Chapter 8

GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURES

Mauricio G.C. Resende
AT&T Labs Research
E-mail: mgcr@research.att.com

Celso C. Ribeiro
Catholic University of Rio de Janeiro
E-mail: celso@inf.puc-rio.br

Abstract GRASP is a multi-start metaheuristic for combinatorial 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 and parameter tuning strategies 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: Reactive GRASP, cost perturbations, bias functions, memory and learning, local search on partially constructed solutions, hashing, and filtering. We also discuss in detail implementation strategies of memory-based intensification and post-optimization techniques using path-relinking. Hybridizations with other metaheuristics, parallelization strategies, and applications are also reviewed.

1 INTRODUCTION

We consider in this chapter a combinatorial optimization problem, defined by a finite ground set $E = \{1, \ldots, n\}$, a set of feasible solutions $F \subseteq 2^E$, and an objective function $f : 2^E \rightarrow \mathbb{R}$. In the minimization version, we search an optimal solution $S^* \in F$ such that $f(S^*) \leq f(S), \forall S \in F$. The ground set E, the cost function f, and the set of feasible solutions F are defined for each specific problem. For instance, in the case of the traveling salesman problem, the ground set E is that of all edges connecting the cities to be visited, $f(S)$ is the sum of the costs of all edges $e \in S$, and F is formed by all egde subsets that determine a Hamiltonian cycle.

The GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic [38,39] is a multi-start or iterative process, in which each iteration consists 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. An extensive survey of

```
procedure GRASP(Max_Iterations, Seed)
1    Read_Input();
2    for k = 1, ..., Max_Iterations do
3        Solution ← Greedy_Randomized_Construction(Seed);
4        Solution ← Local_Search(Solution);
5        Update_Solution(Solution, Best_Solution);
6    end;
7    return Best_Solution;
end GRASP.
```

Figure 8.1. Pseudo-code of the GRASP metaheuristic.

```
procedure Greedy_Randomized_Construction(Seed)
1    Solution ← ∅;
2    Evaluate the incremental costs of the candidate elements;
3    while Solution is not a complete solution do
4        Build the restricted candidate list (RCL);
5        Select an element s from the RCL at random;
6        Solution ← Solution ∪{s};
7        Reevaluate the incremental costs;
8    end;
9    return Solution;
end Greedy_Randomized_Construction.
```

Figure 8.2. Pseudo-code of the construction phase.

the literature is given in [44]. The pseudo-code in Figure 8.1 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 pseudorandom number generator.

Figure 8.2 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 that can be incorporated to the partial solution under construction without destroying feasibility. 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 to the partial solution, the candidate list is updated and the incremental costs are reevaluated (this is the adaptive aspect of the heuristic). This strategy is similar to the semi-greedy heuristic proposed by Hart and Shogan [55], which is also a multi-start approach based on greedy randomized constructions, but without local search.

The solutions generated by a greedy randomized construction are not necessarily optimal, even with respect to simple neighborhoods. The local search phase usually

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

Figure 8.3. Pseudo-code of the local search phase.

improves the constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The pseudo-code of a basic local search algorithm starting from the solution `Solution` constructed in the first phase and using a neighborhood N is given in Figure 8.3.

The effectiveness of a local search procedure depends on several aspects, such as the neighborhood structure, the neighborhood search technique, the fast evaluation of the cost function of the neighbors, and the starting solution itself. The construction phase plays a very important role with respect to this last aspect, building high-quality starting solutions for the local search. Simple neighborhoods are usually used. The neighborhood search may be implemented using either a *best-improving* or a *first-improving* strategy. In the case of the best-improving strategy, all neighbors are investigated and the current solution is replaced by the best neighbor. In the case of a first-improving strategy, the current solution moves to the first neighbor whose cost function value is smaller than that of the current solution. In practice, we observed on many applications that quite often both strategies lead to the same final solution, but in smaller computation times when the first-improving strategy is used. We also observed that premature convergence to a non-global 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 efficient data structures to assure quick iterations. GRASP has two main parameters: one related to the stopping criterion and another to the quality of the elements in the restricted candidate list.

The stopping criterion used in the pseudo-code described in Figure 8.1 is determined by the number `Max_Iterations` of iterations. Although the probability of finding a new solution improving the currently best decreases with the number of iterations, the quality of the best solution found may only improve with the latter. Since the computation time does not vary much from iteration to iteration, the total computation time is predictable and increases linearly with the number of iterations. Consequently, the larger the number of iterations, the larger will be the computation time and the better will be the solution found.

