quality solutions near pool elements than near GRASP local optima. Backward path-relinking explores more thoroughly the neighborhood around the pool solution, whereas forward path-relinking explores more the neighborhood around the GRASP local optimum. Experiments in [7, 156] have shown that backward path-relinking usually outperforms forward path-relinking.

4.3 Back and forward path-relinking

Back and forward path-relinking combines forward and backward path-relinking. As shown in [7, 156], it finds solutions at least as good as forward path-relinking or backward path-relinking, but at the expense of taking about twice as long to run. The reason that back and forward path-relinking often finds solutions of better quality than simple backward or forward path-relinking stems from the fact that it thoroughly explores the neighborhoods of both solutions s_1 and s_2.

4.4 Mixed path-relinking

Mixed path-relinking shares the benefits of back and forward path-relinking, i.e. it thoroughly explores both neighborhoods, but does so in about the same time as forward or backward path-relinking alone. This is achieved by interchanging the roles of the initial and guiding solutions at each step of the path-relinking procedure. Therefore, two paths are generated, one starting at s_1 and the other at s_2. The paths evolve and eventually meet at some solution about half way between s_1 and s_2. The joined path relinks these two solutions. Mixed path-relinking was suggested by Glover [88] and was first implemented and tested by Ribeiro and Rosseti [163], where it was shown to outperform forward, backward, and back and forward path-relinking. Figure 7 shows a comparison of pure GRASP and four variants of path-relinking: forward, backward, back and forward, and mixed. The time-to-target plots show that GRASP with mixed path-relinking has the best running time profile among the variants compared.



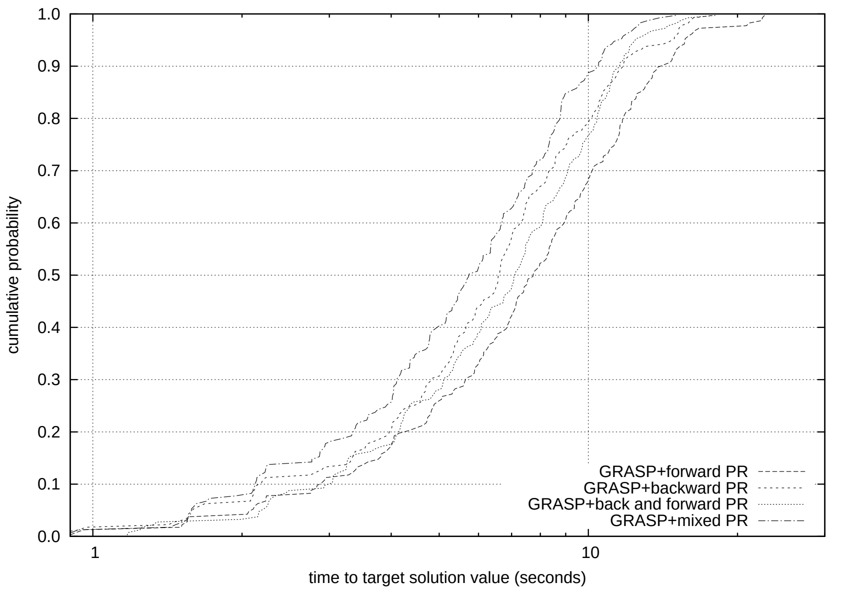


Fig. 7 Time-to-target plots for pure GRASP and four variants of GRASP with path-relinking (forward, backward, back and forward, and mixed) on an instance of the 2-path network design problem.

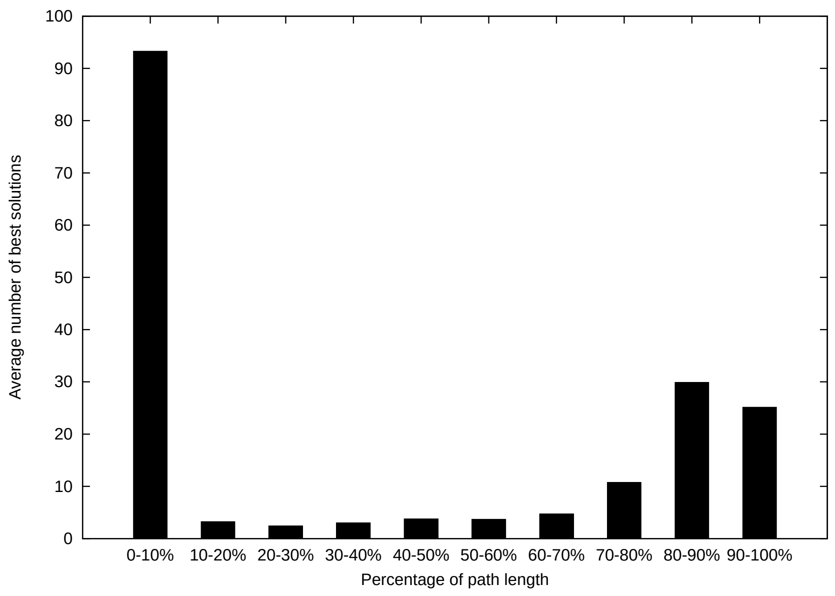


Fig. 8 Average number of best solutions found at different depths of the path from the initial solution to the guiding solution on instances of the max-min diversity problem.

4.5 Truncated path-relinking

Since good-quality solutions tend to be near other good-quality solutions, one would expect to find the best solutions with path-relinking near the initial or guiding solution. Indeed, Resende et al. [151] showed that this is the case for instances of the max-min diversity problem, as shown in Figure 8. In that experiment, a back and forward path-relinking scheme was tested. The figure shows the average number of best solutions found by path-relinking taken over several instances and several applications of path-relinking. The 0-10% range in this figure corresponds to subpaths near the initial solutions for the forward path-relinking phase as well as the backward phase, while the 90-100% range are subpaths near the guiding solutions. As the figure indicates, exploring the subpaths near the extremities may produce solutions about as good as those found by exploring the entire path. There is a higher concentration of better solutions close to the initial solutions explored by path-relinking.

Truncated path-relinking can be applied to either forward, backward, backward and forward, or mixed path-relinking. Instead of exploring the entire path, truncated path-relinking only explores a fraction of the path and, consequently, takes a fraction of the time to run. Truncated path-relinking has been applied in [18, 151].

4.6 Greedy randomized adaptive path-relinking

In path-relinking, the best not yet performed move in set $\Delta(i,g)$ is applied to the current solution, until the guiding solution is reached. If ties are broken deterministically, this strategy will always produce the same path between the initial and guiding solutions. Since the number of paths connecting i and g is exponential in $|\Delta(i,g)|$, exploring a single path can be somewhat limiting.

Greedy randomized adaptive path-relinking, introduced by Binato et al. [39], is a semi-greedy version of path-relinking. Instead of taking the best move in $\Delta(i,g)$ still not performed, a restricted candidate list of good moves still not performed is set up and a randomly selected move from the latter is applied. By applying this strategy several times between the initial and guiding solutions, several paths can be explored. Greedy randomized adaptive path-relinking has been applied in [18, 63, 151].

4.7 Evolutionary path-relinking

GRASP with path-relinking maintains a pool of elite solutions. Applying path-relinking between pairs of pool solutions may result in an even better pool of solutions. Aiex et al. [7] applied path-relinking between all pairs of elite solutions as an intensification scheme to improve the quality of the pool and as a post-optimization step. The application of path-relinking was repeated until no further improvement was possible.

