Greedy Randomized Adaptive Search Procedures: Advances and Extensions

Mauricio G.C. Resende and Celso C. Ribeiro

Abstract – A greedy randomized adaptive search procedure (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, Lagrangean constructive heuristics and Lagrangean GRASP, 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. Restart strategies to speedup the search, hybridizations with other metaheuristics, 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

Mauricio G.C. Resende
Amazon.com and University of Washington, Seattle, WA USA, e-mail: mgcr@uw.edu

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

the set of feasible solutions F are defined for each specific problem. For instance, in the case of the traveling salesman problem, the ground set E is that of all edges connecting the cities to be visited, $f(S)$ is the sum of the costs of all edges in S, and F is formed by all edge subsets that determine a Hamiltonian cycle.

GRASP (Greedy Randomized Adaptive Search Procedure) [90, 91] 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 using a greedy randomized adaptive algorithm. If this solution is not feasible, then it is necessary to apply a repair procedure to achieve feasibility or to make a new attempt to build a feasible solution. 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 on greedy randomized adaptive search procedures are presented in [100, 101, 102, 217, 218, 224]. A first book on GRASP was published in 2016 by Resende and Ribeiro [220].

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)
1   Read_Input();
2   for $k = 1,\ldots,$Max_Iterations do
3       Solution ← Greedy_Randomized_Construction(Seed);
4       if Solution is not feasible then
5           Solution ← Repair(Solution);
6       end;
7       Solution ← Local_Search(Solution);
8       Update_Solution(Solution,Best_Solution);
9   end;
10  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 [121], 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.

A randomized greedy construction procedure is not always able to produce a feasible solution. It may be necessary to apply a repair procedure to the solution to achieve feasibility. Examples of repair procedures can be found in [81, 83, 169, 180].

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 typically used. The neighborhood search can 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 number of iterations. 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 feasibly inserted into the partial solution under construction 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 construction can 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 can be near-optimal or even 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 by 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 [213] 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 former 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 longer than the time taken by $\alpha = 0.9$ (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 [201] show that using a single fixed value for the RCL parameter α very often hinders finding a high-quality solution, which could be found if another value is used. They propose 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 depending on the quality of the solutions obtained along the search. In particular, computational experiments on the problem of traffic assignment in communication satellites [202] show that Reactive GRASP finds better solutions than the basic algorithm for many test instances. These results motivated the study of the behavior of GRASP with different strategies for the variation of the value of the RCL parameter α:

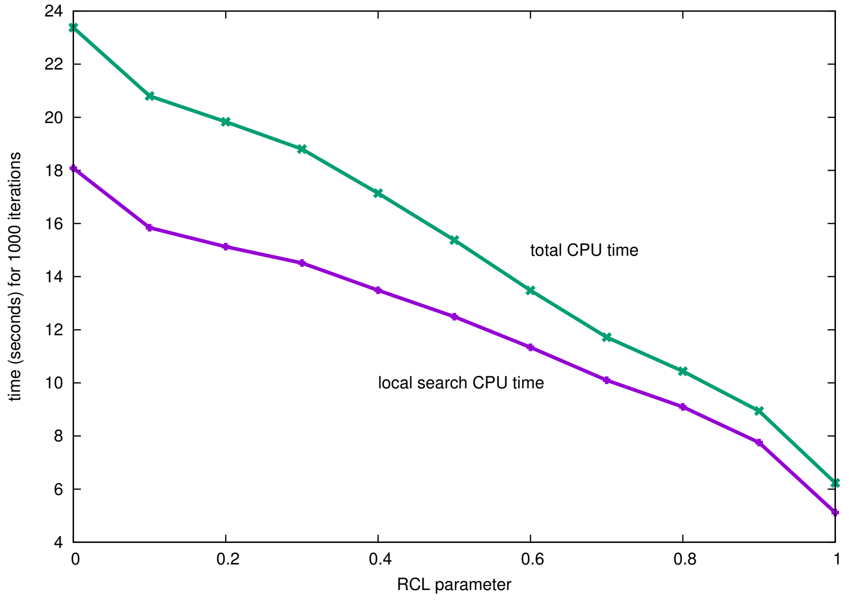


Fig. 5 Total CPU time and local search CPU time as a function of the RCL parameter α for a maximization problem (1000 repetitions for each value of α).

R: α self tuned with a Reactive GRASP procedure;
E: α randomly chosen from a uniform discrete probability distribution;
H: α randomly chosen from a decreasing non-uniform discrete probability distribution;
F: α fixed, close to the purely greedy choice value.

We summarize the results obtained by the experiments reported in [200, 201]. These four strategies are incorporated into the GRASP procedures developed for four different optimization problems: (P-1) matrix decomposition for traffic assignment in communication satellites [202]; (P-2) set covering [90]; (P-3) weighted MAX-SAT [213, 214]; and (P-4) graph planarization [215, 227]. 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 [201] 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 for 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 is 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, local search on partially constructed solutions, and Lagrangean GRASP heuristics.

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 [221], 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 randomly 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 [202]. Let $\Psi = \{\alpha_1, \dots, \alpha_m\}$ 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 $i = 1, \dots, m$. Furthermore, let z^* be the incumbent solution and let A_i be the average value of all solutions found using $\alpha = \alpha_i$, for $i = 1, \dots, m$. 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 [200, 201, 202], this approach has been used in power system transmission network planning [48], job shop scheduling [47], channel assignment in mobile phone networks [117], rural road network development [249], capacitated location [70], strip-packing [16], and a combined production-distribution problem [50].

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 [57, 58]. It adds more flexibility into the 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 [259], used as one of the main building blocks of the construction phase of the hybrid GRASP procedure proposed by Ribeiro et al. [234] 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. [54] for the prize-collecting Steiner tree problem, which makes use of the primal-dual algorithm of Goemans and Williamson [116] to build initial solutions using perturbed costs.

In the case of the GRASP for the prize-collecting Steiner tree problem described in [54], a new solution is built at each iteration using node prizes updated by a perturbation function, based on 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 [57, 58].

The cost perturbation methods used in the GRASP for the minimum Steiner tree problem described in [234] 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 over 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 [234] 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 [51], where a family of such probability distributions is introduced. They are based on the rank $r(e)$ assigned to each candidate element $e \in C$, 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 $r(e)$ denote the rank of element $e \in C$ and let $\texttt{bias}(r(e))$ be one of the bias functions defined above. Once these values have been evaluated for all elements in the candidate set C, the probability $\pi(e)$ of selecting element $e \in C$ is

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

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 [47]. The standard GRASP uses a random bias function.

3.5 Intelligent construction: memory and learning

Fleurent and Glover [105] observed that the basic GRASP does not use a 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 the *intensity function* $I(e)$ 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 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 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 [105] and Binato et al. [47].

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” [114]. Fleurent and Glover [105] 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. [47] after 40% and 80% of the construction moves were performed, as well as at the end of the construction phase.

3.7 Lagrangean GRASP heuristics

Lagrangean relaxation [45, 104] is a mathematical programming technique that can be used to provide lower bounds for minimization problems. Held and Karp [122, 123] were among the first to explore the use of the dual multipliers produced by Lagrangean relaxation to derive lower bounds, applying this idea in the context of the traveling salesman problem. Lagrangean heuristics further explore the use of different dual multipliers to generate feasible solutions. Beasley [43, 44] described a Lagrangean heuristic for set covering.

3.7.1 Lagrangean relaxation and subgradient optimization

Lagrangean relaxation can be used to provide lower bounds for combinatorial optimization problems. However, the primal solutions produced by the algorithms used to solve the Lagrangean dual problem are not necessarily feasible. Lagrangean heuristics exploit dual multipliers to generate primal feasible solutions.

Given a mathematical programming problem $\mathscr{P}$ formulated as

$$f^* = \min f(x) \tag{2}$$

$$g_i(x) \leq 0, \quad i = 1, \ldots, m, \tag{3}$$

$$x \in X, \tag{4}$$

its Lagrangean relaxation is obtained by associating dual multipliers $\lambda_i \in \mathbb{R}_+$ with each inequality (3), for $i = 1, \ldots, m$. This results in the following *Lagrangean relaxation problem LRP*(λ)

$$\min f'(x) = f(x) + \sum_{i=1}^{m} \lambda_i \cdot g_i(x) \tag{5}$$

$$x \in X, \tag{4}$$

whose optimal solution $x(\lambda)$ gives a lower bound $f'(x(\lambda))$ to the optimal value of the original problem $\mathscr{P}$ defined by (2) to (4). The best (dual) lower bound is given by the solution of the *Lagrangean dual problem* $\mathscr{D}$

$$f_{\mathscr{D}} = f'(x(\lambda^*)) = \max_{\lambda \in \mathbb{R}^m_+} f'(x(\lambda)). \tag{6}$$

Subgradient optimization is used to solve the dual problem $\mathscr{D}$ defined by (6). Subgradient algorithms start from any feasible set of dual multipliers, such as $\lambda_i = 0$, for $i = 1, \ldots, m$, and iteratively generate updated multipliers.

At any iteration q, let λ^q be the current vector of multipliers and let $x(\lambda^q)$ be an optimal solution to problem *LRP*(λ^q), whose optimal value is $f'(x(\lambda^q))$. Furthermore, let $\bar{f}$ be a known upper bound to the optimal value of problem $\mathscr{P}$. Additionally, let $g^q \in \mathbb{R}^m$ be a subgradient of $f'(x)$ at $x = x(\lambda^q)$, with $g_i^q = g_i(x(\lambda^q))$ for $i = 1, \ldots, m$. To update the Lagrangean multipliers, the algorithm makes use of a step size

$$d^q = \frac{\eta \cdot (\bar{f} - f'(x(\lambda^q)))}{\sum_{i=1}^{m} (g_i^q)^2}, \tag{7}$$

where $\eta \in (0, 2]$. Multipliers are then updated as

$$\lambda_i^{q+1} = \max\{0; \lambda_i^q - d^q \cdot g_i^q\}, \quad i = 1, \ldots, m, \tag{8}$$

and the subgradient algorithm proceeds to iteration $q+1$.

3.7.2 A template for Lagrangean heuristics

We describe next a template for Lagrangean heuristics that make use of the dual multipliers λ^q and of the optimal solution $x(\lambda^q)$ to each problem $LRP(\lambda^q)$ to build feasible solutions to the original problem $\mathscr{P}$ defined by (2) to (4). In the following, we assume that the objective function and all constraints are linear functions, i.e. $f(x) = \sum_{i=1}^n c_j x_j$ and $g_i(x) = \sum_{j=1}^n d_{ij} x_j - e_i$, for $i = 1, \ldots, m$.

Let $\mathscr{H}$ be a primal heuristic that builds a feasible solution x to $\mathscr{P}$, starting from the initial solution $x^0 = x(\lambda^q)$ at every iteration q of the subgradient algorithm. Heuristic $\mathscr{H}$ is first applied using the original costs c_j, i.e. using the cost function $f(x)$. In any subsequent iteration q of the subgradient algorithm, $\mathscr{H}$ uses either the Lagrangean reduced costs $c'_j = c_j - \sum_{i=1}^m \lambda_i^q d_{ij}$ or the complementary costs $\bar{c}_j = (1 - x_j(\lambda^q)) \cdot c_j$.

Let $x^{\mathscr{H},\gamma}$ be the solution obtained by heuristic $\mathscr{H}$, using a generic cost vector γ corresponding to either one of the above modified cost schemes or to the original cost vector. Its cost can be used to update the upper bound $\bar{f}$ to the optimal value of the original problem. This upper bound can be further improved by local search and is used to adjust the step size defined by equation (7).

Figure 6 shows the pseudo-code of a Lagrangean heuristic. Lines 1 to 4 initialize the upper and lower bounds, the iteration counter, and the dual multipliers. The iterations of the subgradient algorithm are performed along the loop defined in lines 5 to 24. The reduced costs are computed in line 6 and the Lagrangean relaxation problem is solved in line 7. In the first iteration of the Lagrangean heuristic, the original cost vector is assigned to γ in line 9, while in subsequent iterations a modified cost vector is assigned to γ in line 11. Heuristic $\mathscr{H}$ is applied in line 13 at the first iteration and after every H iterations thereafter (i.e., whenever the iteration counter q is a multiple of the input parameter H) to produce a feasible solution $x^{\mathscr{H},\gamma}$ to problem $\mathscr{P}$. If the cost of this solution is smaller than the current upper bound, then the best solution and its cost are updated in lines 14 to 18. If the lower bound $f'(x(\lambda^q))$ is greater than the current lower bound $f_{\mathscr{D}}$, then $f_{\mathscr{D}}$ is updated in line 19. Line 20 computes a subgradient at $x(\lambda^q)$ and line 21 computes the step size. The dual multipliers are updated in line 22 and the iteration counter is incremented in line 23. The best solution found and its cost are returned in line 24.