For the construction of the RCL used in the first phase we consider, without loss of generality, a minimization problem as the one formulated in Section 1. We denote

```
procedure Greedy_Randomized_Construction(α, Seed)
1    Solution ← ∅;
2    Initialize the candidate set: C ← E;
3    Evaluate the incremental cost c(e) for all e ∈ C;
4    while C ≠ ∅ do
5        c^min ← min{c(e)|e ∈ C};
6        c^max ← max{c(e)|e ∈ C};
7        RCL ← {e ∈ C| c(e) ≤ c^min + α(c^max − c^min)};
8        Select an element s from the RCL at random;
9        Solution ← Solution ∪{s};
10       Update the candidate set C;
11       Reevaluate the incremental costs c(e) for all e ∈ C;
12   end;
13   return Solution;
end Greedy_Randomized_Construction.
```

Figure 8.4. Refined pseudo-code of the construction phase.

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 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 "feasible" elements $e \in E$ which can be inserted into the partial solution under construction without destroying feasibility and whose quality is superior to the threshold value, i.e., $c(e) \in [c^{\min}, c^{\min} + \alpha(c^{\max} - c^{\min})]$. The case $\alpha = 0$ corresponds to a pure greedy algorithm, while $\alpha = 1$ is equivalent to a random construction. The pseudo-code in Figure 8.4 is a refinement of the greedy randomized contruction pseudo-code shown in Figure 8.2. It shows that the parameter α controls the amounts of greediness and randomness in the algorithm.

GRASP may be viewed as a repetitive sampling technique. Each iteration produces a sample solution from an unknown distribution, whose mean and variance are functions of the restrictive nature of the RCL. For example, if the RCL is restricted to a single element, then the same solution will be produced at all iterations. The variance of the distribution will be zero and the mean will be equal to the value of the greedy solution. If the RCL is allowed to have more elements, then many different solutions will be produced, implying a larger variance. Since greediness plays a smaller role in this case, the mean solution value should be worse. However, the value of the best solution found outperforms the mean value and very often is optimal. The histograms in Figure 8.5 illustrate this situation on an instance of MAXSAT with 100 variables and 850 clauses, depicting results obtained with 1000 independent constructions using the first phase of the GRASP described in [83,84]. Since this is a maximization problem, the purely greedy construction corresponds to $\alpha = 1$, whereas the random construction occurs with $\alpha = 0$. We notice that when the value of α increases from 0 to 1, the mean