Resende and Werneck [158, 159] described an *evolutionary* path-relinking scheme applied to pairs of elite solutions and used as a post-optimization step. The pool resulting from the GRASP with path-relinking iterations is referred to as population P_0. At step k, all pairs of elite set solutions of population P_k are relinked and the resulting solutions made candidates for inclusion in population P_{k+1} of the next generation. The same rules for acceptance into the pool during GRASP with path-relinking are used for acceptance into P_{k+1}. If the best solution in P_{k+1} is better than the best in P_k, then k is incremented by one and the process is repeated. Resende et al. [151] describe another way to implement evolutionary path-relinking, where a single population is maintained. Each pair of elite solutions is relinked and the resulting solution is a candidate to enter the elite set. If accepted, it replaces an existing elite solution. The process is continued while there are still pairs of elite solutions that have not yet been relinked.

Andrade and Resende [17] used this evolutionary scheme as an intensification process every 100 GRASP iterations. During the intensification phase, every solution in the pool is relinked with the two best ones. Since two elite solutions might be relinked more than once in different calls to the intensification process, greedy randomized adaptive path-relinking was used.

Resende et al. [151] showed that a variant of GRASP with evolutionary path-relinking outperformed several other heuristics using GRASP with path-relinking, simulated annealing, and tabu search for the max-min diversity problem.

5 Extensions

In this section, we comment on some extensions, implementation strategies, and hybridizations of GRASP.

The use of hashing tables to avoid cycling in conjunction with tabu search was proposed by Woodruff and Zemel [178]. A similar approach was later explored by Ribeiro et al. [162] in their tabu search algorithm for query optimization in relational databases. In the context of GRASP implementations, hashing tables were first used by Martins et al. [122] in their multi-neighborhood heuristic for the Steiner problem in graphs, to avoid the application of local search to solutions already visited in previous iterations.

Filtering strategies have also been used to speed up the iterations of GRASP, see e.g. [70, 122, 143]. In these cases, local search is not applied to all solutions obtained at the end of the construction phase, but instead only to some promising unvisited solutions, defined by a threshold with respect to the incumbent.

Almost all randomization effort in the basic GRASP algorithm involves the construction phase. Local search stops at the first local optimum. On the other hand, strategies such as VNS (Variable Neighborhood Search), proposed by Hansen and Mladenović [96, 125], rely almost entirely on the randomization of the local search to escape from local optima. With respect to this issue, GRASP and variable neighborhood strategies may be considered as complementary and potentially capable of leading to effective hybrid methods. A first attempt in this direction was made by Martins et al. [122]. The construction phase of their hybrid heuristic for the Steiner problem in graphs follows the greedy randomized strategy of GRASP, while the local search phase makes use of two different neighborhood structures as a VND (variable neighborhood descent) procedure [96, 125]. Their heuristic was later improved by Ribeiro et al. [165], one of the key components of the new algorithm being another strategy for the exploration of different neighborhoods. Ribeiro and Souza [164] also combined GRASP with VND in a hybrid heuristic for the degree-constrained minimum spanning tree problem. Festa et al. [81] studied different variants and combinations of GRASP and VNS for the MAX-CUT problem, finding and improving the best known solutions for some open instances from the literature.

GRASP has also been used in conjunction with genetic algorithms. Basically, the greedy randomized strategy used in the construction phase of a GRASP heuristic is applied to generate the initial population for a genetic algorithm. We may cite e.g. the genetic algorithm of Ahuja et al. [5] for the quadratic assignment problem, which makes use of the GRASP heuristic proposed by Li et al. [108] to create the initial population of solutions. A similar approach was used by Armony et al. [27], with the initial population made up by both randomly generated solutions and those built by a GRASP algorithm.

The hybridization of GRASP with tabu search was first studied by Laguna and González-Velarde [105]. Delmaire et al. [57] considered two approaches. In the first, GRASP is applied as a powerful diversification strategy in the context of a tabu search procedure. The second approach is an implementation of the Reactive GRASP algorithm presented in Section 3.2, in which the local search phase is strengthened by tabu search. Results reported for the capacitated location problem show that the hybrid approaches perform better than the isolated methods previously used. Two two-stage heuristics are proposed in [1] for solving the multi-floor facility layout problem. GRASP/TS applies a GRASP to find the initial layout and tabu search to refine it.

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 [37]. Lourenço et al. [112] point out that ILS has been rediscovered many times and is also known as iterated descent [35, 36], large step Markov chains [120], iterated Lin-Kernighan [100], and chained local optimization [119]. ILS can be hybridized with GRASP by replacing the standard local search. The GRASP construction produces a solution which is passed to the ILS procedure. Ribeiro and Urrutia [166] presented a hybrid GRASP with ILS heuristic for the mirrored traveling tournament problem, in which perturbations are achieved by randomly generating solutions in the game rotation ejection chain [86, 87] neighborhood.

6 Parallel GRASP

Cung et al. [55] noted that parallel implementations of metaheuristics not only appear as quite natural alternatives to speed up the search for good approximate solutions, but also facilitate solving larger problems and finding improved solutions, with respect to their sequential counterparts. This is due to the partitioning of the search space and to the increased possibilities for search intensification and diversification. As a consequence, parallelism can improve the effectiveness and robustness of metaheuristic-based algorithms. Parallel metaheuristic-based algorithms are less dependent on time consuming parameter tuning experiments and their success is not limited to a few or small classes of problems.

Recent years have witnessed huge advances in computer technology and communication networks. The growing computational power requirements of large scale applications and the high costs of developing and maintaining supercomputers has fueled the drive for cheaper high performance computing environments. With the considerable increase in commodity computers and network performance, cluster computing and, more recently, grid computing [83, 84] have emerged as real alternatives to traditional super-computing environments for executing parallel applications that require significant amounts of computing power.

6.1 Cluster computing

A computing cluster generally consists of a fixed number of homogeneous resources, interconnected on a single administrative network, which together execute one parallel application at a time.

Most parallel implementations of GRASP follow the *multiple-walk independent thread* strategy, based on the distribution of the iterations over the processors [12, 13, 70, 108, 121, 123, 128, 134, 135]. In general, each search thread has to perform `Max_Iterations`$/p$ iterations, where p and `Max_Iterations` are, respectively, the number of processors and the total number of iterations. Each processor has a copy of the sequential algorithm, a copy of the problem data, and an independent seed to generate its own pseudo-random number sequence. A single global variable is required to store the best solution found over all processors. One of the processors acts as the master, reading and distributing problem data, generating the seeds which will be used by the pseudo-random number generators at each processor, distributing the iterations, and collecting the best solution found by each processor. Since the iterations are completely independent and very little information is exchanged, linear speedups are easily obtained provided that no major load imbalance problems occur. The iterations may be evenly distributed over the processors or according with their demands, to improve load balancing.

Martins et al. [123] implemented a parallel GRASP for the Steiner problem in graphs. Parallelization is achieved by the distribution of the iterations over the processors, with the value of the RCL parameter α randomly chosen in the interval $[0.0, 0.3]$ at each iteration. Almost-linear speedups were observed on benchmark problems from the OR-Library [38] for 2, 4, 8, and 16 processors, with respect to the sequential implementation. Path-relinking may be used in conjunction with parallel implementations of GRASP. Almost-linear speedups were also obtained with the multiple-walk independent-thread implementation of Aiex et al. [7] for the 3-index assignment problem, in which each processor applies path-relinking to pairs of elite solutions stored in a local pool.