The strategy proposed by Held, Wolfe, and Crowder [124] is commonly used in the implementation of Lagrangean heuristics to update the dual multipliers from one iteration to the next. Beasley [44] reported as computationally useful the adjustment of components of the subgradients to zero whenever they do not effectively contribute to the update of the multipliers, i.e., arbitrarily setting $g_i^q = 0$ whenever $g_i^q > 0$ and $\lambda_i^q = 0$, for $i = 1, \ldots, m$.

```
procedure Lagrangean_Heuristic($H$)
1   $\bar{f} \leftarrow +\infty$;
2   $f_{\mathscr{D}} \leftarrow -\infty$;
3   $q \leftarrow 0$;
4   $\lambda_i^q \leftarrow 0,\ i = 1,\dots,m$;
5   repeat
6       Compute reduced costs: $c'_j \leftarrow c_j - \sum_{i=1}^m \lambda_i^q d_{ij},\ j = 1,\dots,n$;
7       Solve $LRP(\lambda^q)$ to obtain a solution $x(\lambda^q)$;
8       if $q = 0$ then
9           $\gamma \leftarrow c$;
10      else
11          Set $\gamma$ to the modified cost vector $c'$ or $\bar{c}$;
12      end-if;
13      if $q$ is a multiple of $H$ then apply heuristic $\mathscr{H}$ with cost vector $\gamma$ to obtain $x^{\mathscr{H},\gamma}$;
14      if $f(x^{\mathscr{H},\gamma}) < \bar{f}$
15      then do;
16          $x^* \leftarrow x^{\mathscr{H},\gamma}$;
17          $\bar{f} \leftarrow f(x^{\mathscr{H},\gamma})$;
18      end-if;
19      if $f'(x(\lambda^q)) > f_{\mathscr{D}}$ then $f_{\mathscr{D}} \leftarrow f'(x(\lambda^q))$;
20      Compute a subgradient: $g_i^q \leftarrow g_i(x(\lambda^q)),\ i = 1,\dots,m$;
21      Compute the step size: $d^q \leftarrow \eta \cdot (\bar{f} - f'(x(\lambda^q)))/\sum_{i=1}^m (g_i^q)^2$;
22      Update the dual multipliers: $\lambda_i^{q+1} \leftarrow \max\{0, \lambda_i^q - d^q g_i^q\},\ i = 1,\dots,m$;
23      $q \leftarrow q + 1$;
24  until stopping criterion satisfied;
25  return $x^*, f(x^*)$;
end Lagrangean_Heuristic.
```

Fig. 6 Pseudo-code of a template for a Lagrangean heuristic.

Different choices for the initial solution x^0, for the modified costs γ, and for the primal heuristic $\mathscr{H}$ itself lead to different variants of the above algorithm. The integer parameter H defines the frequency in which $\mathscr{H}$ is applied. The smaller the value of H, the greater the number of times $\mathscr{H}$ is applied. Therefore, the computation time increases as the value of H decreases. In particular, one should set $H = 1$ if the primal heuristic $\mathscr{H}$ is to be applied at every iteration.

3.7.3 Lagrangean GRASP

Pessoa, Resende, and Ribeiro [195, 196] proposed the hybridization of GRASP and Lagrangean relaxation leading to the Lagrangean GRASP heuristic described below. Different choices for the primal heuristic $\mathscr{H}$ in the template of the algorithm in Figure 6 lead to distinct Lagrangean heuristics. We consider two variants: the first makes use of a greedy algorithm with local search, while in the second a GRASP with path-relinking (see Section 4) is used.

Greedy heuristic: This heuristic greedily repairs the solution $x(\lambda^q)$ produced in line 7 of the Lagrangean heuristic described in Figure 6 to make it feasible for problem $\mathscr{P}$. It makes use of the modified costs (c' or $\bar{c}$). Local search can be applied to the resulting solution, using the original cost vector c. We refer to this approach as a *greedy Lagrangean heuristic* (GLH).

GRASP heuristic: Instead of simply performing one construction step followed by local search, as GLH does, this variant applies a GRASP heuristic to repair the solution $x(\lambda^q)$ produced in line 7 of the Lagrangean heuristic to make it feasible for problem $\mathscr{P}$.

Although the GRASP heuristic produces better solutions than the greedy heuristic, the greedy heuristic is much faster. To appropriately address this trade-off, we adapt line 10 of Figure 6 to use the GRASP heuristic with probability β and the greedy heuristic with probability $1-\beta$, where β is a parameter of the algorithm.

We note that this strategy involves three main parameters: the number H of iterations after which the basic heuristic is always applied, the number Q of iterations performed by the GRASP heuristic when it is chosen as the primal heuristic, and the probability β of choosing the GRASP heuristic as $\mathscr{H}$. We shall refer to the Lagrangean heuristic that uses this hybrid strategy as LAGRASP(β,H,Q).

We next summarize computational results obtained for 135 instances of the set k-covering problem. These instances have up to 400 constraints and 4000 binary variables. The set k-covering, or set multi-covering, problem is an extension of the classical set covering problem, in which each element is required to be covered at least k times. The problem finds applications in the design of communication networks and in computational biology.

The first experiment with the GRASP Lagrangean heuristic established the relationship between running times and solution quality for different parameter settings. Parameter β, the probability of GRASP being applied as the heuristic $\mathscr{H}$, was set to 0, 0.25, 0.50, 0.75, and 1. Parameter H, the number of iterations between successive calls to the heuristic $\mathscr{H}$, was set to 1, 5, 10, and 50. Parameter Q, the number of iterations carried out by the GRASP heuristic, was set to 1, 5, 10, and 50. By combining some of these parameter values, 68 variants of the hybrid LAGRASP(β,H,Q) heuristic were created. Each variant was applied eight times to a subset of 21 instances, with different initial seeds being given to the random number generator.

The plot in Figure 7 summarizes the results for all variants evaluated, displaying points whose coordinates are the values of the average deviation from the best known solution value and the total time in seconds for processing the eight runs on all instances, for each combination of parameter values. Eight variants of special interest are identified and labeled with the corresponding parameters β, H, and Q, in this order. These variants correspond to selected Pareto points in the plot in Figure 7. Setting $\beta = 0$ and $H = 1$ corresponds to the greedy Lagrangean heuristic (GLH) or, equivalently, to LAGRASP(0,1,-), whose average deviation (in percentage) from the best value amounts to 0.12% in 4,859.16 seconds of total running time. Table 3 shows the average deviation from the best known solution value and the total time for each of the eight selected variants.

In another experiment, all 135 test instances were considered for the comparison of the above selected eight variants of LAGRASP. Table 4 summarizes the results

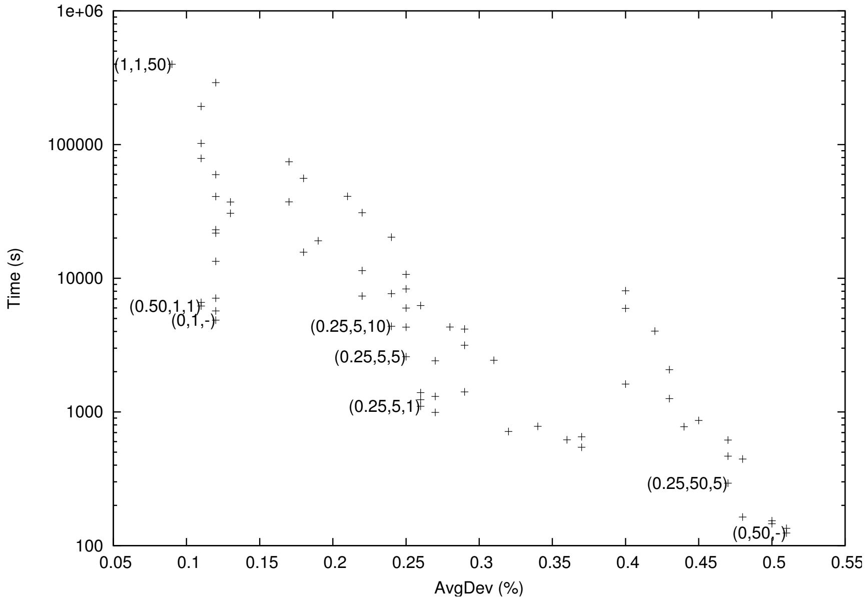


Fig. 7 Average deviation from the best value and total running time for 68 different variants of LAGRASP on a reduced set of 21 instances of the set k-covering problem: each point represents a unique combination of parameters β, H, and Q.

Table 3 Summary of the numerical results obtained with the selected variants of the GRASP Lagrangean heuristic on a reduced set of 21 instances of the set k-covering problem. These values correspond to the coordinates of the selected variants in Figure 7. The total time is given in seconds.

Heuristic	Average deviation	Total time (s)
LAGRASP(1,1,50)	0.09 %	399,101.14
LAGRASP(0.50,1,1)	0.11 %	6,198.46
LAGRASP(0,1,-)	0.12 %	4,859.16
LAGRASP(0.25,5,10)	0.24 %	4,373.56
LAGRASP(0.25,5,5)	0.25 %	2,589.79
LAGRASP(0.25,5,1)	0.26 %	1,101.64
LAGRASP(0.25,50,5)	0.47 %	292.95
LAGRASP(0,50,-)	0.51 %	124.26

obtained by the eight selected variants. It shows that LAGRASP(1,1,50) found the best solutions, with an average deviation from the best values amounting to 0.079%. It also found the best known solutions in 365 runs (each variant was run eight times on each instance), again with the best performance when the eight variants are evaluated side by side, although its running times are the largest. On the other hand, the smallest running times were observed for LAGRASP(0,50,-), which was over 3000 times faster than LAGRASP(1,1,50) but found the worst-quality solutions among the eight variants considered.

Table 4 Summary of the numerical results obtained with the selected variants of the GRASP Lagrangean heuristic on the full set of 135 instances of the set *k*-covering problem. The total time is given in seconds.

Heuristic	Average deviation	Hits	Total time (s)
LAGRASP(1,1,50)	0.079 %	365	1,803,283.64
LAGRASP(0.50,1,1)	0.134 %	242	30,489.17
LAGRASP(0,1,-)	0.135 %	238	24,274.72
LAGRASP(0.25,5,10)	0.235 %	168	22,475.54
LAGRASP(0.25,5,5)	0.247 %	163	11,263.80
LAGRASP(0.25,5,1)	0.249 %	164	5,347.78
LAGRASP(0.25,50,5)	0.442 %	100	1,553.35
LAGRASP(0,50,-)	0.439 %	97	569.30

Figure 8 illustrates the merit of the proposed approach for one of the test instances. We first observe that all variants reach the same lower bounds, as expected, since they depend exclusively on the common subgradient algorithm. However, as the lower bound appears to stabilize, the upper bound obtained by GLH (LAGRASP(0,1,-) also seems to freeze. On the other hand, the other variants continue to make improvements by discovering better upper bounds, since the randomized GRASP construction helps them to escape from locally optimal solutions and find new, improved upper bounds.

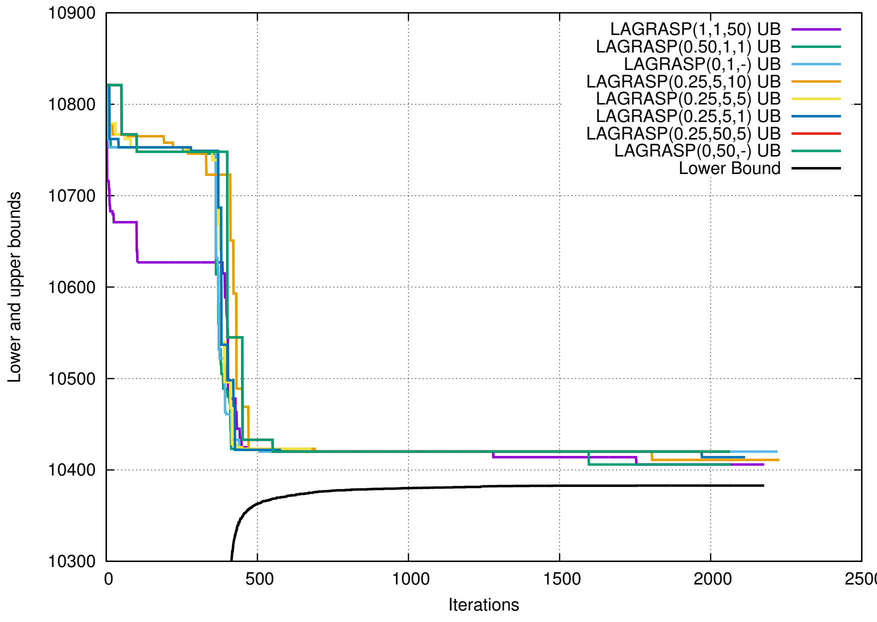


Fig. 8 Evolution of lower and upper bounds over the iterations for different variants of LAGRASP. The number of iterations taken by each LAGRASP variant depends on the step-size, which in turn depends on the upper bounds produced by each heuristic.