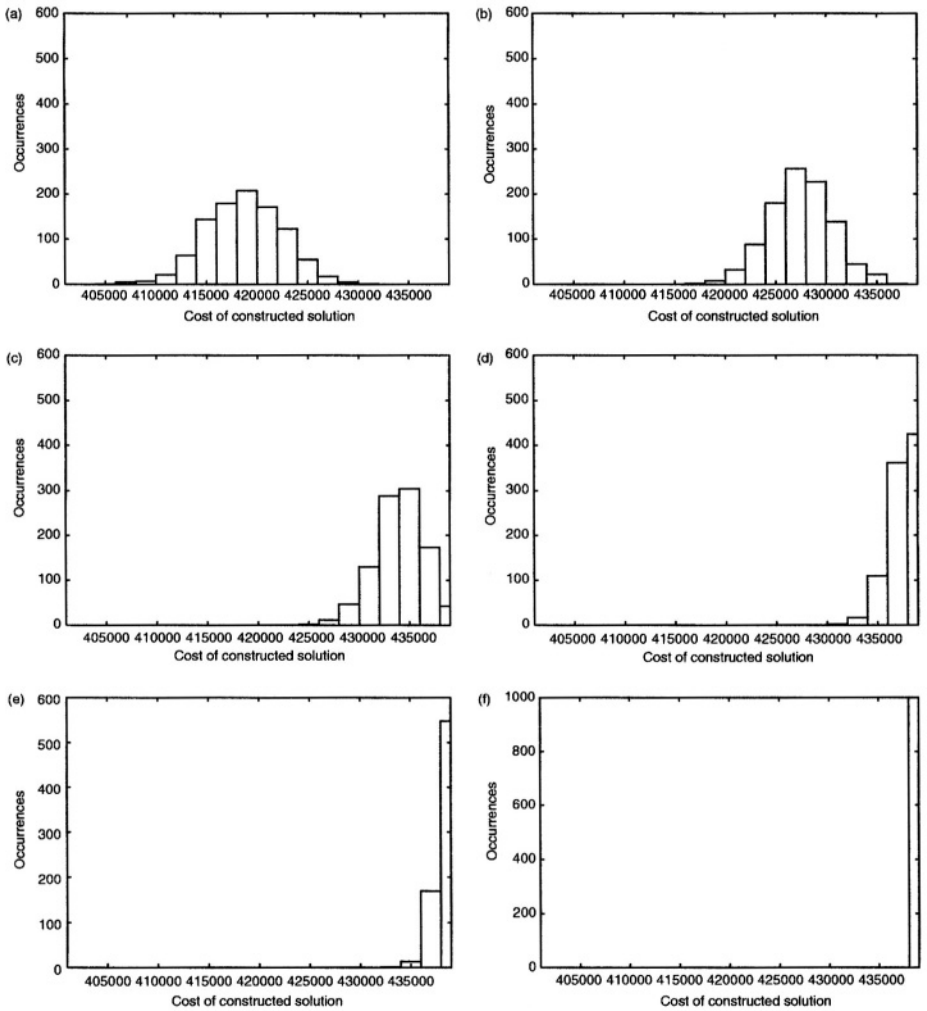

Figure 8.5. Distribution of construction phase solution values as a function of the RCL parameter α (1000 repetitions were recorded for each value of α). (a) $\alpha = 0.0$ (random construction), (b) $\alpha = 0.2$, (c) $\alpha = 0.4$, (d) $\alpha = 0.6$, (e) $\alpha = 0.8$ and (f) $\alpha = 1.0$ (greedy construction).

solution value increases towards the purely greedy solution value, while the variance approaches zero.

For each value of α, we present in Figure 8.6 histograms with the results obtained by applying local search to each of the 1000 constructed solutions. Figure 8.7 summarizes the overall results of this experiment in terms of solution diversity, solution quality, and computation time. We first observe that the larger the variance of the solution values obtained in the construction phase, the larger is the variance of the overall solution values, as shown in the top graph. The graph in the middle illustrates the relationship between the variance of the solution values and the average solution values, and how this affects the best solution found. It is unlikely that GRASP will find an optimal solution if the average solution value is low, even if there is a large variance in the