Alvim and Ribeiro [12, 13] have shown that multiple-walk independent-thread approaches for the parallelization of GRASP may benefit much from load balancing techniques, whenever heterogeneous processors are used or if the parallel machine is simultaneously shared by several users. In this case, almost-linear speedups may be obtained with a heterogeneous distribution of the iterations over the p processors in q packets, with $q > p$. Each processor starts performing one packet of $\lceil$`Max_Iterations`$/q\rceil$ iterations and informs the master when it finishes its packet of iterations. The master stops the execution of each worker processor when there are no more iterations to be performed and collects the best solution found. Faster or less loaded processors will perform more iterations than the others. In the case of the parallel GRASP heuristic implemented for the problem of traffic assignment described in [143], this dynamic load balancing strategy allowed reductions in the elapsed times of up to 15% with respect to the times observed for the static strategy, in which the iterations were uniformly distributed over the processors.

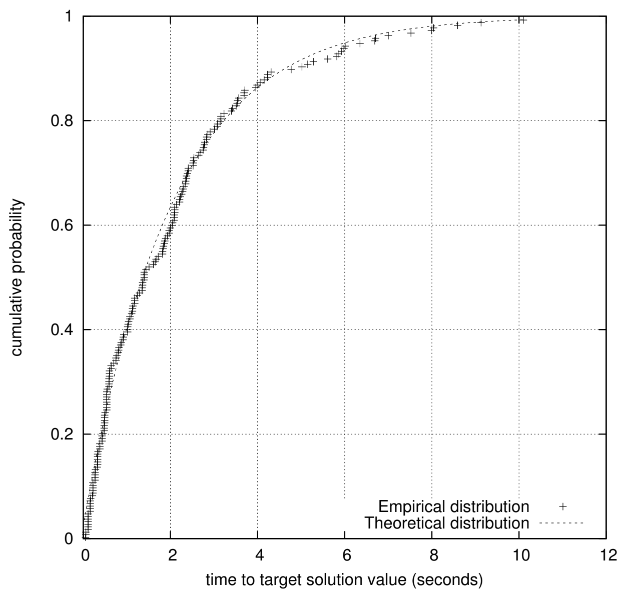


Fig. 9 Superimposed empirical and theoretical distributions (times to target solution values measured in seconds on an SGI Challenge computer with 28 processors).

For a given problem instance and a target value `look4`, let *time-to-target* be a random variable representing the time taken by a GRASP implementation to find a solution whose cost is at least as good as `look4` for this instance. Aiex et al. [8] have shown experimentally that this random variable fits an exponential distribution or, in the case where the setup times are not negligible, a shifted (two-parameter) exponential distribution. The probability density function $p(t)$ of the random variable time-to-target is given by $p(t) = (1/\lambda) \cdot e^{-t/\lambda}$ in the first case or by $p(t) = (1/\lambda) \cdot e^{-(t-\mu)/\lambda}$ in the second, with the parameters $\lambda \in \mathbf{R}^+$ and $\mu \in \mathbf{R}^+$ being associated with the shape and the shift of the exponential function, respectively.

Figure 9 illustrates this result, depicting the superimposed empirical and theoretical distributions observed for one of the cases studied along the computational experiments reported in [8], which involved 2400 runs of GRASP procedures for each of five different problem types: maximum independent set [70, 150], quadratic assignment [108, 152], graph planarization [155, 161], maximum weighted satisfiability [154], and maximum covering [148].

We now assume that p identical processors are available and used to search in parallel for the same target value `look4`. Let X_i be the time taken by processor $i = 1, \dots, p$ to find a solution whose cost is at least as good as `look4` and consider the random variable $Y = \min\{X_1, \dots, X_p\}$. Since all processors are independent and fit the same exponential distribution with average equal to λ, the random variable Y fits an exponential distribution whose average is λ/p. Therefore, linear speedups can be achieved if multiple identical processors are used independently to search in parallel for the same target value.

However, we notice that if path-relinking is applied as an intensification step at the end of each GRASP iteration (see e.g. [46, 157]), then the iterations are no longer independent and the memoryless characteristic of GRASP may be destroyed. Consequently, the time-to-target random variable may not fit an exponential distribution. Aiex et al. [7] have shown experimentally that, even in this case, the time-to-target random variable may be reasonably approximated by a shifted (two-parameter) exponential distribution in some circumstances.

In the case of *multiple-walk cooperative-thread* strategies, the search threads running in parallel exchange and share information collected along the trajectories they investigate. One expects not only to speed up the convergence to the best solution but, also, to find better solutions than independent-thread strategies. The most difficult aspect to be set up is the determination of the nature of the information to be shared or exchanged to improve the search, without taking too much additional memory or time to be collected. Cooperative-thread strategies may be implemented using path-relinking, by combining elite solutions stored in a central pool with the local optima found by each processor at the end of each GRASP iteration.

Ribeiro and Rosseti [163] applied this scheme to implement a parallel GRASP heuristic for the 2-path network design problem. One of the processors acts as the master and handles a centralized pool of elite solutions, collecting and distributing them upon request. The other processors act as workers and exchange the elite solutions found along their search trajectories. Cooperation between the processors is implemented via path-relinking using the centralized pool of elite solutions. In this implementation, each worker may send up to three different solutions to the master at each GRASP iteration: the solution obtained by local search and the solutions obtained by forward and backward path-relinking. The performance of the parallel implementation is quite uniform over all problem instances.

Computational results illustrating the behavior of the independent and cooperative parallel implementations for an instance with 100 nodes, 4950 edges, and 1000 origin-destination pairs are presented below. The plots in Figure 10 display the empirical probability distribution of the time-to-target random variable for both the independent and the cooperative parallel implementations in C and MPI, for 200 runs on 2, 4, 8, and 16 processors of a 32-machine Linux cluster, with the `look4` target value set at 683. We notice that the independent strategy performs better when only two processors are used. This is so because the independent strategy makes use of the two processors to perform GRASP iterations, while the cooperative strategy makes use of one processor to perform iterations and the other to handle the pool. However, as the number of processors increases, the gain obtained through cooperation becomes more important than the loss of one processor to perform iterations. The cooperative implementation is already faster than the independent one for eight processors. These plots establish the usefulness and the efficiency of the cooperative implementation. Other implementations of multiple-walk cooperative-thread GRASP heuristics can be found e.g. in Aiex et al. [6, 7].

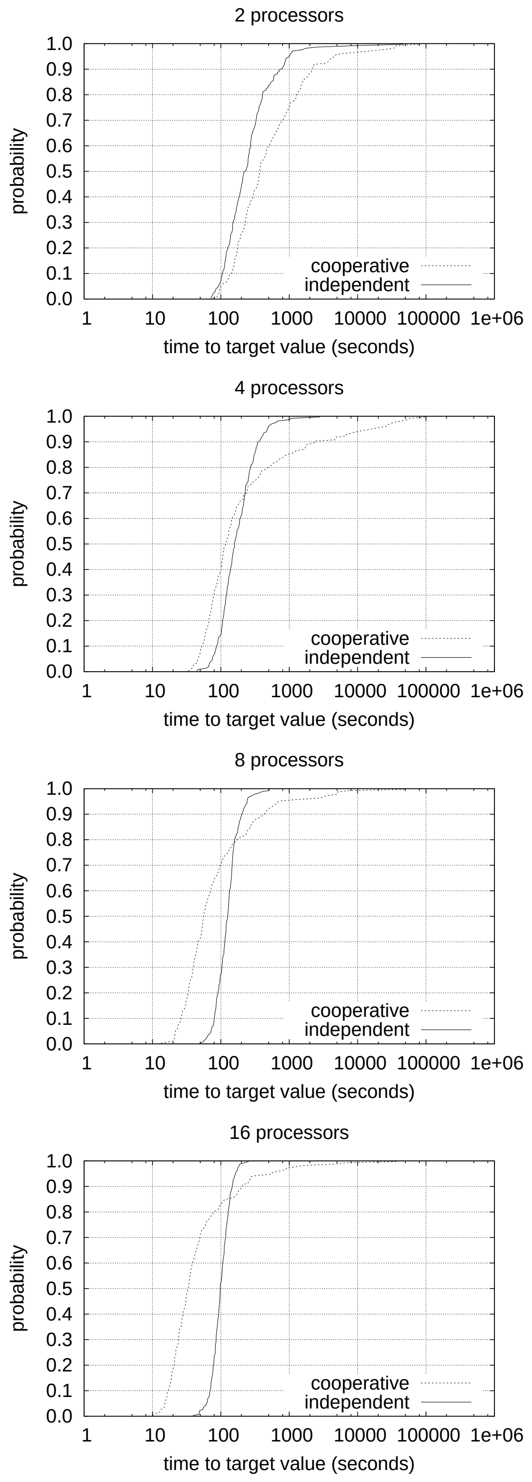


Fig. 10 Empirical distributions of the time-to-target random variable for the independent and the cooperative parallelizations on 2, 4, 8, and 16 processors (target value set at 683).