Finally, we provide a comparison between GRASP with backward path-relinking and the LAGRASP variants on all 135 test instances when the same time limits are used to stop all heuristics. Eight runs were performed for each heuristic and each instance, using different initial seeds for the random number generator. Each heuristic was run a total of $(8 \times 135 =)$ 1080 times. The results in Table 5 show that all variants of LAGRASP outperformed GRASP with backward path-relinking and were able to find solutions whose costs are very close to or as good as the best known solution values, while GRASP with backward path-relinking found solutions whose costs are on average 4.05% larger than the best known solution values.

Table 5 Summary of results for the best variants of LAGRASP and GRASP.

Heuristic	Average deviation	Hits
LAGRASP(1,1,50)	3.30 %	0
LAGRASP(0.50,1,1)	0.35 %	171
LAGRASP(0,1,-)	0.35 %	173
LAGRASP(0.25,5,10)	0.45 %	138
LAGRASP(0.25,5,5)	0.45 %	143
LAGRASP(0.25,5,1)	0.46 %	137
LAGRASP(0.25,50,5)	0.65 %	97
LAGRASP(0,50,-)	0.65 %	93
GRASP with backward path-relinking	4.05 %	0

Figure 9 displays for one test instance the typical behavior of these heuristics. As opposed to the GRASP with path-relinking, the Lagrangean heuristics are able to escape from local optima for longer and keep on improving the solutions to obtain the best results.

We note that an important feature of Lagrangean heuristics is that they provide not only a feasible solution (which gives an upper bound, in the case of a minimization problem), but also a lower bound that may be used to give an estimate of the optimality gap that may be considered as a stopping criterion.

4 Path-relinking

The LAGRASP heuristics presented in Section 3.7.3 made use of 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 [111] as an intensification strategy to explore trajectories connecting elite solutions obtained by tabu search or scatter search [112, 114, 115].

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 $s \in S$.

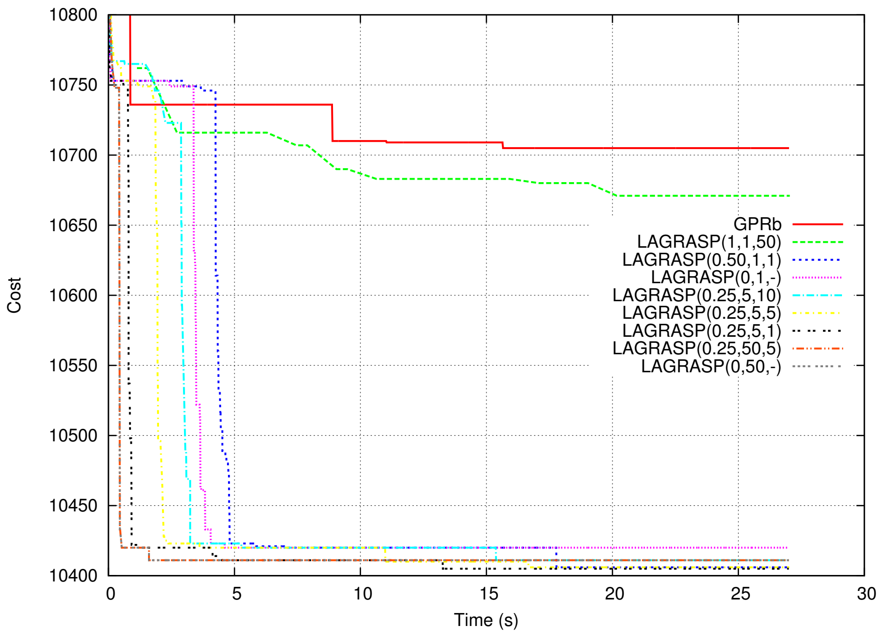


Fig. 9 Evolution of solution costs with time for the best variants of LAGRASP and GRASP with backward path-relinking (GPRb).

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 best solution 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 candidates to follow s 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 [147]. It was followed by several extensions, improvements, and successful applications [6, 7, 22, 54, 97, 183, 206, 217, 221, 222, 229, 234, 249]. A survey of GRASP with path-relinking can be found in [218].

Enhancing GRASP with path-relinking almost always improves the performance of the heuristic. As an illustration, Figure 10 shows time-to-target plots [9, 10, 230, 231, 232] 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.

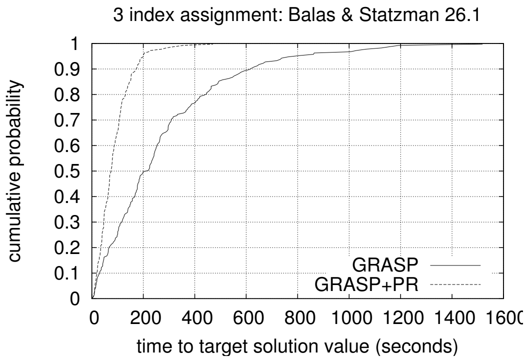
GRASP with path-relinking makes use of an *elite set* to collect a diverse pool of high-quality solutions found during the search. This pool is limited in size, i.e. it can have at most `Max_Elite` solutions. Several schemes have been proposed for the implementation of path-relinking, which may be applied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

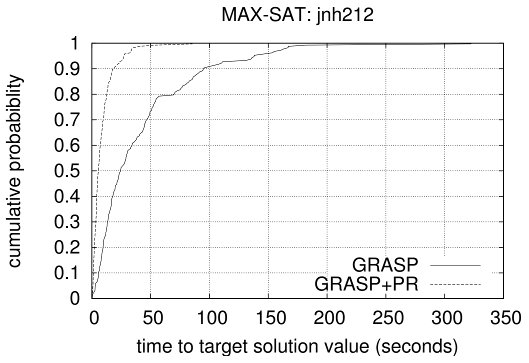
- an intensification strategy, between each local optimum obtained after the local search phase and one or more elite solutions;
- a post-optimization step, between every pair of elite solutions;
- an intensification strategy, periodically (after a fixed number of GRASP iterations since the last intensification 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.

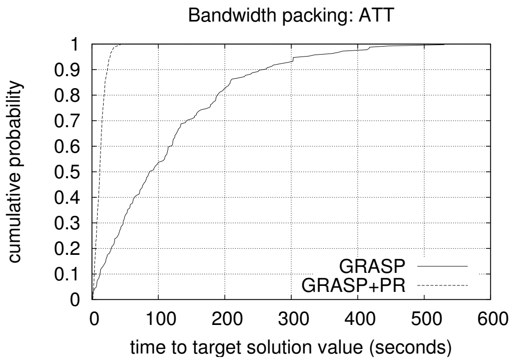
The pool of elite solutions is initially empty. Each locally optimal solution obtained by local search and each solution resulting from path-relinking is considered as a candidate to be inserted into the pool. If the pool is not yet full, the candidate is simply added to the pool. Otherwise, if the candidate is better than the incumbent (best solution found so far), it replaces an element of the pool. In case the candidate is better than the worst element 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 [221].

Given a local optimum s_1 produced at the end of a GRASP iteration, we need to select at random a solution s_2 from the pool to apply path-relinking between s_1 and s_2. 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 0-1 indicator vectors, we should favor pairs of solutions that are far from each other, based on their Hamming distance (i.e., the number of components that take on different values in each solution). A strategy introduced in Resende and Werneck [221] is to select a pool element s_2 at random with a 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 with 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







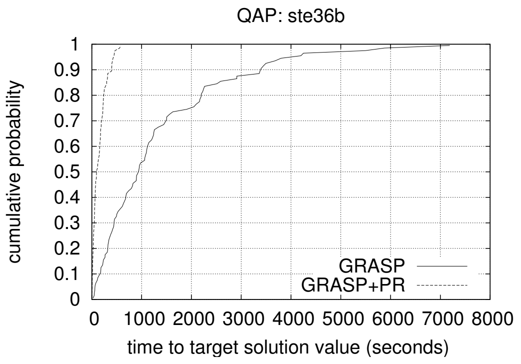


Fig. 10 Time to target plots comparing running times of pure GRASP and GRASP with path-relinking on four instances of distinct problem types: three index assignment, maximum satisfiability, bandwidth packing, and quadratic assignment.

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.

```
procedure GRASP+PR(Seed);
1    Set pool of elite solutions $\mathcal{E} \leftarrow \varnothing$;
2    Set best solution value $f^* \leftarrow \infty$;
3    while stopping criterion not satisfied do
4        Solution ← Greedy_Randomized_Construction(Seed);
5        if Solution is not feasible then
6            Solution ← Repair(Solution);
7        end-if;
8        Solution ← Local_Search(Solution);
9        if $|\mathcal{E}| > 0$ then
10           Select an elite solution Solution' at random from $\mathcal{E}$;
11           Solution ← PR(Solution,Solution');
12       end-if;
13       if $f$(Solution) $< f^*$ then
14           Best_Solution ← Solution;
15           $f^* \leftarrow f(S)$;
16       end-if;
17       Update the pool of elite solutions $\mathcal{E}$ with Solution;
18   end-while;
19   return Best_Solution;
end GRASP+PR.
```

Fig. 11 Pseudo-code of a template of a GRASP with path-relinking for a minimization problem.

The pseudo-code shown in Figure 11 summarizes the steps of a GRASP with path-relinking for a minimization problem. The pseudo-code follows the structure of the basic GRASP algorithm in Figure 1. Lines 1 and 2 initialize the pool of elite solutions and the best solution value, respectively. Path-relinking is performed in line 11 between the solution `Solution` obtained at the end of the local search phase (line 8) and a solution `Solution′` randomly selected from the pool of elite solutions $\mathcal{E}$ (line 10). Procedure `PR(Solution,Solution′)` could make use, for example, of any variant of a pure or combined path-relinking strategy. The best overall solution found `Best_Solution` is returned in line 19 after the stopping criterion is satisfied.

Several alternatives have been considered and combined in recent implementations of path-relinking. These include forward, backward, back and forward, mixed, truncated, greedy randomized adaptive, evolutionary, and external path-relinking. All these alternatives, which are described in the following, 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 [147].

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 [217]. 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, 217] 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, 217], 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 [111] and was first implemented and tested by Ribeiro and Rosseti [229], where it was shown to outperform forward, backward, and back and forward path-relinking. Figure 12 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.

4.5 Truncated path-relinking

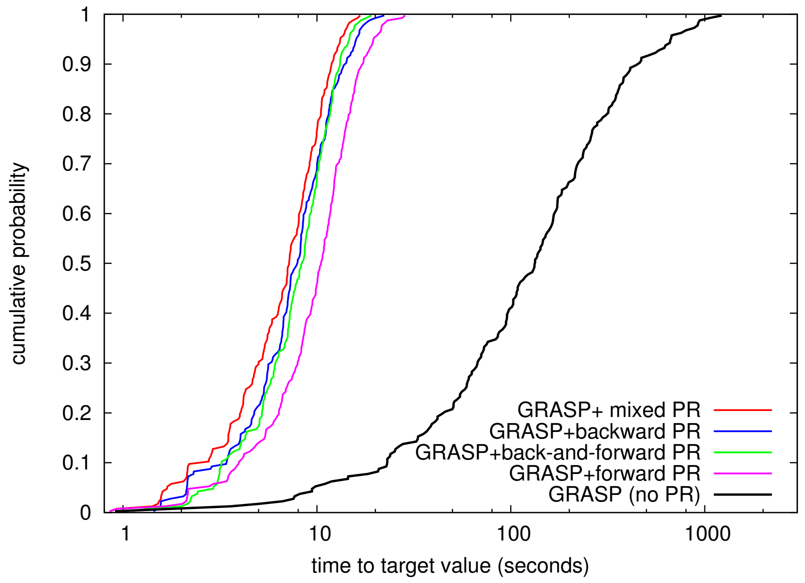
Since good-quality solutions tend to be near other good-quality solutions, one would expect to find the best solutions with path-relinking near the initial or guiding solution. Indeed, Resende et al. [211] showed that this is the case for instances of the max-min diversity problem, as shown in Figure 13. 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 [22, 211].

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. [46], 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 [22, 85, 211].