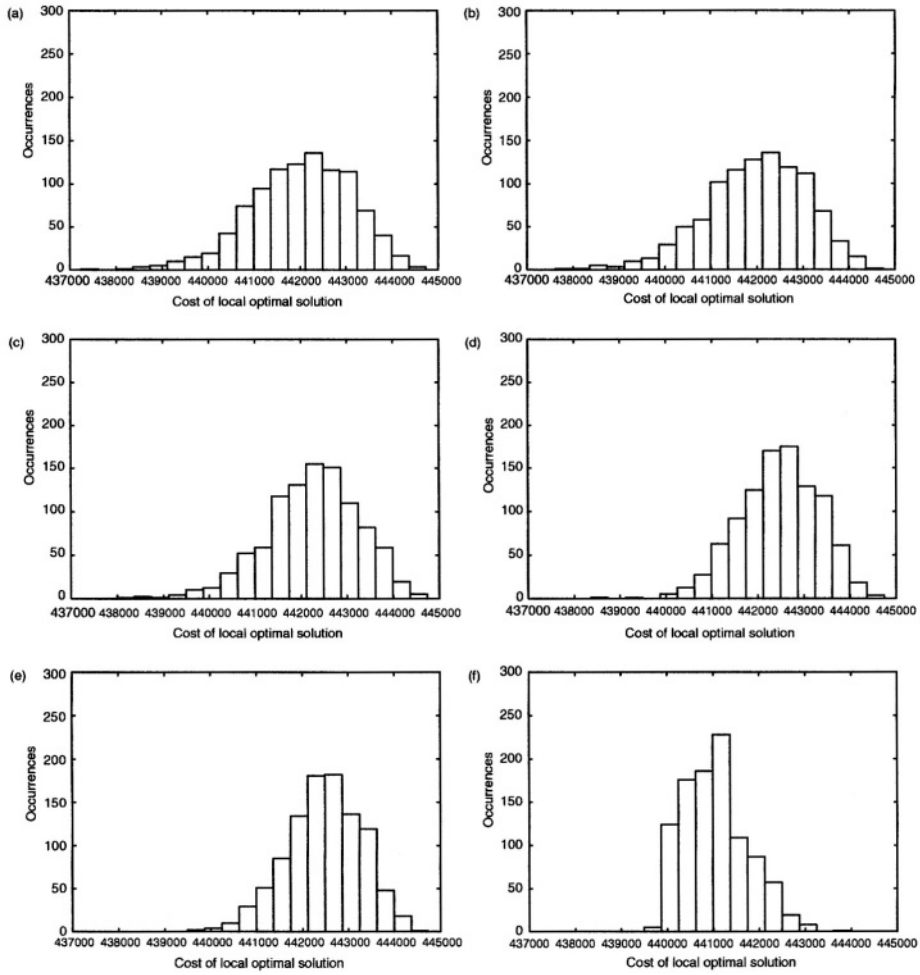

Figure 8.6. Distribution of local search phase solution values as a function of the RCL parameter α (1000 repetitions for each value of α). (a) $\alpha = 0.0$ (random), (b) $\alpha = 0.2$, (c) $\alpha = 0.4$, (d) $\alpha = 0.6$, (e) $\alpha = 0.8$ and (f) $\alpha = 1.0$ (greedy).

overall solution values, such as is the case for $\alpha = 0$. On the other hand, if there is little variance in the overall solution values, it is also unlikely that GRASP will find an optimal solution, even if the average solution is high, as is the case for $\alpha = 1$. What often leads to good solutions are relatively high average solution values in the presence of a relatively large variance, such as is the case for $\alpha = 0.8$. The middle graph also shows that the distance between the average solution value and the value of the best solution found increases as the construction phase moves from more greedy to more random. This causes the average time taken by the local search to increase, as shown in the graph in the bottom. Very often, many GRASP solutions are generated in the same amount of time required for the local optimization procedure to converge from a single random start.

These results are illustrated in Table 8.1 and Figure 8.8, for another instance of MAXSAT where 1000 iterations were run. For each value of α ranging from 0 (purely

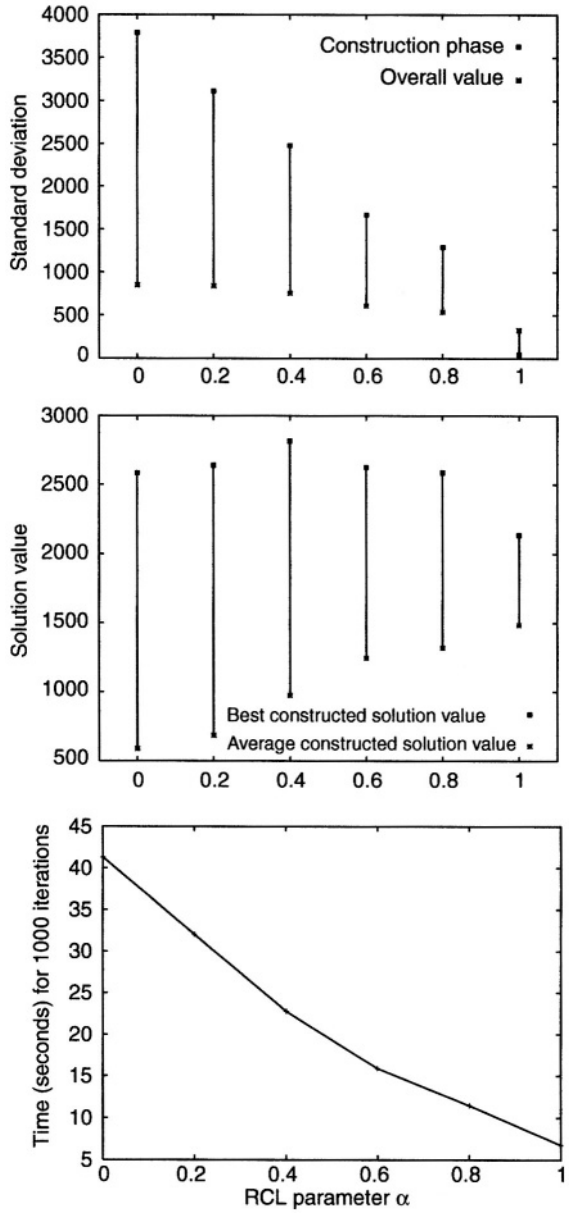

Figure 8.7. Standard deviation of the solution values found, best and average solution values found, and total processing time as a function of the RCL parameter α (1000 repetitions were recorded for each value of α).

random construction) to 1 (purely greedy construction), we give in Table 8.1 the average Hamming distance between each solution built at the end of the construction phase and the corresponding local optimum obtained after local search, the average number of moves from the first to the latter, the local search time in seconds, and the total processing time in seconds. Figure 8.8 summarizes the values observed for the total