6.2 Grid computing

Grids aim to harness available computing power from a diverse pool of resources available over the Internet to execute a number of applications simultaneously. Grids aggregate geographically distributed collections (or sites) of resources which typically have different owners and thus are shared between multiple users. The fact that these resources are distributed, heterogeneous, and non-dedicated requires careful consideration when developing grid enabled applications and makes writing parallel grid-aware heuristics very challenging [83].

Araújo et al. [22] described some strategies based on the master-worker paradigm for the parallelization in grid environments of the hybrid GRASP with ILS heuristic for the mirrored traveling tournament problem proposed in [166]. In the best of these strategies, PAR-*M*P, the master is dedicated to managing a centralized pool of elite solutions, including collecting and distributing them upon request. The workers start their searches from different initial solutions and exchange elite solutions found along their trajectories. Although it lead to improvements in the results obtained by the sequential implementation, it was not able to make full use of the characteristics of grid environments.

Araújo [21] proposed an autonomic hierarchical distributed strategy for the implementation of cooperative metaheuristics in grids, in which local pools of elite solutions internal to each site support intensification strategies, while a global pool is used to ensure diversification. This autonomic strategy is much more adapted to grid computations and leads to better results with respect to both the master-worker PAR-*M*P parallel strategy for the mirrored traveling tournament problem and the sequential hybrid heuristic combining GRASP and ILS for the diameter constrained minimum spanning tree problem [114].

Table 3 displays comparative results reported in [21] for large National Football League instances of the mirrored traveling tournament problem with the number of teams ranging from 16 to 32. For each instance, we give the costs of the solutions obtained by the sequential implementation and by the hierarchical strategy running on ten processors. The running times range from approximately three to ten hours, as observed for instances `nfl18` and `nfl24`, respectively. We notice that the hierarchical strategy improved the solutions obtained by the sequential heuristic for eight out of the nine test instances.

Figure 11 displays time-to-target plots obtained after 100 runs of the hierarchical distributed implementation of the GRASP with ILS heuristic for the diameter constrained minimum spanning tree on a typical instance with 100 nodes, using 15, 30, and 60 processors. These plots show that the approach scales appropriately when the number of processors increase. We display in Table 4 some results obtained by the sequential and the hierarchical distributed implementations of the GRASP with ILS heuristic for this problem. The distributed strategy runs on ten processors. The sequential heuristic is allowed to run by as much as ten times the time taken by the grid implementation. We give the number of nodes and edges for each instance, together with the costs of the best solutions found by each implementation and the time given to the sequential heuristic. These results illustrate the robustness of the

Table 3 Solution costs found by the sequential and grid implementations of the hybrid GRASP with ILS heuristic for the mirrored traveling tournament problem.

Instance	Sequential	Grid
`nfl16`	251289	249806
`nfl18`	299903	299112
`nfl20`	359748	359748
`nfl22`	418086	418022
`nfl24`	467135	465491
`nfl26`	554670	548643
`nfl28`	618801	609788
`nfl30`	740458	739697
`nfl32`	924559	914620

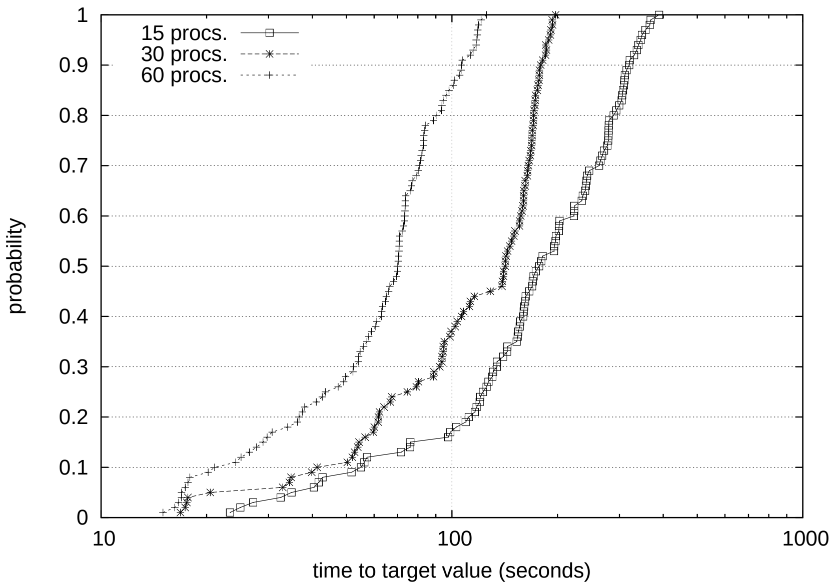


Fig. 11 Time-to-target plots for the hierarchical distributed implementation of the GRASP with ILS heuristic for the diameter constrained minimum spanning tree on an instance with 100 nodes running on a grid environment using 15, 30, and 60 processors.

hierarchical distributed strategy (due to the effectiveness of the cooperation through the pools in two different levels), since it was able to systematically find better solutions than those obtained by the sequential strategy in computation times ten times larger.

Table 4 Best solutions found by the sequential heuristic and by the grid implementation running on ten processors. The sequential heuristic is allowed to run by as much as ten times the time taken by the grid implementation.

Nodes	Edges	Grid	Sequential	Time (seconds)
60	600	738.000	740.000	2300.00
60	600	150.000	152.000	400.00
70	2415	6.981	6.983	230.00
70	2415	7.486	7.499	3000.00
70	2415	7.238	7.245	690.00
100	4950	7.757	7.835	1400.00
100	4950	7.930	7.961	5000.00
100	4950	8.176	8.204	3400.00

7 Applications

The first application of GRASP described in the literature concerned the set covering problem [68]. Since then, GRASP has been applied to a wide range of problems. The main applications areas are summarized below with links to specific references:

- routing [25, 29, 34, 45, 47, 53, 103, 110]
- logic [58, 75, 135, 149, 153, 154]
- covering and partitioning [10, 23, 26, 68, 85, 94]
- location [50, 54, 95, 1, 98, 57, 101, 132, 176, 177]
- minimum Steiner tree [46, 121, 122, 123, 165]
- optimization in graphs [2, 3, 4, 28, 70, 76, 77, 99, 104, 107, 116, 117, 122, 133, 137, 148, 150, 155, 161, 165, 173]
- assignment [5, 7, 67, 82, 108, 111, 113, 124, 127, 128, 130, 134, 136, 139, 143, 144, 152, 169]
- timetabling, scheduling, and manufacturing [6, 9, 11, 16, 18, 20, 31, 32, 33, 40, 43, 52, 56, 59, 64, 65, 66, 71, 72, 102, 105, 109, 126, 145, 166, 167, 168, 170, 179, 180]
- transportation [25, 30, 64, 67, 172]
- power systems [41, 42, 63]
- telecommunications [2, 15, 14, 16, 18, 27, 51, 101, 111, 138, 140, 143, 147, 148, 156, 174]
- graph and map drawing [54, 73, 106, 115, 116, 118, 131, 155, 161]
- biology [19, 62, 74]
- VLSI [23, 24]

The reader is referred to Festa and Resende [80] for a complete annotated bibliography of GRASP applications.