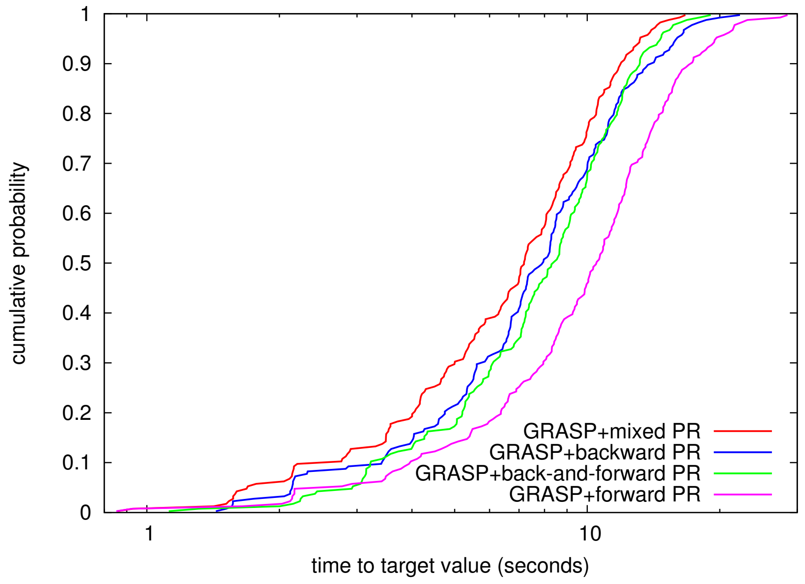


Fig. 12 Time-to-target plots for pure GRASP and four variants of GRASP with path-relinking (forward, backward, back and forward, and mixed) on an instance of the 2-path network design problem.

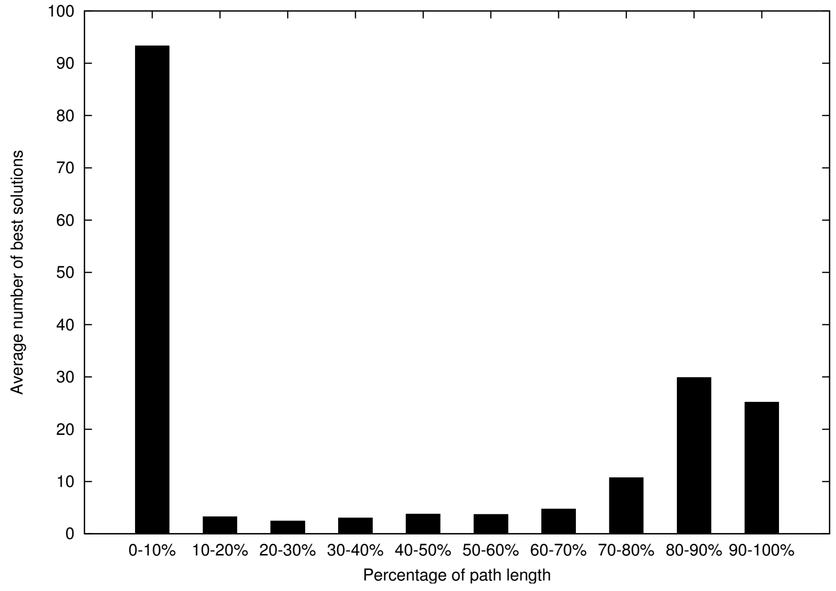


Fig. 13 Average number of best solutions found at different depths of the path from the initial solution to the guiding solution on instances of the max-min diversity problem.

4.7 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 [221, 222] 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 are 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. [211] 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.

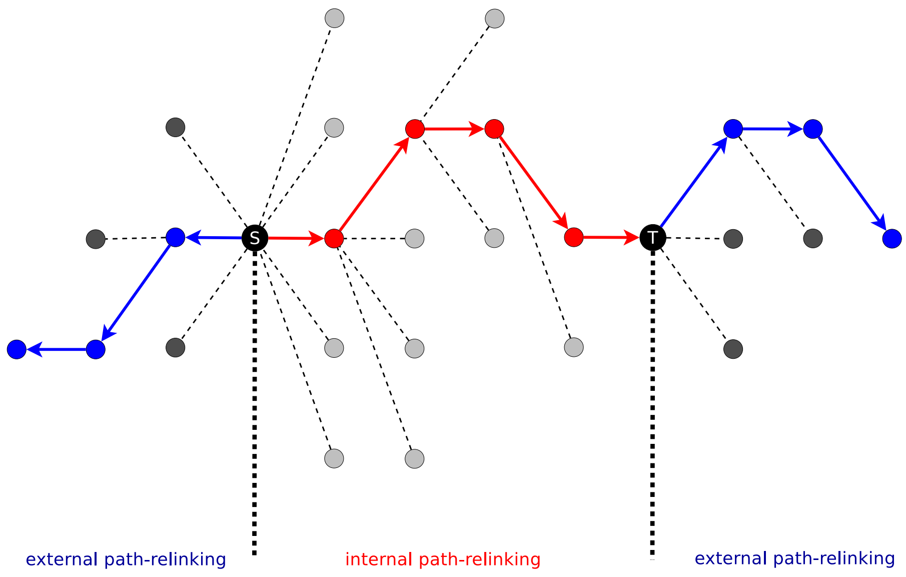


Fig. 14 An internal path (red arcs, red nodes) from solution S to solution T and two external (blue arcs, blue nodes) paths, one emanating from solution S and the other from solution T. These paths are produced by internal and external path-relinking.

Andrade and Resende [21] 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 may be relinked more than once in different calls to the intensification process, greedy randomized adaptive path-relinking was used.

Resende et al. [211] 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.

4.8 External path-relinking and diversification

So far in this section, we have considered variants of path-relinking in which a path in the search space graph connects two feasible solutions by progressively introducing in one of them (the initial solution) attributes of the other (the guiding solution). Since attributes common to both solutions are not changed and all solutions visited belong to a path between the two solutions, we may also refer to this type of path-relinking as *internal path-relinking*.

External path-relinking extends any path connecting two feasible solutions S and T beyond its extremities. To extend such a path beyond S, attributes not present in either S or T are introduced in S. Symmetrically, to extend it beyond T, attributes not present in either S or T are introduced in T. In its greedy variant, all moves are evaluated and the solution chosen to be next in the path is one with best cost or, in case they are all infeasible, the one with least infeasibility. In either direction, the procedure stops when all attributes that do not appear in either S or T have been tested for extending the path. Once both paths are complete, local search may be applied to the best solution in each of them. The best of the two local minima is returned as the solution produced by the external path-relinking procedure.

Figure 14 illustrates internal and external path-relinking. The path with red nodes and edges is the one resulting from internal path-relinking applied with S as the initial solution and T as the guiding solution. We observe that the orientation introduced by the arcs in this path is due only to the choice of the initial and guiding solutions. If the roles of solutions S and T were interchanged, it could have been computed and generated in the reverse direction. The same figure also illustrates two paths obtained by external path-relinking, one emanating from S and the other from T, both represented with blue nodes and edges. The orientations of the arcs in each of these paths indicate that they necessarily emanate from either solution S or T.

To conclude, we establish a parallel between internal and external path-relinking. Since internal path-relinking works by fixing all attributes common to the initial and guiding solutions and searches for paths between them satisfying this property, it is clearly an intensification strategy. Contrarily, external path-relinking progressively removes common attributes and replaces them by others that do not appear in either one of the initial or guiding solution. Therefore, it can be seen as a diversification strategy which produces solutions increasingly farther from both the initial and the guiding solutions. External path-relinking becomes therefore a tool for search diversification.

External path-relinking was introduced by Glover [113] and first applied by Duarte et al. [82] in a heuristic for differential dispersion minimization.

5 Restart strategies

Figure 15 shows a typical iteration count distribution for a GRASP with path-relinking. Observe in this example that for most of the independent runs whose iteration counts make up the plot, the algorithm finds a target solution in relatively few iterations: about 25% of the runs take at most 101 iterations; about 50% take at most 192 iterations; and about 75% take at most 345. However, some runs take much longer: 10% take over 1000 iterations; 5% over 2000; and 2% over 9715 iterations. The longest run took 11607 iterations to find a solution at least as good as the target. These long tails contribute to a large average iteration count as well as to a high standard deviation. This section proposes strategies to reduce the tail of the distribution, consequently reducing the average iteration count and its standard deviation.

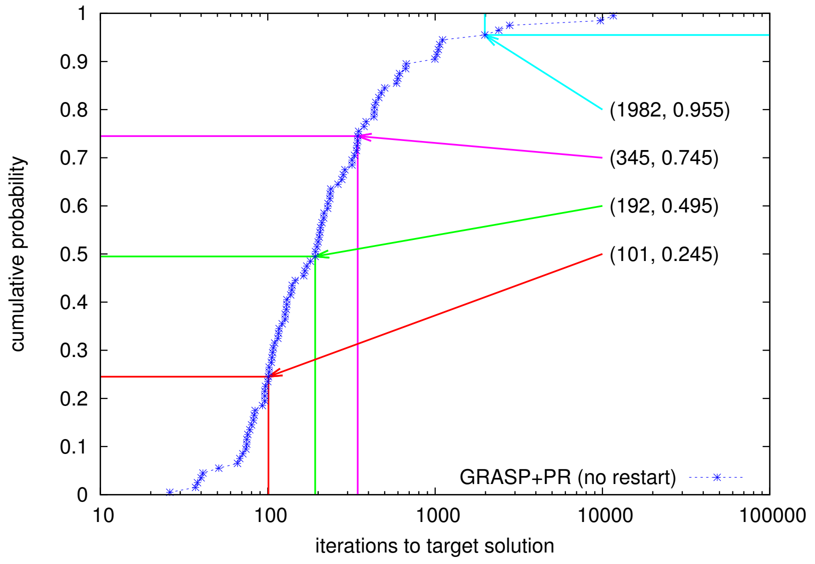


Fig. 15 Typical iteration count distribution of GRASP with path-relinking.

Consider again the distribution in Figure 15. The distribution shows that each run will take over 345 iterations with a probability of about 25%. Therefore, any time the algorithm is restarted, the probability that the new run will take over 345 iterations is also about 25%. By restarting the algorithm after 345 iterations, the new run will take more than 345 iterations with probability of also about 25%. Therefore, the probability that the algorithm will be still running after $345+345 = 690$ iterations is the probability that it takes more than 345 iterations multiplied by the probability that it takes more than 690 iterations given that it took more than 345 iterations, i.e., about $(1/4) \times (1/4) = (1/4)^2$. It follows by induction that the probability that the algorithm will still be running after k periods of 345 iterations is $1/(4^k)$. In this example, the probability that the algorithm will be running after 1725 iterations will be about 0.1%, i.e., much less than the 5% probability that the algorithm will take over 2000 iterations without restart.

A *restart strategy* is defined as an infinite sequence of time intervals $\tau_1, \tau_2, \tau_3, \ldots$ which define epochs $\tau_1, \tau_1+\tau_2, \tau_1+\tau_2+\tau_3, \ldots$ when the algorithm is restarted from scratch. It can be shown that the optimal restart strategy uses $\tau_1 = \tau_2 = \cdots = \tau^*$, where τ^* is some (unknown) constant. Strategies for speeding up stochastic local search algorithms using restarts were first proposed by Luby, Sinclair and Zuckerman [156], where they proved the existence of an optimal restart strategy. Restart strategies in metaheuristics have been addressed in [67, 138, 182, 187, 250]. Further work on restart strategies can be found in [251, 252].

Implementing the optimal strategy may be difficult in practice because it requires the constant value τ^*. Runtimes can vary greatly for different combinations of algorithm, instance, and solution quality sought. Since usually one has no prior information about the runtime distribution of the stochastic search algorithm for the optimization problem under consideration, one runs the risk of choosing a value of τ^* that is either too small or too large. On the one hand, a value that is too small can cause the restart variant of the algorithm to take much longer to converge than a no-restart variant. On the other hand, a value that is too large may never lead to a restart, causing the restart-variant of the algorithm to take as long to converge as the no-restart variant. Figure 16 illustrates the restart strategies with time-to-target plots for the maximum cut instance *G12* [125] on an 800-node graph with edge density of 0.63% with target solution value 554 for $\tau = 6$, 9, 12, 18, 24, 30, and 42 seconds. For each value of τ, 100 independent runs of a GRASP with path-relinking with restarts were performed. The variant with $\tau = \infty$ corresponds to the heuristic without restart. The figure shows that, for some values of τ, the resulting heuristic outperformed its counterpart with no restart by a large margin.