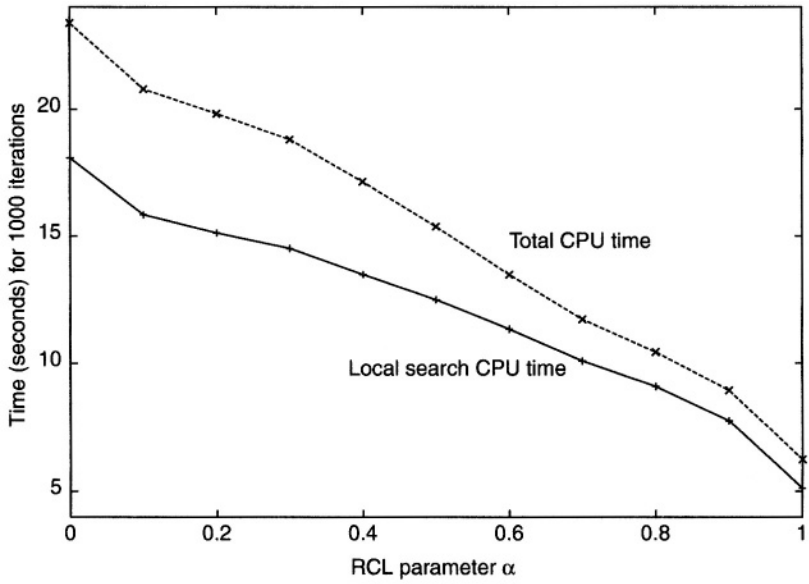

Figure 8.8. Total CPU time and local search CPU time as a function of the RCL parameter α (1000 repetitions for each value of α).

Table 8.1. Average number of moves and local search time as a function of the RCL parameter α

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

processing time and the local search time. We notice that both time measures considerably decrease as α tends to 1, approaching the purely greedy choice. In particular, we observe that the average local search time taken by $\alpha = 0$ (purely random) is approximately 2.5 times that taken in the case $\alpha = 0.9$ (almost greedy). In this example, two to three greedily constructed solutions can be investigated in the same time needed to apply local search to one single randomly constructed solution. The appropriate choice of the value of the RCL parameter α is clearly critical and relevant to achieve a good balance between computation time and solution quality.

Prais and Ribeiro [77] have shown that using a single fixed value for the value of RCL parameter α very often hinders finding a high-quality solution, which eventually

could be found if another value was used. They proposed an extension of the basic GRASP procedure, which they call *Reactive GRASP,* in which the parameter α is self-tuned and its value is periodically modifed according with the quality of the solutions obtained recently. In particular, computational experiments on the problem of traffic assignment in communication satellites [78] have shown that Reactive GRASP found better solutions than the basic algorithm for many test instances. These results motivated the study of the behavior of GRASP for different strategies for the variation of the value of the RCL parameter α:

(R) α self tuned according with the Reactive GRASP procedure;

(E) α randomly chosen from a uniform discrete probability distribution;

(H) α randomly chosen from a decreasing non-uniform discrete probability distribution; and

(F) fixed value of α, close to the purely greedy choice.

We summarize the results obtained by the experiments reported in [76,77]. These four strategies were incorporated into the GRASP procedures developed for four different optimization problems: (P-1) matrix decomposition for traffic assignment in communication satellite [78], (P-2) set covering [38], (P-3) weighted MAX-SAT [83,84], and (P-4) graph planarization [85,87]. 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 based on using the discrete uniform distribution (E), all choice probabilities are equal to $1/m$. The third case corresponds to the a hybrid strategy (H), in which we typically consider $p(\alpha = 0.1) = 0.5$, $p(\alpha = 0.2) = 0.25$, $p(\alpha = 0.3) = 0.125$, $p(\alpha = 0.4) = 0.03$, $p(\alpha = 0.5) = 0.03$, $p(\alpha = 0.6) = 0.03$, $p(\alpha = 0.7) = 0.01$, $p(\alpha = 0.8) = 0.01$, $p(\alpha = 0.9) = 0.01$, and $p(\alpha = 1.0) = 0.005$. Finally, in the last strategy (F), the value of α is fixed as recommended in the original reference 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 [77] are summarized in Table 8.2. 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.

Strategy (F) presented the shortest average computation times for three out of the four problem types. It was also the one with the least variability in the constructed

Table 8.2. Computational results for different strategies for the variation of parameter α

		R		*E*		*H*		*F*	
Problem	Total	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	

solutions and, in consequence, found the best solution the fewest times. The reactive strategy (R) is the one which most often found the best solutions, however, at the cost of computation times that are longer than those of some of the other strategies. The high number of hits observed by strategy (E) also illustrates the effectiveness of strategies based on the variation of the RCL parameter.