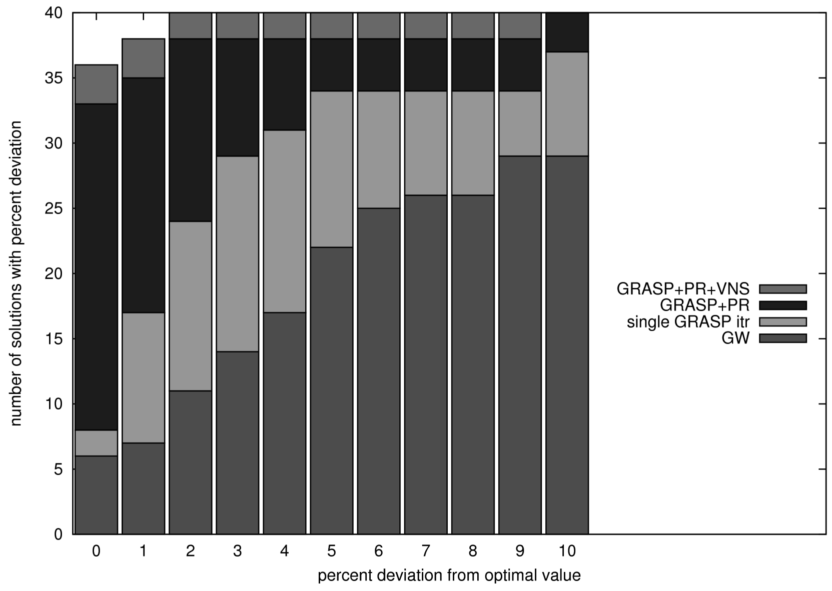


Fig. 12 Performance of GW approximation algorithm, a single GRASP iteration (GW followed by local search), 500 iterations of GRASP with path-relinking, and 500 iterations of GRASP with path-relinking followed by VNS for series C prize-collecting Steiner tree problems.

8 Concluding remarks

The results described in this chapter reflect successful applications of GRASP to a large number of classical combinatorial optimization problems, as well as to those that arise in real-world situations in different areas of business, science, and technology.

We underscore the simplicity of implementation of GRASP, which makes use of simple building blocks (solution construction procedures and local search methods) that are often readily available. Contrary to what occurs with other metaheuristics, such as tabu search or genetic algorithms, which make use of a large number of parameters in their implementations, the basic version of GRASP requires the adjustment of a single parameter.

Recent developments, presented in this chapter, show that different extensions of the basic procedure allow further improvements in the solutions found by GRASP. Among these, we highlight reactive GRASP, which automates the adjustment of the restricted candidate list parameter; variable neighborhoods, which permit accelerated and intensified local search; and path-relinking, which beyond allowing the implementation of intensification strategies based on the memory of elite solutions, opens the way for the development of very effective cooperative parallel strategies.

These and other extensions make up a set of tools that can be added to simpler heuristics to find better-quality solutions. To illustrate the effect of additional extensions on solution quality, Figure 12 shows some results obtained for the prize-collecting Steiner tree problem, as discussed in [46]. We consider the 40 instances of series C. The figure shows results for eleven different levels of solution accuracy (varying from optimal to ten percent from optimal). For each level of solution accuracy, the figure shows the number of instances for which each component found solutions within the accuracy level. The components were the primal-dual constructive algorithm (GW) of Goemans and Williamson [92], GW followed by local search (GW+LS), corresponding to the first GRASP iteration, 500 iterations of GRASP with path-relinking (GRASP+PR), and the complete algorithm, using variable neighborhood search as a post-optimization procedure (GRASP+PR+VNS). For example, we observe that the number of optimal solutions found goes from six, using only the constructive algorithm, to a total of 36, using the complete algorithm described in [46]. The largest relative deviation with respect to the optimal value decreases from 36.4% in the first case, to only 1.1% for the complete algorithm. It is easy to notice the contribution made by each additional extension.

Parallel implementations of GRASP are quite robust and lead to linear speedups both in independent and cooperative strategies. Cooperative strategies are based on the collaboration between processors through path-relinking and a global pool of elite solutions. This allows the use of more processors to find better solutions in less computation time.

References

1. S. Abdinnour-Helm and S.W. Hadley. Tabu search based heuristics for multi-floor facility layout. *International J. of Production Research*, 38:365–383, 2000.
2. J. Abello, P.M. Pardalos, and M.G.C. Resende. On maximum clique problems in very large graphs. In J. Abello and J. Vitter, editors, *External Memory Algorithms and Visualization*, volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 199–130. American Mathematical Society, 1999.
3. J. Abello, M.G.C. Resende, and S. Sudarsky. Massive quasi-clique detection. In S. Rajsbaum, editor, *LATIN 2002: Theoretical Informatics*, volume 2286 of *Lecture Notes in Computer Science*, pages 598–612. Springer-Verlag, 2002.
4. R.K. Ahuja, J.B. Orlin, and D. Sharma. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91:71–97, 2001.
5. 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.
6. R.M. Aiex, S. Binato, and M.G.C. Resende. Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing*, 29:393–430, 2003.
7. 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.
8. R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. Probability distribution of solution time in GRASP: An experimental investigation. *J. of Heuristics*, 8:343–373, 2002.
9. R. Álvarez-Valdés, E. Crespo, J.M. Tamarit, and F. Villa. GRASP and path relinking for project scheduling under partially renewable resources. *European J. of Operational Research*, 189:1153–1170, 2008.
10. R. Álvarez-Valdés, F. Parreno, and J.M. Tamarit. A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems. *J. of the Operational Research Society*, 56:414–425, 2005.