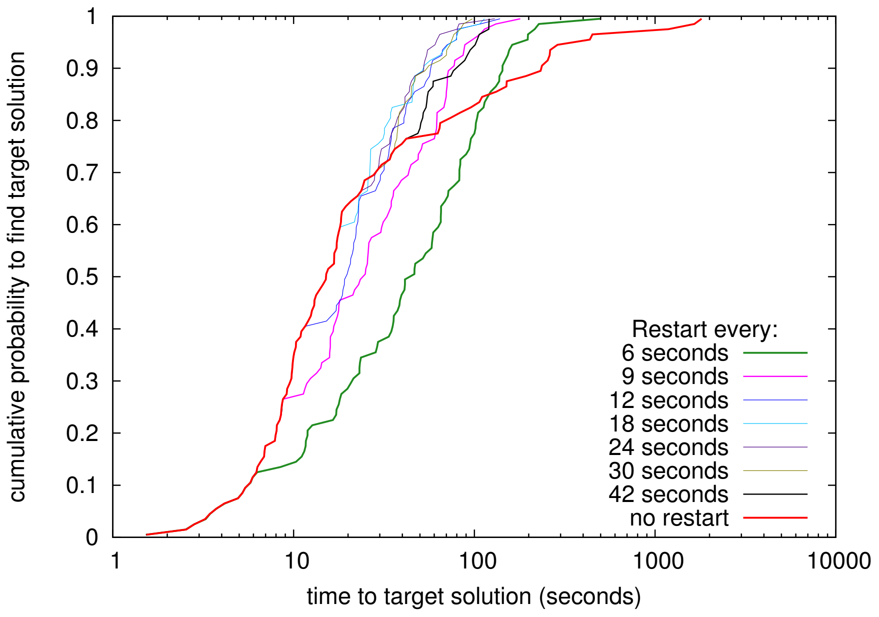


Fig. 16 Time-to-target plot for target solution value of 554 for a GRASP with path-linking with restart on the maximum cut instance *G12* using different values of τ.

In GRASP with path-relinking, the number of iterations between improvements of the incumbent solution tends to vary less than the runtimes for different combinations of instance and solution quality sought. If one takes this into account, a simple and effective restart strategy for GRASP with path-relinking is to keep track of the last iteration when the incumbent solution was improved and restart the GRASP with path-relinking if κ iterations have gone by without improvement. We shall call such a strategy restart(κ). A restart consists in saving the incumbent and emptying out the elite set.

```
procedure GRASP+PR+Restarts(Seed);
1   Set pool of elite solutions $\mathcal{E} \leftarrow \varnothing$;
2   Set best solution value $f^* \leftarrow \infty$;
3   LastImprov $\leftarrow 0$;
4   CurrentIter $\leftarrow 0$;
5   while stopping criterion not satisfied do
6       CurrentIter $\leftarrow$ CurrentIter $+ 1$;
7       Solution $\leftarrow$ Greedy_Randomized_Construction(Seed);
8       if Solution is not feasible then
9           Solution $\leftarrow$ Repair(Solution);
10      end-if;
11      Solution $\leftarrow$ Local_Search(Solution);
12      if $|\mathcal{E}| > 0$ then
13          Select an elite solution Solution' at random from $\mathcal{E}$;
14          Solution $\leftarrow$ forward-PR(Solution,Solution');
15      end-if;
16      if $f($Solution$) < f^*$ then
17          Best_Solution $\leftarrow$ Solution;
18          $f^* \leftarrow f(S)$;
19          LastImprov $\leftarrow$ CurrentIter;
20          end-if;
21      if CurrentIter $-$ LastImprov $> \kappa$ then
22          $\mathcal{E} \leftarrow \varnothing$;
23          LastImprov $\leftarrow$ CurrentIter;
24      else
25          Update the pool of elite solutions $\mathcal{E}$ with Solution;
26      end-if;
27      end-while;
28  return Best_Solution;
end GRASP+PR+Restarts.
```

Fig. 17 Pseudo-code of a template of a GRASP with path-relinking with restarts for a minimization problem.

The pseudo-code shown in Figure 17 summarizes the steps of a GRASP with path-relinking using the restart(κ) strategy for a minimization problem. The algorithm keeps track of the current iteration (CurrentIter), as well as of the last iteration when an improving solution was found (LastImprov). If an improving solution is detected in line 16, then this solution and its cost are saved in lines 17 and 18, respectively, and the iteration of the last improvement is set to the current iteration in line 19. If, in line 21, it is determined that more than κ iterations have gone by since the last improvement of the incumbent, then a restart is triggered, emptying out the elite set in line 22 and resetting the iteration of the last improvement to the current iteration in line 23. If restart is not triggered, then in line 25 the current solution is tested for inclusion in the elite set and the set is updated if it is accepted. The best overall solution found `Best_Solution` is returned in line 28 after the stopping criterion is satisfied.

As an illustration of the use of the restart(κ) strategy within a GRASP with path-relinking, consider the maximum cut instance *G12*. For the values $\kappa = 50$, 100, 200, 300, 500, 1000, 2000, and 5000, the heuristic was run independently 100 times, and was stopped when a cut of weight 554 or higher was found. A strategy without restarts was also implemented. Figures 18 and 19, as well as Table 6, summarize these runs, showing the average time to target solution as a function of the value of κ and the time-to-target plots for different values of κ. These figures illustrate well the effect on running time of selecting a value of κ that is either too small ($\kappa = 50, 100$) or too large ($\kappa = 2000, 5000$). They further show that there is a wide range of κ values (κ = 200, 300, 500, 1000) that result in lower runtimes when compared to the strategy without restarts.

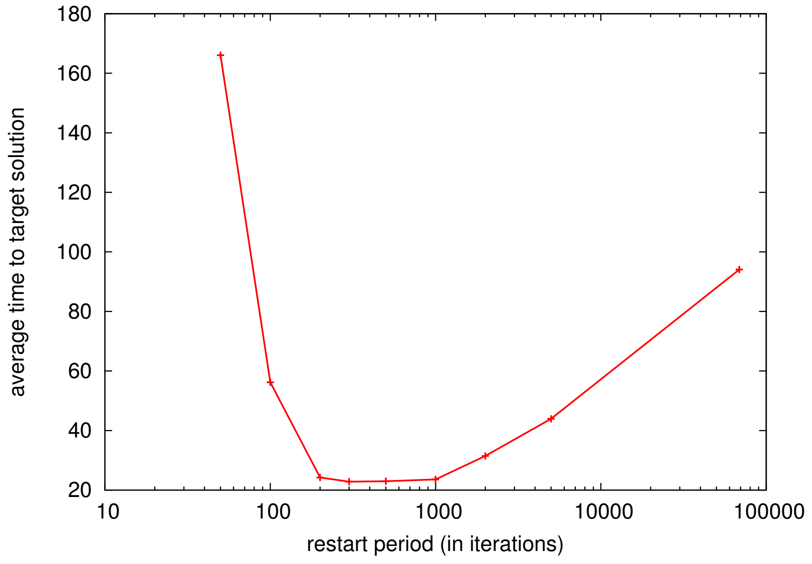


Fig. 18 Average time to target solution for maximum cut instance *G12* using different values of κ. All runs of all strategies have found a solution at least as good as the target value of 554.

Figure 20 further illustrates the behavior of the restart(100), restart(500), and restart(1000) strategies for the previous example, when compared with the strategy without restarts on the same maximum cut instance *G12*. However, in this figure, for each strategy, we plot the number of iterations to the target solution value. It is interesting to note that, as expected, each strategy restart(κ) behaves exactly like the strategy without restarts for the κ first iterations, for $\kappa = 100, 500, 1000$. After this point, each trajectory deviates from that of the strategy without restarts. Among these strategies, restart(500) is the one with the best performance.

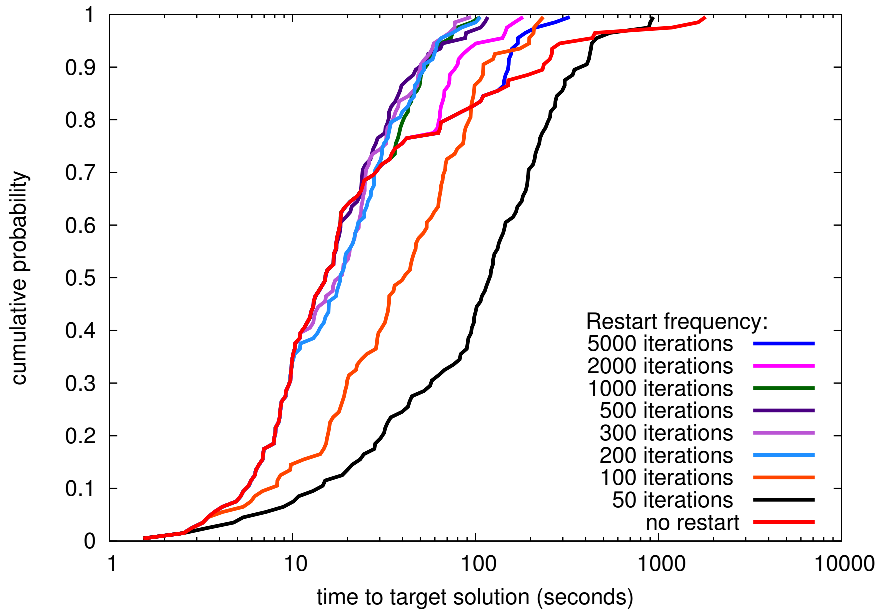


Fig. 19 Time-to-target plots for maximum cut instance *G12* using different values of κ. The figure also shows the time-to-target plot for the strategy without restarts. All runs of all strategies found a solution at least as good as the target value of 554.

We make some final observations about these experiments. The effect of the restart strategies can be mainly observed in the column corresponding to the fourth quartile of Table 6. Entries in this quartile correspond to those in the heavy tails of the distributions. The restart strategies in general did not affect the other quartiles of the distributions, which is a desirable characteristic. Compared to the no-restart strategy, restart strategies restart(500) and restart(1000) were able to reduce the maximum number of iterations, as well as the average and the standard deviation.

Table 6 Summary of computational results on maximum cut instance *G12* with four strategies. For each strategy, 100 independent runs were executed, each stopped when a solution as good as the target solution value 554 was found. For each strategy, the table shows the distribution of the number of iterations by quartile. For each quartile, the table gives the maximum number of iterations taken by all runs in that quartile, i.e., the slowest of the fastest 25% (1st), 50% (2nd), 75% (3rd), and 100% (4th) of the runs. The average number of iterations over the 100 runs and the standard deviation (st.dev.) are also given for each strategy.

	Iterations in quartile					
Strategy	1st	2nd	3rd	4th	Average	st.dev.
No restarts	326	550	1596	68813	4525.1	11927.0
restart(1000)	326	550	1423	5014	953.2	942.1
restart(500)	326	550	1152	4178	835.0	746.1
restart(100)	509	1243	3247	8382	2055.0	2005.9

Strategy restart(100) did so, too, but not as much as restart(500) and restart(1000). Restart strategies restart(500) and restart(1000) were clearly the best strategies of those tested.

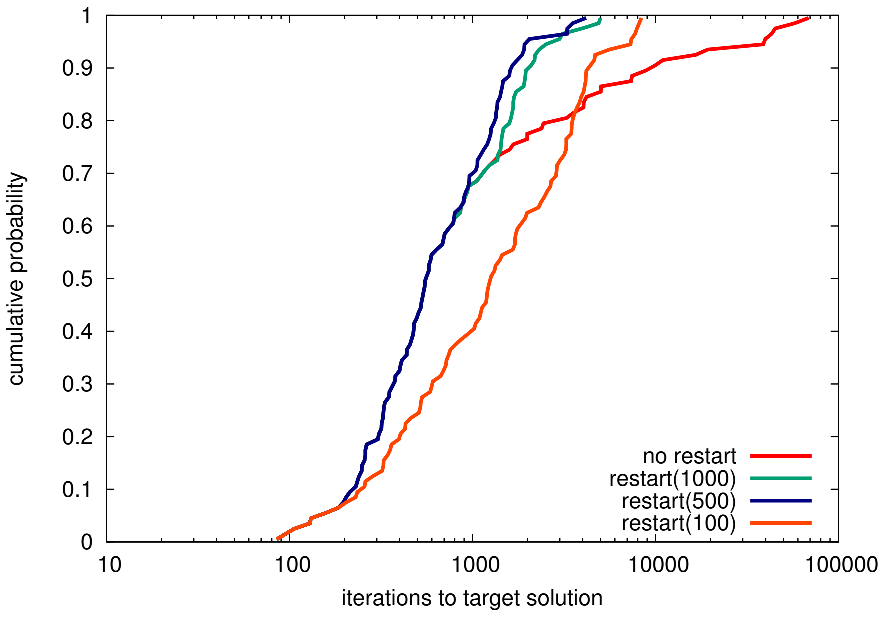


Fig. 20 Comparison of the iterations-to-target plots for maximum cut instance *G12* using strategies restart(100), restart(500), and restart(1000). The figure also shows the iterations-to-target plot for the strategy without restarts. All runs of all strategies found a solution at least as good as the target value of 554.