3 ALTERNATIVE CONSTRUCTION MECHANISMS

One possible shortcoming of the standard GRASP framework is the independence of its iterations, i.e., the fact that it does not learn from the history of solutions found in previous iterations. This is so because the basic algorithm discards information about any solution 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. In this section, we consider enhancements and alternative techniques for the construction phase of GRASP. They include Reactive GRASP, cost perturbations in place of randomized selection, bias functions, memory and learning, and local search on partially constructed solutions.

3.1 Reactive GRASP

A first possible strategy to incorporate a learning mechanism in the memoryless construction phase of the basic GRASP is the Reactive GRASP procedure introduced in Section 2. In this case, the value of the RCL parameter α is not fixed, but instead is 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 [78]. Let $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ be the set of possible values for α. The probabilities associated with the choice of each value are all initially made equal to $p_i = 1/m, i = 1, \ldots, m$. Furthermore, let z^* be the incumbent solution and let A_i be the average value of all solutions found using $\alpha = \alpha_i, i = 1, \ldots, m$. The selection probabilities are periodically reevaluated by taking $p_i = q_i / \sum_{j=1}^{m} q_j$, with $q_i = z^*/A_i$ for $i = 1, \ldots, m$. The value of q_i will be larger for values of $\alpha = \alpha_i$ leading to the best solutions on average. Larger values of q_i correspond to more suitable values for the parameter α. The probabilities associated with these more appropriate values will then increase when they are reevaluated.

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 [76–78], this approach has been used in power system transmission network planning [20] and in a capacitated location problem [29].

3.2 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 [25,26]. It adds more flexibility into algorithm design and may be even more effective than the greedy randomized construction of the basic GRASP procedure, in circumstances where the construction algorithms are not very sensitive to randomization. This is indeed the case for the

shortest-path heuristic of Takahashi and Matsuyama [95], used as one of the main building blocks of the construction phase of the hybrid GRASP procedure proposed by Ribeiro et al. [90] for the Steiner problem in graphs. Another situation where cost perturbations can be effective appears when no greedy algorithm is available for straight randomization. This happens to be the case of the hybrid GRASP developed by Canuto et al. [22] for the prize-collecting Steiner tree problem, which makes use of the primal-dual algorithm of Goemans and Williamson [52] to build initial solutions using perturbed costs.

In the case of the GRASP for the prize-collecting Steiner tree problem described in [22], a new solution is built at each iteration using node prizes updated by a perturbation function, according to the structure of the current solution. Two different prize perturbation schemes are used:

- *Perturbation by eliminations*: To enforce search diversification, the primal-dual algorithm used in the construction phase is driven to build a new solution without some of the nodes appearing in the solution constructed in the previous iteration. This is done by changing to zero the prizes of some persistent nodes, which appeared in the last solution built and remained at the end of the local search. A parameter α controls the fraction of the persistent nodes whose prizes are temporarily set to zero.
- *Perturbation by prize changes*: Another strategy to enforce the primal-dual algorithm to build different, but still good solutions, consists in introducing some noise into the node prizes, similarly to what is proposed in [25,26], so as to change the objective function. For each node i, a perturbation factor $\beta(i)$ is randomly generated in the interval $[1 - a,\ 1 + a]$,where a is an implementation parameter. The prize associated with node i is temporarily changed to $\overline{\pi}(i) = \pi(i) \cdot \beta(i)$, where $\pi(i)$ is its original prize.

The cost perturbation methods used in the GRASP for the minimum Steiner tree problem described in [90] incorporate learning mechanisms associated with intensification and diversification strategies, originally proposed in the context of tabu search. Let w_e denote the weight of edge e. Three distinct weight randomization methods (D, I, U) are applied. At a given GRASP iteration i, the modified weight w_e^i of each edge e is randomly selected from a uniform distribution between w_e and $r_i(e) \cdot w_e$, where the coefficient $r_i(e)$depends on the selected weight randomization method applied at iteration i. Let $t_{i-1}(e)$ be the number of locally optimal solutions in which edge e appeared, after $i - 1$ iterations of the hybrid GRASP procedure have been performed. Clearly, $0 \leq t_{i-1}(e) \leq i - 1$. Table 8.3 displays how the coefficients $r_i(e)$ are computed by each randomization method.