11. R. Alvarez-Valdesa, F. Parreno, and J.M. Tamarit. Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, 35:1065–1083, 2008.
12. A.C. Alvim. Parallelization strategies for the metaheuristic GRASP. Master's thesis, Department of Computer Science, Catholic University of Rio de Janeiro, Brazil, 1998. In Portuguese.
13. A.C. Alvim and C.C. Ribeiro. Load balancing for the parallelization of the GRASP metaheuristic. In *Proceedings of the X Brazilian Symposium on Computer Architecture*, pages 279–282, Búzios, 1998. In Portuguese.
14. E. Amaldi, A. Capone, and F. Malucelli. Planning UMTS base station location: Optimization models With power control and algorithms. *IEEE Transactions on Wireless Communications*, 2:939–952, 2003.
15. E. Amaldi, A. Capone, F. Malucelli, and F. Signori. Optimization models and algorithms for downlink UMTS radio planning. In *Proceedings of Wireless Communications and Networking*, volume 2, pages 827–831, 2003.
16. D.V. Andrade and M.G.C. Resende. A GRASP for PBX telephone migration scheduling. In *Proceedings of The Eighth INFORMS Telecommunications Conference*, 2006.
17. D.V. Andrade and M.G.C. Resende. GRASP with evolutionary path-relinking. Technical Report TD-6XPTS7, AT&T Labs Research, Florham Park, 2007.
18. D.V. Andrade and M.G.C. Resende. GRASP with path-relinking for network migration scheduling. In *Proceedings of the International Network Optimization Conference*, 2007.
19. A.A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. *J. of Heuristics*, 8:429–447, 2002.
20. C. Andrés, C. Miralles, and R. Pastor. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European J. of Operational Research*, 187:1212–1223, 2008.
21. A.P.F. Araújo. *Autonomic Parallelization of Metaheuristics in Grid Environments*. PhD thesis, Department of Computer Science, Catholic University of Rio de Janeiro, 2008. In Portuguese.
22. A.P.F. Araújo, C. Boeres, V.E.F. Rebello, C.C. Ribeiro, and S. Urrutia. Exploring grid implementations of parallel cooperative metaheuristics: A case study for the mirrored traveling tournament problem. In K.F. Doerner, M. Gendreau, P. Greistorfer, W. Gutjahr, R.F. Hartl, and M. Reimann, editors, *Metaheuristics: Progress in Complex Systems Optimization*, pages 297–322. Springer, 2007.
23. S. Areibi and A. Vannelli. A GRASP clustering technique for circuit partitioning. In J. Gu and P.M. Pardalos, editors, *Satisfiability Problems*, volume 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 711–724. American Mathematical Society, 1997.
24. S.M. Areibi. GRASP: An effective constructive technique for VLSI circuit partitioning. In *Proc. IEEE Canadian Conference on Electrical and Computer Engineering*, 1999.
25. M.F. Argüello, J.F. Bard, and G. Yu. A GRASP for aircraft routing in response to groundings and delays. *J. of Combinatorial Optimization*, 1:211–228, 1997.
26. M.F. Argüello, T.A. Feo, and O. Goldschmidt. Randomized methods for the number partitioning problem. *Computers and Operations Research*, 23:103–111, 1996.
27. M. Armony, J.C. Klincewicz, H. Luss, and M.B. Rosenwein. Design of stacked self-healing rings using a genetic algorithm. *J. of Heuristics*, 6:85–105, 2000.
28. J.E.C. Arroyo, P.S. Vieira, and D.S. Vianna. A GRASP algorithm for the multi-criteria minimum spanning tree problem. *Annals of Operations Research*, 159:125–133, 2008.
29. J.B. Atkinson. A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *J. of the Operatinal Research Society*, 49:700–708, 1998.
30. J.F. Bard. An analysis of a rail car unloading area for a consumer products manufacturer. *J. of the Operational Research Society*, 48:873–883, 1997.
31. J.F. Bard and T.A. Feo. Operations sequencing in discrete parts manufacturing. *Management Science*, 35:249–255, 1989.

32. J.F. Bard and T.A. Feo. An algorithm for the manufacturing equipment selection problem. *IIE Transactions*, 23:83–92, 1991.
33. J.F. Bard, T.A. Feo, and S. Holland. A GRASP for scheduling printed wiring board assembly. *IIE Transactions*, 28:155–165, 1996.
34. J.F. Bard, L. Huang, P. Jaillet, and M. Dror. A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Science*, 32:189–203, 1998.
35. E.B. Baum. Iterated descent: A better algorithm for local search in combinatorial optimization problems. Technical report, California Institute of Technology, 1986.
36. 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, 1987. American Institute of Physics Inc.
37. J. Baxter. Local optima avoidance in depot location. *J. of the Operational Research Society*, 32:815–819, 1981.
38. J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *J. of the Operational Research Society*, 41:1069–1072, 1990.
39. S. Binato, H. Faria Jr., and M.G.C. Resende. Greedy randomized adaptive path relinking. In J.P. Sousa, editor, *Proceedings of the IV Metaheuristics International Conference*, pages 393–397, 2001.
40. S. Binato, W.J. Hery, D. Loewenstern, and M.G.C. Resende. A GRASP for job shop scheduling. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 59–79. Kluwer Academic Publishers, 2002.
41. S. Binato and G.C. Oliveira. A reactive GRASP for transmission network expansion planning. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 81–100. Kluwer Academic Publishers, 2002.
42. S. Binato, G.C. Oliveira, and J.L. Araújo. A greedy randomized adaptive search procedure for transmission expansion planning. *IEEE Transactions on Power Systems*, 16:247–253, 2001.
43. M. Boudia, M.A.O. Louly, and C. Prins. A reactive GRASP and path relinking for a combined production-distribution problem. *Computers and Operations Research*, 34:3402–3419, 2007.
44. J.L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 271–278, Portland, 1996.
45. A.M. Campbell and B.W. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42:1–21, 2008.
46. 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.
47. C. Carreto and B. Baker. A GRASP interactive approach to the vehicle routing problem with backhauls. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 185–199. Kluwer Academic Publishers, 2002.
48. I. Charon and O. Hudry. The noising method: A new method for combinatorial optimization. *Operations Research Letters*, 14:133–137, 1993.
49. I. Charon and O. Hudry. The noising methods: A survey. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 245–261. Kluwer Academic Publishers, 2002.
50. R. Colomé and D. Serra. Consumer choice in competitive location models: Formulations and heuristics. *Papers in Regional Science*, 80:439–464, 2001.
51. C. Commander, C.A.S. Oliveira, P.M. Pardalos, and M.G.C. Resende. A GRASP heuristic for the cooperative communication problem in ad hoc networks. In *Proceedings of the VI Metaheuristics International Conference*, pages 225–330, 2005.
52. C.W. Commander, S.I. Butenko, P.M. Pardalos, and C.A.S. Oliveira. Reactive GRASP with path relinking for the broadcast scheduling problem. In *Proceedings of the 40th Annual International Telemetry Conference*, pages 792–800, 2004.
53. A. Corberán, R. Martí, and J.M. Sanchís. A GRASP heuristic for the mixed Chinese postman problem. *European J. of Operational Research*, 142:70–80, 2002.

54. G.L. Cravo, G.M. Ribeiro, and L.A. Nogueira Lorena. A greedy randomized adaptive search procedure for the point-feature cartographic label placement. *Computers and Geosciences*, 34:373–386, 2008.
55. V.-D. Cung, S.L. Martins, C.C. Ribeiro, and C. Roucairol. Strategies for the parallel implementation of metaheuristics. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publishers, 2002.
56. P. De, J.B. Ghosj, and C.E. Wells. Solving a generalized model for con due date assignment and sequencing. *International J. of Production Economics*, 34:179–185, 1994.
57. 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.
58. A.S. Deshpande and E. Triantaphyllou. A greedy randomized adaptive search procedure (GRASP) for inferring logical clauses from examples in polynomial time and some extensions. *Mathematical Computer Modelling*, 27:75–99, 1998.
59. A. Drexl and F. Salewski. Distribution requirements and compactness constraints in school timetabling. *European J. of Operational Research*, 102:193–214, 1997.
60. 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.
61. 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.
62. C.C. Ribeiro e 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.
63. H. Faria Jr., S. Binato, M.G.C. Resende, and D.J. Falcão. Transmission network design by a greedy randomized adaptive path relinking approach. *IEEE Transactions on Power Systems*, 20:43–49, 2005.
64. T.A. Feo and J.F. Bard. Flight scheduling and maintenance base planning. *Management Science*, 35:1415–1432, 1989.
65. T.A. Feo and J.F. Bard. The cutting path and tool selection problem in computer-aided process planning. *J. of Manufacturing Systems*, 8:17–26, 1989.
66. T.A. Feo, J.F. Bard, and S. Holland. Facility-wide planning and scheduling of printed wiring board assembly. *Operations Research*, 43:219–230, 1995.
67. T.A. Feo and J.L. González-Velarde. The intermodal trailer assignment problem: Models, algorithms, and heuristics. *Transportation Science*, 29:330–341, 1995.
68. 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.
69. T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *J. of Global Optimization*, 6:109–133, 1995.
70. 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.
71. T.A. Feo, K. Sarathy, and J. McGahan. A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers and Operations Research*, 23:881–895, 1996.
72. T.A. Feo, K. Venkatraman, and J.F. Bard. A GRASP for a difficult single machine scheduling problem. *Computers and Operations Research*, 18:635–643, 1991.
73. E. Fernández and R. Martí. GRASP for seam drawing in mosaicking of aerial photographic maps. *J. of Heuristics*, 5:181–197, 1999.
74. P. Festa. On some optimization problems in molecular biology. *Mathematical Bioscience*, 207:219–234, 2007.
75. 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*, 11:1–16, 2006.
76. P. Festa, P.M. Pardalos, and M.G.C. Resende. Algorithm 815: FORTRAN subroutines for computing approximate solution to feedback set problems using GRASP. *ACM Transactions on Mathematical Software*, 27:456–464, 2001.