The restart(κ) strategy for GRASP with path-relinking discussed in this section was originally proposed by Resende and Ribeiro [219]. Besides the experiments presented in this chapter for the maximum cut instance *G12*, that paper also considered five other instances of maximum cut, maximum weighted satisfiability, and bandwidth packing. Interian and Ribeiro [135] implemented restart strategies for GRASP with path-relinking for the Steiner traveling salesman problem.

6 Extensions

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

The use of hash tables to avoid cycling in conjunction with tabu search was proposed by Woodruff and Zemel [266]. A similar approach was later explored by Ribeiro et al. [228] in their tabu search algorithm for query optimization in relational databases. In the context of GRASP implementations, hash tables were first used by Martins et al. [167] 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 are used to speed up the iterations of GRASP, see e.g. [92, 167, 202]. With filtering, local search is not applied to all solutions obtained at the end of the construction phase, but only to some more 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 [120, 172], rely almost entirely on the randomization of the local search to escape from local optima. With respect to randomization, GRASP and variable neighborhood strategies can be considered complementary and potentially capable of leading to effective hybrid methods. A first attempt in this direction was made by Martins et al. [167] where the construction phase of a 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, like the VND (variable neighborhood descent) procedure [120, 172]. That heuristic was later improved by Ribeiro et al. [234], where one of the key components of the new algorithm was another strategy for the exploration of different neighborhoods. Ribeiro and Souza [233] also combined GRASP with VND in a hybrid heuristic for the degree-constrained minimum spanning tree problem. Festa et al. [103] studied different variants and combinations of GRASP and VNS for the maximum 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. The greedy randomized strategy used in the construction phase of a GRASP heuristic is applied to generate the initial population for a genetic algorithm. As an example, consider the genetic algorithm of Ahuja et al. [5] for the quadratic assignment problem. It makes use of the GRASP heuristic proposed by Li et al. [150] to create the initial population of solutions. A similar approach was used by Armony et al. [31], with the initial population made up of both randomly generated solutions and those built by a GRASP heuristic.

The hybridization of GRASP with tabu search was first studied by Laguna and González-Velarde [146]. Delmaire et al. [70] 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 [42]. Lourenço et al. [154] point out that ILS has been rediscovered many times and is also known as iterated descent [40, 41], large step Markov chains [165], iterated Lin-Kernighan [136], and chained local optimization [164]. A GRASP/ILS hybrid can be obtained by replacing the standard local search of GRASP by ILS. The GRASP construction produces a solution which is passed to the ILS procedure. Ribeiro and Urrutia [235] presented a hybrid GRASP with ILS for the mirrored traveling tournament problem, in which perturbations are achieved by randomly generating solutions in the game rotation ejection chain [109, 110] neighborhood.

7 Applications

The first application of GRASP was described in the literature in 1989 [90]. In that paper, GRASP was applied to difficult set covering problems Since then, GRASP has been applied to a wide range of problems. The main applications areas are summarized below with links to specific references:

- Assignment problems [5, 7, 89, 105, 150, 153, 155, 169, 177, 178, 183, 189, 170, 191, 198, 202, 204, 212, 241]
- Biology [23, 64, 68, 75, 96, 107, 236]
- Computer vision [53, 131, 246, 247]
- Covering, packing, and partitioning [14, 15, 16, 27, 30, 71, 76, 90, 108, 118, 192, 195, 196, 223, 239, 243, 245]
- Diversity and dispersion [78, 82, 163, 211]
- Finance [19, 126]
- Graph and map drawing [66, 95, 147, 159, 160, 162, 184, 215, 227]
- Location and layout [1, 60, 66, 70, 119, 132, 140, 171, 181, 186, 253, 255, 260, 261]
- Logic [74, 97, 190, 209, 213, 214]
- Minimum Steiner tree [54, 166, 167, 168, 234]
- Optimization in graphs [2, 3, 4, 12, 32, 56, 72, 80, 79, 92, 98, 99, 133, 145, 148, 157, 160, 161, 167, 179, 188, 193, 208, 210, 215, 227, 234, 248, 257]
- Power systems [25, 48, 49, 85, 203, 263]
- Robotics [143, 240]
- Routing [29, 33, 38, 52, 55, 63, 135, 142, 144, 152, 175, 181, 206, 262, 264, 265]
- Software engineering [158]
- Sports [26, 139, 225, 235]
- Telecommunications [2, 18, 17, 20, 22, 31, 61, 106, 140, 153, 174, 176, 194, 197, 199, 202, 207, 208, 217, 226, 258]
- Timetabling, scheduling, and manufacturing [6, 11, 13, 20, 22, 24, 35, 36, 37, 39, 47, 50, 59, 62, 65, 69, 73, 77, 84, 86, 87, 88, 93, 94, 137, 141, 146, 149, 151, 173, 180, 185, 205, 235, 237, 238, 242, 244, 267, 268]

- Transportation [29, 34, 86, 89, 256]
- VLSI design [27, 28]

The reader is referred to Festa and Resende [102] and the book by Resende and Ribeiro [220] for extended annotated bibliographies of GRASP applications.

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 problems 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 most other metaheuristics, such as tabu search or genetic algorithms, that make use of a large number of parameters in their implementations, the basic variant of GRASP requires the adjustment of a single parameter, i.e. the restricted candidate list (RCL) parameter α.

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; 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 [6, 7, 8, 229]; and restart strategies to speedup the search.

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 21 shows some results obtained for the prize-collecting Steiner tree problem (PCSTP), as discussed by Canuto et al. in [54]. The figure shows results for eleven different levels of solution accuracy (varying from optimal to ten percent from optimal) on 40 PCSTP instances. 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 are the primal-dual constructive algorithm (GW) of Goemans and Williamson [116], 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). 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 [54]. 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.

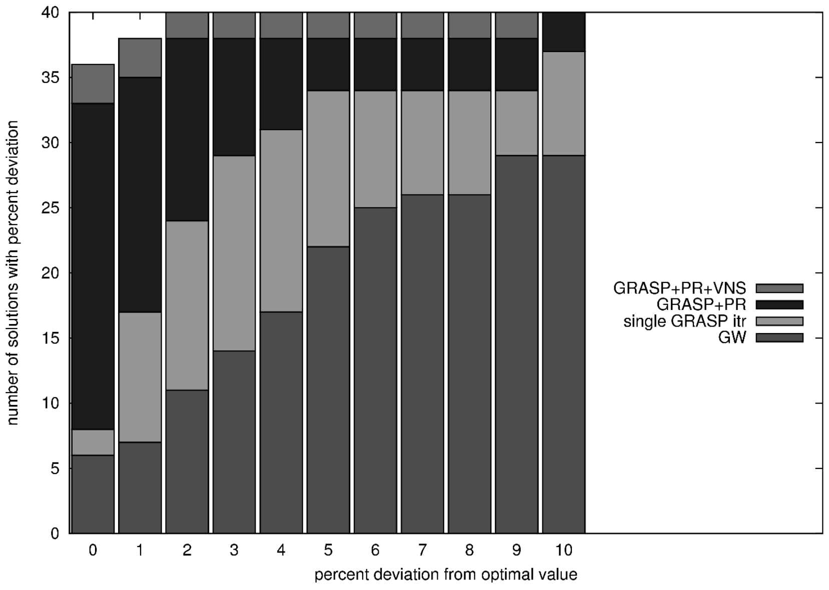


Fig. 21 Performance of GW approximation algorithm, a single GRASP iteration (GW followed by local search), 500 iterations of GRASP with path-relinking, and 500 iterations of GRASP with path-relinking followed by VNS for series C prize-collecting Steiner tree problems.

The structure of GRASP makes it very amenable to straightforward, efficient parallel implementations that benefit from the computer architecture. Parallel implementations of GRASP [6, 7, 8, 229] 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. Many parallel implementations of GRASP have been reported in the literature, see e.g. [167, 168, 178, 189, 190]. In many of these papers, a common observation was made: the speedups in the measured running times were proportional to the number of processors. This observation can be explained if the random variable *time-to-target-solution-value* is exponentially distributed. Aiex, Resende and Ribeiro [9] developed a graphical methodology based on runtime distributions to empirically show that the running times of GRASP heuristics fit exponential distributions, as summarized below.

Runtime distributions or time-to-target plots display on the ordinate axis the probability that an algorithm will find a solution at least as good as a given target value within a given running time, shown on the abscissa axis. They provide a very useful tool to characterize the running times of stochastic algorithms for combinatorial optimization problems and to compare different algorithms or strategies for solving a given problem. Time-to-target plots were first used by Feo, Resende and Smith [92] and have been widely used as a tool for algorithm design and comparison. Runtime distributions have also been advocated by Hoos and Stützle [134] as a way to characterize the running times of stochastic local search algorithms for combinatorial optimization. In particular, they have been largely applied to evaluate and compare the efficiency of different strategies of sequential and parallel implementations of GRASP with (and without) path-relinking heuristics. Aiex, Resende and Ribeiro [9] used time-to-target plots to show experimentally that the running times of GRASP heuristics fit shifted (or two-parameter) exponential distributions, reporting computational results for 2400 runs of GRASP heuristics for each of five different problems: maximum stable set, quadratic assignment, graph planarization [215, 216, 227], maximum weighted satisfiability, and maximum covering. Aiex, Resende, and Ribeiro [10] developed a Perl program to create time-to-target plots for measured times that are assumed to fit a shifted exponential distribution, following closely the work in [9]. Ribeiro, Rosseti, and Vallejos [231] developed a closed form result to compare two exponential algorithms and an iterative procedure to compare two algorithms following generic runtime distributions. This work was extended by Ribeiro, Rosseti and Vallejos [232] and was also applied in the comparison of parallel heuristics. Ribeiro and Rosseti [230] developed a code to compare runtime distributions of randomized algorithms.

This chapter provides the reader with the tools to build a basic GRASP to find optimal or near-optimal solutions to a combinatorial optimization problem. The chapter also provides the means to add to this basic GRASP more advanced features, like path-relinking and restart strategies, that enable better performance, both with respect to solution quality as well as solution run time. Left out of this chapter is the use of GRASP for solving continuous optimization problems. The interested reader is pointed to [127, 128, 129, 130, 220, 254] for an introduction to C-GRASP, or Continuous GRASP, as well as to some software and applications of C-GRASP.

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 and M.G.C. Resende. Parallel strategies for GRASP with path-relinking. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: Progress as real problem solvers*, pages 301–331. Springer, New York, 2005.
9. 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.
10. R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. TTTPLOTS: A perl program to create time-to-target plots. *Optimization Letters*, 1:355–366, 2007.
11. E. Alekseeva, M. Mezmaz, D. Tuyttens, and N. Melab. Parallel multi-core hyper-heuristic GRASP to solve permutation flow-shop problem. *Concurrency and Computation: Practice and Experience*, 29:e3835, 2017.
12. D. Aloise and C.C. Ribeiro. Adaptive memory in multistart heuristics for multicommodity network design. *J. of Heuristics*, 17:153–179, 2011.
13. 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.
14. 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.
15. R. Alvarez-Valdes, F. Parreño, and J.M. Tamarit. A GRASP/path relinking algorithm for two- and three-dimensional multiple bin-size bin packing problems. *Computers & Operations Research*, 40:3081–3090, 2013.
16. R. Alvarez-Valdesa, F. Parreno, and J.M. Tamarit. Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, 35:1065–1083, 2008.
17. 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.
18. 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.
19. K.P. Anagnostopoulos, P.D. Chatzoglou, and S. Katsavounis. A reactive greedy randomized adaptive search procedure for a mixed integer portfolio optimization problem. *Managerial Finance*, 36:1057–1065, 2010.
20. D.V. Andrade and M.G.C. Resende. A GRASP for PBX telephone migration scheduling. In *Proceedings of The Eighth INFORMS Telecommunications Conference*, 2006.
21. D.V. Andrade and M.G.C. Resende. GRASP with evolutionary path-relinking. Technical Report TD-6XPTS7, AT&T Labs Research, Florham Park, 2007.
22. 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.
23. A.A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. *J. of Heuristics*, 8:429–447, 2002.
24. 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.
25. C.H. Antunes, E. Oliveira, and P. Lima. A multi-objective GRASP procedure for reactive power compensation planning. *Optimization and Engineering*, 15:199–215, 2014.
26. 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.
27. 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.