Table 8.3. Maximum randomization coefficients

Method	$r_i(e)$
D	$1.25 + 0.75 \cdot t_{i-1}(e)/(i-1)$
I	$2 - 0.75 \cdot t_{i-1}(e)/(i-1)$
U	2

In method *D*, values of the coefficients $r_i(e)$ are larger for edges which appeared more frequently in previously found local optima. This scheme leads to a diversification strategy, since more frequently used edges are likely to be penalized with stronger augmentations. Contrarily, method *I* is an intensification strategy penalizing less frequent edges with larger coefficients $r_i(e)$. Finally, the third randomization method *U* uses a uniform penalization strategy, independent of frequency information. The original weights without any penalization are used in the first three iterations, combined with three different construction heuristics. The weight randomization methods are then cyclically applied, one at each of the remaining iterations, starting with method *I*, next *D*, then *U*, then *I* again, and so on. The alternation between diversifying (method *D*) and intensifying (method *I*) iterations characterizes a strategic oscillation approach [49]. The experimental results reported in [90] show that the strategy combining these three perturbation methods is more robust than any of them used isolated, leading to the best overall results on a quite broad mix of test instances with different characteristics. The hybrid GRASP with path-relinking using this cost perturbation strategy is among the most effective heuristics currently available for the Steiner problem in graphs.

3.3 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 [21], where a family of such probability distributions is introduced. They are based on the rank $r(\sigma)$ assigned to each candidate element σ, according to its value of the greedy function. Several bias functions are introduced, 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: $\texttt{bias}(r) = e^{-r}$;
- polynomial bias of order n: $\texttt{bias}(r) = r^{-n}$.

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

$$\pi(\sigma) = \frac{\texttt{bias}(r(\sigma))}{\sum_{\sigma' \in \text{RCL}} \texttt{bias}(r(\sigma'))}. \tag{8.1}$$

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 [19]. Note that the standard GRASP uses a random bias function.

3.4 Intelligent Construction: Memory and Learning

Fleurent and Glover [46] observed that the basic GRASP does not use long-term memory (information gathered in previous iterations) and proposed a long-term memory

scheme to address this issue in multi-start heuristics. Long-term memory is one of the fundamentals on which tabu search relies.

Their scheme maintains a pool of elite solutions to be used in the construction phase. To become an elite solution, a solution must be either better than the best member of the pool, or better than its worst member and sufficiently different from the other solutions in the pool. For example, one can count identical solution vector components and set a threshold for rejection.

A *strongly determined variable* is one that cannot be changed without eroding the objective or changing significantly other valuables. A *consistent variable* is one that receives a particular value in a large portion of the elite solution set. Let $I(e)$ be a measure of the strongly determined and consistent features of solution element $e \in E$. Then, $I(e)$ becomes larger as e appears more often in the pool of elite solutions. The intensity function $I(e)$ is used in the construction phase as follows. Recall that $c(e)$ is the greedy function, i.e., the incremental cost associated with the incorporation of element $e \in E$ into the solution under construction. Let $K(e) = F(c(e), I(e))$ be a function of the greedy and the intensification functions. For example, $K(e) = \lambda c(e) + I(e)$. The intensification scheme biases selection from the RCL to those elements $e \in E$ with a high value of $K(e)$ by setting its selection probability to be $p(e) = K(e)/\sum_{s \in \mathrm{RCL}} K(s)$.

The function $K(e)$ can vary with time by changing the value of λ, e.g., initially 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 [46] and Binato et al. [19].

3.5 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" [50]. Fleurent and Glover [46] provided a GRASP interpretation of this principle. They suggested that imperfections introduced during steps of 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 is to apply local search during a few points in the construction phase and not during each construction iteration. In Binato et al. [19], local search is applied after 40% and 80% of the construction moves have been taken, as well as at the end of the construction phase.

4 PATH-RELINKING

Path-relinking is another enhancement to the basic GRASP procedure, leading to significant improvements in solution quality. Path-relinking was originally proposed by Glover [48] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [49–51]. Starting from one or more elite solutions, paths in the solution space leading towards other elite solutions are generated and explored in the search for better solutions. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions. Path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions, by favoring these attributes in the selected moves.

The 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 [62]. It was followed by several extensions, improvements, and successful applications [4,22,86,90]. Two basic strategies are used:

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

Applying path-relinking as an intensification strategy to each local optimum seems to be more effective than simply using it as a post-optimization step. In this context, path-relinking is applied to pairs (x_1, x_2) of solutions, where x_1 is the locally optimal solution obtained after local search and x_2 is one of a few elite solutions randomly chosen from a pool with a limited number `Max_Elite` of elite solutions found along the search. The pool is originally empty. Each locally optimal solution obtained by local search is considered as a candidate to be inserted into the pool if it is sufficiently different from every other solution currently in the pool. If the pool already has `Max_Elite` solutions and the candidate is better than the worst of them, then the former replaces the latter. If the pool is not full, the candidate is simply inserted.