77. 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.
78. 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.
79. P. Festa and M.G.C. Resende. An annotated bibliography of GRASP, Part I: Algorithms. *International Transactions in Operational Research*, 16:1–24, 2009.
80. P. Festa and M.G.C. Resende. An annotated bibliography of GRASP, Part II: Applications. *International Transactions in Operational Research*, 16:131–172, 2009.
81. P. Festa, M.G.C. Resende, P. Pardalos, and C.C. Ribeiro. GRASP and VNS for Max-Cut. In *Extended Abstracts of the Fourth Metaheuristics International Conference*, pages 371–376, Porto, July 2001.
82. C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS J. on Computing*, 11:198–204, 1999.
83. I. Foster and C. Kesselman, editors. *The GRID: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.
84. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. of Supercomputer Applications*, 15:200–222, 2001.
85. J.B. Ghosh. Computatinal aspects of the maximum diversity problem. *Operations Research Letters*, 19:175–181, 1996.
86. F. Glover. New ejection chain and alternating path methods for traveling salesman problems. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research: New Developments in Their Interfaces*, pages 449–509. Elsevier, 1992.
87. F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65:223–254, 1996.
88. 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 Academic Publishers, 1996.
89. 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 Academic Publishers, 2000.
90. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
91. F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.
92. 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.
93. F.C. Gomes, C.S. Oliveira, P.M. Pardalos, and M.G.C. Resende. Reactive GRASP with path relinking for channel assignment in mobile phone networks. In *Proceedings of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 60–67. ACM Press, 2001.
94. P.L. Hammer and D.J. Rader, Jr. Maximally disjoint solutions of the set covering problem. *J. of Heuristics*, 7:131–144, 2001.
95. B.T. Han and V.T. Raja. A GRASP heuristic for solving an extended capacitated concentrator location problem. *International J. of Information Technology and Decision Making*, 2:597–617, 2003.
96. P. Hansen and N. Mladenović. Developments of variable neighborhood search. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 415–439. Kluwer Academic Publishers, 2002.
97. J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.
98. K. Holmqvist, A. Migdalas, and P.M. Pardalos. Greedy randomized adaptive search for a location problem with economies of scale. In I.M. Bomze et al., editor, *Developments in Global Optimization*, pages 301–313. Kluwer Academic Publishers, 1997.

99. K. Holmqvist, A. Migdalas, and P.M. Pardalos. A GRASP algorithm for the single source uncapacitated minimum concave-cost network flow problem. In P.M. Pardalos and D.-Z. Du, editors, *Network design: Connectivity and facilities location*, volume 40 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 131–142. American Mathematical Society, 1998.
100. 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.
101. J.G. Klincewicz. Avoiding local optima in the *p*-hub location problem using tabu search and GRASP. *Annals of Operations Research*, 40:283–302, 1992.
102. J.G. Klincewicz and A. Rajan. Using GRASP to solve the component grouping problem. *Naval Research Logistics*, 41:893–912, 1994.
103. G. Kontoravdis and J.F. Bard. A GRASP for the vehicle routing problem with time windows. *ORSA J. on Computing*, 7:10–23, 1995.
104. M. Laguna, T.A. Feo, and H.C. Elrod. A greedy randomized adaptive search procedure for the two-partition problem. *Operations Research*, 42:677–687, 1994.
105. M. Laguna and J.L. González-Velarde. A search heuristic for just-in-time scheduling in parallel machines. *J. of Intelligent Manufacturing*, 2:253–260, 1991.
106. M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, 11:44–52, 1999.
107. M. Laguna and R. Martí. A GRASP for coloring sparse graphs. *Computaional Optimization and Applications*, 19:165–178, 2001.
108. Y. Li, P.M. Pardalos, and M.G.C. Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems*, volume 16 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 237–261. American Mathematical Society, 1994.
109. A. Lim, B. Rodrigues, and C. Wang. Two-machine flow shop problems with a single server. *J. of Scheduling*, 9:515–543, 2006.
110. A. Lim and F. Wang. A smoothed dynamic tabu search embedded GRASP for *m*-VRPTW. In *Proceedings of ICTAI 2004*, pages 704–708, 2004.
111. 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.
112. 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.
113. H.R. Lourenço and D. Serra. Adaptive approach heuristics for the generalized assignment problem. *Mathware and Soft Computing*, 9:209–234, 2002.
114. A.P. Lucena, C.C. Ribeiro, and A.C. Santos. A hybrid heuristic for the diameter constrained minimum spanning tree problem, 2007.
115. R. Martí. Arc crossing minimization in graphs with GRASP. *IEE Transactions*, 33:913–919, 2001.
116. R. Martí. Arc crossing minimization in graphs with GRASP. *IEEE Transactions*, 33:913–919, 2002.
117. R. Martí and V. Estruch. Incremental bipartite drawing problem. *Computers and Operations Research*, 28:1287–1298, 2001.
118. R. Martí and M. Laguna. Heuristics and meta-heuristics for 2-layer straight line crossing minimization. *Discrete Applied Mathematics*, 127:665–678, 2003.
119. O. Martin and S.W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.
120. O. Martin, S.W. Otto, and E.W. Felten. Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.

121. 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.
122. S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P. Pardalos. A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *J. of Global Optimization*, 17:267–283, 2000.
123. S.L. Martins, C.C. Ribeiro, and M.C. Souza. A parallel GRASP for the Steiner problem in graphs. In A. Ferreira and J. Rolim, editors, *Proceedings of IRREGULAR'98 – 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 285–297. Springer-Verlag, 1998.
124. T. Mavridou, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A GRASP for the biquadratic assignment problem. *European J. of Operational Research*, 105:613–621, 1998.
125. N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.
126. S.K. Monkman, D.J. Morrice, and J.F. Bard. A production scheduling heuristic for an electronics manufacturer with sequence-dependent setup costs. *European J. of Operational Research*, 187:1100–1114, 2008.
127. R.A. Murphey, P.M. Pardalos, and L.S. Pitsoulis. A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. In P.M. Pardalos and D.-Z. Du, editors, *Network design: Connectivity and facilities location*, volume 40 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 277–301. American Mathematical Society, 1998.
128. R.A. Murphey, P.M. Pardalos, and L.S. Pitsoulis. A parallel GRASP for the data association multidimensional assignment problem. In P.M. Pardalos, editor, *Parallel Processing of Discrete Problems*, volume 106 of *The IMA Volumes in Mathematics and Its Applications*, pages 159–180. Springer-Verlag, 1998.
129. 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.
130. 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*, volume 3059, pages 356–368. Springer, 2004.
131. I.H. Osman, B. Al-Ayoubi, and M. Barake. A greedy random adaptive search procedure for the weighted maximal planar graph problem. *Computers and Industrial Engineering*, 45:635–651, 2003.
132. J.A. Pacheco and S. Casado. Solving two location models with few facilities by using a hybrid heuristic: A real health resources case. *Computers and Operations Research*, 32:3075–3091, 2005.
133. P. M. Pardalos, T. Qian, and M. G. C. Resende. A greedy randomized adaptive search procedure for the feedback vertex set problem. *J. of Combinatorial Optimization*, 2:399–412, 1999.
134. P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP implementation for the quadratic assignment problem. In A. Ferreira and J. Rolim, editors, *Parallel Algorithms for Irregularly Structured Problems – Irregular'94*, pages 115–133. Kluwer Academic Publishers, 1995.
135. P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP for MAX-SAT problems. *Lecture Notes in Computer Science*, 1184:575–585, 1996.
136. P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. Algorithm 769: Fortran subroutines for approximate solution of sparse quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software*, 23:196–208, 1997.
137. R.A. Patterson, H. Pirkul, and E. Rolland. A memory adaptive reasoning technique for solving the capacitated minimum spanning tree problem. *J. of Heuristics*, 5:159–180, 1999.