28. S.M. Areibi. GRASP: An effective constructive technique for VLSI circuit partitioning. In *Proc. IEEE Canadian Conference on Electrical and Computer Engineering*, pages 462–467, Edmonton, Canada, 1999.
29. 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.
30. M.F. Argüello, T.A. Feo, and O. Goldschmidt. Randomized methods for the number partitioning problem. *Computers and Operations Research*, 23:103–111, 1996.
31. 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.
32. 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.
33. 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.
34. 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.
35. J.F. Bard and T.A. Feo. Operations sequencing in discrete parts manufacturing. *Management Science*, 35:249–255, 1989.
36. J.F. Bard and T.A. Feo. An algorithm for the manufacturing equipment selection problem. *IIE Transactions*, 23:83–92, 1991.
37. J.F. Bard, T.A. Feo, and S. Holland. A GRASP for scheduling printed wiring board assembly. *IIE Transactions*, 28:155–165, 1996.
38. 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.
39. J.F. Bard, Y. Shao, and A.I. Jarrah. A sequential GRASP for the therapist routing and scheduling problem. *J. of Scheduling*, 17:109–133, 2014.
40. E.B. Baum. Iterated descent: A better algorithm for local search in combinatorial optimization problems. Technical report, California Institute of Technology, 1986.
41. 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.
42. J. Baxter. Local optima avoidance in depot location. *J. of the Operational Research Society*, 32:815–819, 1981.
43. J.E. Beasley. An algorithm for set-covering problems. *European J. of Operational Research*, 31:85–93, 1987.
44. J.E. Beasley. A Lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37:151–164, 1990.
45. J.E. Beasley. Lagrangean relaxation. In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pages 243–303. Blackwell Scientific Publications, Oxford, 1993.
46. 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.
47. 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.
48. 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.
49. 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.
50. 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.

51. J.L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 271–278, Portland, 1996.
52. A.M. Campbell and B.W. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42:1–21, 2008.
53. R.G. Cano, G. Kunigami, C.C. de Souza, and P.J. de Rezende. A hybrid GRASP heuristic to construct effective drawings of proportional symbol maps. *Computers & Operations Research*, 40:1435–1447, 2013.
54. 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.
55. 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.
56. W.A. Chaovalitwongse, C.A.S Oliveira, B. Chiarini, P.M. Pardalos, and M.G.C. Resende. Revised GRASP with path-relinking for the linear ordering problem. *J. of Combinatorial Optimization*, 22:572–593, 2011.
57. I. Charon and O. Hudry. The noising method: A new method for combinatorial optimization. *Operations Research Letters*, 14:133–137, 1993.
58. 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.
59. M. Chica, O. Cordón, S. Damas, and J. Bautista. A multiobjective GRASP for the 1/3 variant of the time and space assembly line balancing problem. In N. García-Pedrajas, F. Herrera, C. Fyfe, J. Benítez, and M. Ali, editors, *Trends in Applied Intelligent Systems*, volume 6098 of *Lecture Notes in Computer Science*, pages 656–665. Springer, Berlin, 2010.
60. R. Colomé and D. Serra. Consumer choice in competitive location models: Formulations and heuristics. *Papers in Regional Science*, 80:439–464, 2001.
61. 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.
62. 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.
63. 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.
64. R. Cordone and G. Lulli. A GRASP metaheuristic for microarray data analysis. *Computers & Operations Research*, 40:3108–3120, 2013.
65. J.F. Correcher, M.T. Alonso, F. Parre no, and R. Alvarez-Valdes. Solving a large multicontainer loading problem in the car manufacturing industry. *Computers & Operations Research*, 82:139–152, 2017.
66. 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.
67. M.M. D'Apuzzo, A. Migdalas, P.M. Pardalos, and G. Toraldo. Parallel computing in global optimization. In E. Kontoghiorghes, editor, *Handbook of Parallel Computing and Statistics*. Chapman & Hall / CRC, Boca Raton, 2006.
68. S. Das and S.M. Idicula. Application of reactive GRASP to the biclustering of gene expression data. In *Proceedings of the International Symposium on Biocomputing*, page 14, Calicut, 2010. ACM.
69. 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.
70. 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.

71. X. Delorme, X. Gandibleux, and F. Degoutin. Evolutionary, constructive and hybrid procedures for the bi-objective set packing problem. *European J. of Operational Research*, 204:206–217, 2010.
72. Y. Deng and J.F. Bard. A reactive GRASP with path relinking for capacitated clustering. *J. of Heuristics*, 17:119–152, 2011.
73. Y. Deng, J.F. Bard, G.R. Chacon, and J. Stuber. Scheduling back-end operations in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 23:210–220, 2010.
74. 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.
75. S. Dharan and A.S. Nair. Biclustering of gene expression data using reactive greedy randomized adaptive search procedure. *BMC Bioinformatics*, 10 (Suppl 1):S27, 2009.
76. J.A. Díaz, D.E. Luna, J.-F. Camacho-Vallejo, and M.-S. Casas-Ramírez. GRASP and hybrid GRASP-Tabu heuristics to solve a maximal covering location problem with customer preference ordering. *Expert Systems with Applications*, 82:67–76, 2017.
77. A. Drexl and F. Salewski. Distribution requirements and compactness constraints in school timetabling. *European J. of Operational Research*, 102:193–214, 1997.
78. A. Duarte and R. Martí. Tabu search and GRASP for the maximum diversity problem. *European J. of Operational Research*, 178:71–84, 2007.
79. A. Duarte, R. Martí, A. Álvarez, and F. Ángel-Bello. Metaheuristics for the linear ordering problem with cumulative costs. *European J. of Operational Research*, 216:270–277, 2012.
80. A. Duarte, R. Martí, M.G.C. Resende, and R.M.A. Silva. GRASP with path relinking heuristics for the antibandwidth problem. *Networks*, 58:171–189, 2011.
81. 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.
82. A. Duarte, J. Sánchez-Oro, M.G.C. Resende, F. Glover, and R. Martí. GRASP with exterior path relinking for differential dispersion minimization. *Information Sciences*, 296:46–60, 2015.
83. 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.
84. M. Essafi, X. Delorme, and Al Dolgui. Balancing lines with CNC machines: A multi-start and based heuristic. *CIRP J. of Manufacturing Science and Technology*, 2:176–182, 2010.
85. 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.
86. T.A. Feo and J.F. Bard. Flight scheduling and maintenance base planning. *Management Science*, 35:1415–1432, 1989.
87. 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.
88. 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.
89. T.A. Feo and J.L. González-Velarde. The intermodal trailer assignment problem: Models, algorithms, and heuristics. *Transportation Science*, 29:330–341, 1995.
90. 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.
91. T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *J. of Global Optimization*, 6:109–133, 1995.
92. 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.
93. 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.

94. 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.
95. E. Fernández and R. Martí. GRASP for seam drawing in mosaicking of aerial photographic maps. *J. of Heuristics*, 5:181–197, 1999.
96. P. Festa. On some optimization problems in molecular biology. *Mathematical Bioscience*, 207:219–234, 2007.
97. 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.
98. 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.
99. 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.
100. 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.
101. P. Festa and M.G.C. Resende. An annotated bibliography of GRASP, Part I: Algorithms. *International Transactions in Operational Research*, 16:1–24, 2009.
102. P. Festa and M.G.C. Resende. An annotated bibliography of GRASP, Part II: Applications. *International Transactions in Operational Research*, 16:131–172, 2009.
103. 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.
104. M.L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 50:1861–1871, 2004.
105. 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.
106. E. Fonseca, R. Fuchsuber, L.F.M. Santos, A. Plastino, and S.L. Martins. Exploring the hybrid metaheuristic DM-GRASP for efficient server replication for reliable multicast. In *International Conference on Metaheuristics and Nature Inspired Computing*, Hammamet, 2008.
107. R.D. Frinhani, R.M. Silva, G.R. Mateus, P. Festa, and M.G.C. Resende. GRASP with path-relinking for data clustering: A case study for biological data. In P.M. Pardalos and S. Rebennack, editors, *Experimental algorithms*, volume 6630 of *Lecture Notes in Computer Science*, pages 410–420. Springer, Berlin, 2011.
108. J.B. Ghosh. Computatinal aspects of the maximum diversity problem. *Operations Research Letters*, 19:175–181, 1996.
109. 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.
110. F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65:223–254, 1996.
111. 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.
112. 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.
113. F. Glover. Exterior path relinking for zero-one optimization. *International J. of Applied Metaheuristic Computing*, 5:1–8, 2014.
114. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
115. F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

116. 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.
117. 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.
118. P.L. Hammer and D.J. Rader, Jr. Maximally disjoint solutions of the set covering problem. *J. of Heuristics*, 7:131–144, 2001.
119. 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.
120. 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.
121. J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.
122. M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.
123. M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971.
124. M. Held, P. Wolfe, and H.P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.
125. C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM J. on Optimization*, 10:673–696, 2000.
126. A.J. Higgins, S. Hajkowicz, and E. Bui. A multi-objective model for environmental investment decision making. *Computers & Operations Research*, 35:253–266, 2008.
127. M.J. Hirsch. *GRASP-based heuristics for continuous global optimization problems*. PhD thesis, Department of Industrial and Systems Engineering, University of Florida, Gainesville, 2006.
128. M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and M.G.C. Resende. Global optimization by continuous GRASP. *Optimization Letters*, 1:201–212, 2007.
129. M.J. Hirsch, P.M. Pardalos, and M.G.C. Resende. Solving systems of nonlinear equations with continuous GRASP. *Nonlinear Analysis: Real World Applications*, 10:2000–2006, 2009.
130. M.J. Hirsch, P.M. Pardalos, and M.G.C. Resende. Speeding up continuous GRASP. *European Journal of Operational Research*, 205:507–521, 2010.
131. M.J. Hirsch, P.M. Pardalos, and M.G.C. Resende. Correspondence of projected 3D points and lines using a continuous GRASP. *International Transactions in Operational Research*, 18:493–511, 2011.
132. 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.
133. 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.
134. H.H. Hoos and T. Stützle. Evaluation of Las Vegas algorithms - Pitfalls and remedies. In G. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 238–245, Madison, 1998.
135. R. Interian and C. C. Ribeiro. A GRASP heuristic using path-relinking and restarts for the Steiner traveling salesman problem. *International Transactions in Operational Research*, 24:1307–1323, 2017.

136. 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.
137. E.H. Kampke, J.E.C. Arroyo, and A.G. Santos. Reactive GRASP with path relinking for solving parallel machines scheduling problem with resource-assignable sequence dependent setup times. In *Proceedings of the World Congress on Nature and Biologically Inspired Computing*, pages 924–929, Coimbatore, 2009. IEEE.
138. H. Kautz, E. Horvitz, Y. Ruan, C. Gomes, and B. Selman. Dynamic restart policies. In *Proceedings of the Eighteenth National Conference on Artificial intelligence*, pages 674–681, Edmonton, 2002. American Association for Artificial Intelligence.
139. G. Kendall, S. Knust, C.C. Ribeiro, and S. Urrutia. Scheduling in sports: An annotated bibliography. *Computers & Operations Research*, 37:1–19, 2010.
140. 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.
141. J.G. Klincewicz and A. Rajan. Using GRASP to solve the component grouping problem. *Naval Research Logistics*, 41:893–912, 1994.
142. G. Kontoravdis and J.F. Bard. A GRASP for the vehicle routing problem with time windows. *ORSA J. on Computing*, 7:10–23, 1995.
143. M. Kulich, J. J. Miranda-Bront, and L. Preucil. A meta-heuristic based goal-selection strategy for mobile robot search in an unknown environment. *Computers & Operations Research*, 84:178–187, 2017.
144. N. Labadi, C. Prins, and M. Reghioui. GRASP with path relinking for the capacitated arc routing problem with time windows. In A. Fink and F. Rothlauf, editors, *Advances in Computational Intelligence in Transport, Logistics, and Supply Chain Management*, pages 111–135. Springer, Berlin, 2008.
145. 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.
146. 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.
147. M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, 11:44–52, 1999.
148. M. Laguna and R. Martí. A GRASP for coloring sparse graphs. *Computaional Optimization and Applications*, 19:165–178, 2001.
149. R. De Leone, P. Festa, and E. Marchitto. Solving a bus driver scheduling problem with randomized multistart heuristics. *International Transactions in Operational Research*, 18:707–727, 2011.
150. 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.
151. A. Lim, B. Rodrigues, and C. Wang. Two-machine flow shop problems with a single server. *J. of Scheduling*, 9:515–543, 2006.
152. A. Lim and F. Wang. A smoothed dynamic tabu search embedded GRASP for m-VRPTW. In *Proceedings of ICTAI 2004*, pages 704–708, 2004.
153. 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.
154. 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.
155. H.R. Lourenço and D. Serra. Adaptive approach heuristics for the generalized assignment problem. *Mathware and Soft Computing*, 9:209–234, 2002.