138. E. Pinana, I. Plana, V. Campos, and R. Martí. GRASP and path relinking for the matrix bandwidth minimization. *European J. of Operational Research*, 153:200–210, 2004.
139. L.S. Pitsoulis, P.M. Pardalos, and D.W. Hearn. Approximate solutions to the turbine balancing problem. *European J. of Operational Research*, 130:147–155, 2001.
140. F. Poppe, M. Pickavet, P. Arijs, and P. Demeester. Design techniques for SDH mesh-restorable networks. In *Proceedings of the European Conference on Networks and Optical Communications, Volume 2: Core and ATM Networks*, pages 94–101, 1997.
141. M. Prais and C.C. Ribeiro. Parameter variation in GRASP implementations. In *Extended Abstracts of the Third Metaheuristics International Conference*, pages 375–380, Angra dos Reis, 1999.
142. M. Prais and C.C. Ribeiro. Parameter variation in GRASP procedures. *Investigación Operativa*, 9:1–20, 2000.
143. M. Prais and C.C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS J. on Computing*, 12:164–176, 2000.
144. M.C. Rangel, N.M.M. Abreu, and P.O. Boaventura Netto. GRASP in the QAP: An acceptance bound for initial solutions. *Pesquisa Operacional*, 20:45–58, 2000.
145. M.G. Ravetti, F.G. Nakamura, C.N. Meneses, M.G.C. Resende, G.R. Mateus, and P.M. Pardalos. Hybrid heuristics for the permutation flow shop problem. Technical report, AT&T Labs Research Technical Report, Florham Park, 2006.
146. M. Reghioui, C. Prins, and Nacima Labadi. GRASP with path relinking for the capacitated arc routing problem with time windows. In M. Giacobini et al., editor, *Applications of Evolutinary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 722–731. Springer, 2007.
147. L.I.P. Resende and M.G.C. Resende. A GRASP for frame relay permanent virtual circuit routing. In C.C. Ribeiro and P. Hansen, editors, *Extended Abstracts of the III Metaheuristics International Conference*, pages 397–401, Angra dos Reis, 1999.
148. M.G.C. Resende. Computing approximate solutions of the maximum covering problem using GRASP. *J. of Heuristics*, 4:161–171, 1998.
149. M.G.C. Resende and T.A. Feo. A GRASP for satisfiability. In D.S. Johnson and M.A. Trick, editors, *Cliques, Coloring, and Satisfiability: The Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 499–520. American Mathematical Society, 1996.
150. 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.
151. 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. Published online 28 May 2008, `doi:10.1016/j.cor.2008.05.011`.
152. M.G.C. Resende, P.M. Pardalos, and Y. Li. Algorithm 754: Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software*, 22:104–118, 1996.
153. 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.
154. 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.
155. M.G.C. Resende and C.C. Ribeiro. A GRASP for graph planarization. *Networks*, 29:173–189, 1997.
156. M.G.C. Resende and C.C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41:104–114, 2003.
157. 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.

158. M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the *p*-median problem. *J. of Heuristics*, 10:59–88, 2004.
159. 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.
160. C.C. Ribeiro. GRASP: Une métaheuristique gloutone et probabiliste. In J. Teghem and M. Pirlot, editors, *Optimisation approchée en recherche opérationnelle*, pages 153–176. Hermès, 2002.
161. C.C. Ribeiro and M.G.C. Resende. Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP. *ACM Transactions on Mathematical Software*, 25:342–352, 1999.
162. C.C. Ribeiro, C.D. Ribeiro, and R.S. Lanzelotte. Query optimization in distributed relational databases. *J. of Heuristics*, 3:5–23, 1997.
163. C.C. Ribeiro and I. Rosseti. Efficient parallel cooperative implementations of GRASP heuristics. *Parallel Computing*, 33:21–35, 2007.
164. 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.
165. C.C. Ribeiro, E. Uchoa, and R.F. Werneck. A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS J. on Computing*, 14:228–246, 2002.
166. C.C. Ribeiro and S. Urrutia. Heuristics for the mirrored traveling tournament problem. *European J. of Operational Research*, 179:775–787, 2007.
167. R.Z. Ríos-Mercado and J.F. Bard. Heuristics for the flow line problem with setup costs. *European J. of Operational Research*, pages 76–98, 1998.
168. R.Z. Ríos-Mercado and J.F. Bard. An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup costs. *J. of Heuristics*, 5:57–74, 1999.
169. A.J. Robertson. A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem. *Computational Optimization and Applications*, 19:145–164, 2001.
170. P.L. Rocha, M.G. Ravetti, and G.R. Mateus. The metaheuristic GRASP as an upper bound for a branch and bound algorithm in a scheduling problem with non-related parallel machines and sequence-dependent setup times. In *Proceedings of the 4th EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-Heuristica*, volume 1, pages 62–67, 2004.
171. M. Scaparra and R. Church. A GRASP and path relinking heuristic for rural road network development. *J. of Heuristics*, 11:89–108, 2005.
172. D. Sosnowska. Optimization of a simplified fleet assignment problem with metaheuristics: Simulated annealing and GRASP. In P.M. Pardalos, editor, *Approximation and complexity in numerical optimization*. Kluwer Academic Publishers, 2000.
173. M.C. 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. Souza, editors, *Metaheuristics: Computer Decision-Making*, pages 627–658. Kluwer Academic Publisher, 2004.
174. A. Srinivasan, K.G. Ramakrishnan, K. Kumaram, M. Aravamudam, and S. Naqvi. Optimal design of signaling networks for Internet telephony. In *IEEE INFOCOM 2000*, volume 2, pages 707–716, 2000.
175. H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.
176. T.L. Urban. Solution procedures for the dynamic facility layout problem. *Annals of Operations Research*, 76:323–342, 1998.
177. T.L. Urban, W.-C. Chiang, and R.A. Russel. The integrated machine allocation and layout problem. *International J. of Production Research*, 38:2913–2930, 2000.
178. D.L. Woodruff and E. Zemel. Hashing vectors for tabu search. *Annals of Operations Research*, 41:123–137, 1993.
179. J.Y. Xu and S.Y. Chiu. Effective heuristic procedure for a field technician scheduling problem. *J. of Heuristics*, 7:495–509, 2001.
180. J. Yen, M. Carlsson, M. Chang, J.M. Garcia, and H. Nguyen. Constraint solving for inkjet print mask design. *J. of Imaging Science and Technology*, 44:391–397, 2000.