156. M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47:173–180, 1993.
157. M. Luis, S. Salhi, and G. Nagy. A guided reactive GRASP for the capacitated multi-source Weber problem. *Computers & Operations Research*, 38:1014–1024, 2011.
158. C.L.B. Maia, R.A.F. Carmo, F.G. Freitas, G.A.L. Campos, and J.T. Souza. Automated test case prioritization with reactive GRASP. *Advances in Software Engineering*, 2010, 2010. Article ID 428521.
159. R. Martí. Arc crossing minimization in graphs with GRASP. *IEE Transactions*, 33:913–919, 2001.
160. R. Martí. Arc crossing minimization in graphs with GRASP. *IEEE Transactions*, 33:913–919, 2002.
161. R. Martí and V. Estruch. Incremental bipartite drawing problem. *Computers and Operations Research*, 28:1287–1298, 2001.
162. R. Martí and M. Laguna. Heuristics and meta-heuristics for 2-layer straight line crossing minimization. *Discrete Applied Mathematics*, 127:665–678, 2003.
163. R. Martí and F. Sandoya. GRASP and path relinking for the equitable dispersion problem. *Computers & Operations Research*, 40:3091–3099, 2013.
164. O. Martin and S.W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.
165. O. Martin, S.W. Otto, and E.W. Felten. Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.
166. 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.
167. S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P.M. Pardalos. A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization*, 17:267–283, 2000.
168. 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.
169. G.R. Mateus, M.G.C. Resende, and R.M.A. Silva. GRASP with path-relinking for the generalized quadratic assignment problem. *J. of Heuristics*, 17:527–565, 2011.
170. 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.
171. M. Mestria, L.S. Ochi, and S.L. Martins. GRASP with path relinking for the symmetric Euclidean clustered traveling salesman problem. *Computers & Operations Research*, 40:3218–3229, 2013.
172. N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.
173. 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.
174. R.E.N. Moraes and C.C. Ribeiro. Power optimization in ad hoc wireless network topology control with biconnectivity requirements. *Computers & Operations Research*, 40:3188–3196, 2013.
175. L.F. Morán-Mirabal, J.L. González-Velarde, and M.G.C. Resende. Randomized heuristics for the family traveling salesperson problem. *International Transactions in Operational Research*, 21:41–57, 2014.
176. L.F. Morán-Mirabal, J.L. González-Velarde, M.G.C. Resende, and R.M.A. Silva. Randomized heuristics for handover minimization in mobility networks. *J. of Heuristics*, 19:845–880, 2013.

177. 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.
178. 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.
179. M.C.V. Nascimento and L. Pitsoulis. Community detection by modularity maximization using GRASP with path relinking. *Computers & Operations Research*, 40:3121–3131, 2013.
180. M.C.V. Nascimento, M.G.C. Resende, and F.M.B. Toledo. GRASP heuristic with path-relinking for the multi-plant capacitated lot sizing problem. *European J. of Operational Research*, 200:747–754, 2010.
181. V.-P. Nguyen, C. Prins, and C. Prodhon. Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. *European J. of Operational Research*, 216:113–126, 2012.
182. E. Nowicki and C. Smutnicki. An advanced tabu search algorithm for the job shop problem. *J. of Scheduling*, 8:145–159, 2005.
183. 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.
184. 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.
185. A.V.F. Pacheco, , G.M. Ribeiro, and G.R. Mauri. A GRASP with path-relinking for the workover rig scheduling problem. *International J. of Natural Computing Research*, 1:1–14, 2010.
186. 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.
187. G. Palubeckis. Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. *Annals of Operations Research*, 131:259–282, 2004.
188. 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.
189. 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.
190. 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.
191. 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.
192. F. Parreño, R. Alvarez-Valdes, J.M. Tamarit, and J.F. Oliveira. A maximal-space algorithm for the container loading problem. *INFORMS J. on Computing*, 20:412–422, 2008.
193. 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.
194. O. Pedrola, M. Ruiz, L. Velasco, D. Careglio, O. González de Dios, and J. Comellas. A GRASP with path-relinking heuristic for the survivable IP/MPLS-over-WSON multi-layer network optimization problem. *Computers & Operations Research*, 40:3174–3187, 2013.
195. L.S. Pessoa, M.G.C. Resende, and C.C. Ribeiro. Experiments with the LAGRASP heuristic for set k-covering. *Optimization Letters*, 5:407–419, 2011.

196. L.S. Pessoa, M.G.C. Resende, and C.C. Ribeiro. A hybrid Lagrangean heuristic with GRASP and path-relinking for set k-covering. *Computers & Operations Research*, 40:3132–3146, 2013.
197. 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.
198. 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.
199. 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.
200. 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.
201. M. Prais and C.C. Ribeiro. Parameter variation in GRASP procedures. *Investigación Operativa*, 9:1–20, 2000.
202. 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.
203. M. Rahmani, M. Rashidinejad, E.M. Carreno, and R.A. Romero. Evolutionary multi-move path-relinking for transmission network expansion planning. In *2010 IEEE Power and Energy Society General Meeting*, pages 1–6, Minneapolis, 2010. IEEE.
204. 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.
205. 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.
206. 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.
207. 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.
208. M.G.C. Resende. Computing approximate solutions of the maximum covering problem using GRASP. *J. of Heuristics*, 4:161–171, 1998.
209. 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.
210. 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.
211. M.G.C. Resende, R. Martí, M. Gallego, and A. Duarte. GRASP and path relinking for the max-min diversity problem. *Computers & Operations Research*, 37:498–508, 2010.
212. 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.
213. 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.
214. 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.
215. M.G.C. Resende and C.C. Ribeiro. A GRASP for graph planarization. *Networks*, 29:173–189, 1997.

216. M.G.C. Resende and C.C. Ribeiro. Graph planarization. In C. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, volume 2, pages 368–373. Kluwer Academic Publishers, Boston, 2001.
217. M.G.C. Resende and C.C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41:104–114, 2003.
218. 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.
219. M.G.C. Resende and C.C. Ribeiro. Restart strategies for GRASP with path-relinking heuristics. *Optimization Letters*, 5:467–478, 2011.
220. M.G.C. Resende and C.C. Ribeiro. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. Springer, 2016.
221. M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the p-median problem. *J. of Heuristics*, 10:59–88, 2004.
222. 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.
223. A.P. Reynolds and B. de la Iglesia. A multi-objective GRASP for partial classification. *Soft Computing*, 13:227–243, 2009.
224. 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.
225. C.C. Ribeiro. Sports scheduling: Problems and applications. *International Transactions in Operational Research*, 19:201–226, 2012.
226. C.C. Ribeiro, S.L. Martins, and I. Rosseti. Metaheuristics for optimization problems in computer communications. *Computer Comunications*, 30:656–669, 2007.
227. 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.
228. C.C. Ribeiro, C.D. Ribeiro, and R.S. Lanzelotte. Query optimization in distributed relational databases. *J. of Heuristics*, 3:5–23, 1997.
229. C.C. Ribeiro and I. Rosseti. Efficient parallel cooperative implementations of GRASP heuristics. *Parallel Computing*, 33:21–35, 2007.
230. C.C. Ribeiro and I. Rosseti. tttplots-compare: A perl program to compare time-to-target plots or general runtime distributions of randomized algorithms. *Optimization Letters*, 9:601–614, 2015.
231. C.C. Ribeiro, I. Rosseti, and R. Vallejos. On the use of run time distributions to evaluate and compare stochastic local search algorithms. In T. Sttzle, M. Biratari, and H.H. Hoos, editors, *Engineering stochastic local search algorithms*, volume 5752 of *Lecture Notes in Computer Science*, pages 16–30. Springer, Berlin, 2009.
232. C.C. Ribeiro, I. Rosseti, and R. Vallejos. Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms. *J. of Global Optimization*, 54:405–429, 2012.
233. 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.
234. 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.
235. C.C. Ribeiro and S. Urrutia. Heuristics for the mirrored traveling tournament problem. *European J. of Operational Research*, 179:775–787, 2007.
236. C.C. Ribeiro and D.S. Vianna. A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. *International Transactions in Operational Research*, 12:325–338, 2005.
237. R.Z. Ríos-Mercado and J.F. Bard. Heuristics for the flow line problem with setup costs. *European J. of Operational Research*, 110:76–98, 1998.
238. 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.

239. R.Z. Ríos-Mercado and E. Fernández. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research*, 36:755–776, 2009.
240. A. Riva and F. Amigoni. A GRASP metaheuristic for the coverage of grid environments with limited-footprint tools. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 484–491, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.
241. 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.
242. 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-Heuristics*, volume 1, pages 62–67, 2004.
243. F. J. Rodriguez, F. Glover, C. García-Martínez, R. Martí, and M. Lozano. Grasp with exterior path-relinking and restricted local search for the multidimensional two-way number partitioning problem. *Computers & Operations Research*, 78:243–254, 2017.
244. F.J. Rodriguez, C. Blum, C. García-Martínez, and M. Lozano. GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times. *Annals of Operations Research*, 201:383–401, 2012.
245. M.A. Salazar-Aguilar, R.Z. Ríos-Mercado, and J.L. González-Velarde. GRASP strategies for a bi-objective commercial territory design problem. *J. of Heuristics*, 19:179–200, 2013.
246. J. Santamaría, O. Cordón, S. Damas, R. Martí, and R.J. Palma. GRASP & evolutionary path relinking for medical image registration based on point matching. In *2010 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.
247. J. Santamaría, O. Cordón, S. Damas, R. Martí, and R.J. Palma. GRASP and path relinking hybridizations for the point matching-based image registration problem. *J. of Heuristics*, 18:169–192, 2012.
248. D. Santos, A. de Sousa, and F. Alvelos. A hybrid column generation with GRASP and path relinking for the network load balancing problem. *Computers & Operations Research*, 40:3147–3158, 2013.
249. M. Scaparra and R. Church. A GRASP and path relinking heuristic for rural road network development. *J. of Heuristics*, 11:89–108, 2005.
250. I.V. Sergienko, V.P. Shilo, and V.A. Roshchin. Optimization parallelizing for discrete programming problems. *Cybernetics and Systems Analysis*, 40:184–189, 2004.
251. O.V. Shylo, T. Middelkoop, and P.M. Pardalos. Restart strategies in optimization: Parallel and serial cases. *Parallel Computing*, 37:60–68, 2011.
252. O.V. Shylo, O.A. Prokopyev, and J. Rajgopal. On algorithm portfolios and restart strategies. *Operations Research Letters*, 39:49–52, 2011.
253. F. Silva and D. Serra. Locating emergency services with different priorities: The priority queuing covering location problem. *J. of the Operational Research Society*, 59:1229–1238, 2007.
254. R.M.A. Silva, M.G.C. Resende, P.M. Pardalos, and M.J. Hirsch. A Python/C library for bound-constrained global optimization with continuous GRASP. *Optimization Letters*, 7:967–984, 2013.
255. R.M.A. Silva, M.G.C. Resende, P.M. Pardalos, G.R. Mateus, and G. de Tomi. GRASP with path-relinking for facility layout. In B.I. Goldengorin, V.A. Kalyagin, and P.M. Pardalos, editors, *Models, algorithms, and technologies for network analysis*, volume 59 of *Springer Proceedings in Mathematics & Statistics*, pages 175–190. Springer, Berlin, 2013.
256. 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.
257. 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.

258. 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.
259. H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.
260. T.L. Urban. Solution procedures for the dynamic facility layout problem. *Annals of Operations Research*, 76:323–342, 1998.
261. 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.
262. F.L. Usberti, P.M. França, and A.L.M. França. GRASP with evolutionary path-relinking for the capacitated arc routing problem. *Computers & Operations Research*, 40:3206–3217, 2013.
263. J.X. Vianna Neto, D.L.A. Bernert, and L.S. Coelho. Continuous GRASP algorithm applied to economic dispatch problem of thermal units. In *Proceedings of the 13th Brazilian Congress of Thermal Sciences and Engineering*, Uberlandia, 2010.
264. J.G. Villegas, C. Prins, C. Prodhon, A.L. Medaglia, and N. Velasco. GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. *Engineering Applications of Artificial Intelligence*, 23:780–794, 2010.
265. J.G. Villegas, C. Prins, C. Prodhon, A.L. Medaglia, and N. Velasco. A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, 38:1319–1334, 2011.
266. D.L. Woodruff and E. Zemel. Hashing vectors for tabu search. *Annals of Operations Research*, 41:123–137, 1993.
267. J.Y. Xu and S.Y. Chiu. Effective heuristic procedure for a field technician scheduling problem. *J. of Heuristics*, 7:495–509, 2001.
268. 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.