The algorithm starts by computing the symmetric difference $\Delta(x_1, x_2)$ between x_1 and x_2, resulting in the set of moves which should be applied to one of them (the initial solution) to reach the other (the guiding solution). Starting from the initial solution, the best move from $\Delta(x_1, x_2)$ still not performed is applied to the current solution, until the guiding solution is attained. The best solution found along this trajectory is also considered as a candidate for insertion in the pool and the incumbent is updated. Several alternatives have been considered and combined in recent implementations:

- do not apply path-relinking at every GRASP iteration, but only periodically;
- explore two different trajectories, using first x_1, then x_2 as the initial solution;
- explore only one trajectory, starting from either x_1 or x_2; and
- do not follow the full trajectory, but instead only part of it (truncated path-relinking).

All these alternatives involve the trade-offs between computation time and solution quality. Ribeiro et al. [90] observed that exploring two different trajectories for each pair (x_1, x_2) takes approximately twice the time needed to explore only one of them, with very marginal improvements in solution quality. They have also observed that if only one trajectory is to be investigated, better solutions are found when path-relinking starts from the best among x_1 and x_2. Since the neighborhood of the initial solution is much more carefully explored than that of the guiding one, starting from the best of them gives the algorithm a better chance to investigate in more detail the neighborhood of the most promising solution. For the same reason, the best solutions are usually found closer to the initial solution than to the guiding solution, allowing pruning the relinking trajectory before the latter is reached.

Detailed computational results illustrating the trade-offs between these strategies for the problem of routing private virtual circuits in frame-relay services are reported by Resende and Ribeiro [86]. In this case, the set of moves corresponding to the symmetric

difference $\Delta(x_1, x_2)$ between any pair (x_1, x_2) of solutions is the subset of private virtual circuits routed through different routes (i.e., using different edges) in x_1 and x_2. We summarize below some of these results, obtained on an SGI Challenge computer (with 28 196-MHz MIPS R10000 processors) with 7.6 Gb of memory. We considered four variants of the GRASP and path-relinking schemes previously discussed:

- `G`: This variant is a pure GRASP with no path-relinking.
- `GPRf`: This variant adds to `G` a one-way (forward) path-relinking starting from a locally optimal solution and using a randomly selected elite solution as the guiding solution.
- `GPRb`: This variant adds to `G` a one way (backwards) path-relinking starting from a randomly selected elite solution and using a locally optimal solution as the guiding solution.
- `GPRfb`: This variant combines `GPRf` and `GPRb`, performing path-relinking in both directions.

These variants are evaluated and compared in terms of their trade-offs between computation time and solution quality.

To study the effect of path-relinking on GRASP, we compared the four variants on two instances: `att` and `fr750a`, see [86] for details. Two hundred independent runs for each variant were performed for each problem. Execution was terminated when a solution of value less than or equal to a given parameter value `look4` was found. The sub-optimal values chosen for this parameter were such that the slowest variant could terminate in a reasonable amount of computation time. Empirical probability distributions for the time to target solution value are plotted in Figures 8.9 and 8.10. To plot the empirical distribution for each algorithm and each instance, we associate with the i-th smallest running time t_i a probability $p_i = (i - \frac{1}{2})/200$, and plot the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, 200$. Due to the time taken by the pure GRASP procedure, we limited its plot in Figure 8.10 to 60 points.

These plots show a similar relative behavior of the four variants on the two instances. Since instance `fr750a` is harder for all variants and the associated computation times are longer, its plot is more discerning. For a given computation time, the probability of finding a solution at least as good as the target value increases from `G` to `GPRf`, from `GPRf` to `GPRfb`, and from `GPRfb` to `GPRb`. For example, there is a 9.25% probability for `GPRfb` to find a target solution value in less than 100s, while this probability increases to 28.75% for `GPRb`. For `G`, there is a 8.33% probability of finding a target solution value within 2000s, while for `GPRf` this probability increases to 65.25%. `GPRb` finds a target solution value in at most 129 s with 50% probability. For the same probability, this time increases to 172, 1727, and 10933 s, respectively, for variants `GPRfb`, `GPRf`, and `G`. In accordance with these results, variant `GPRb`, which does path-relinking backwards from an elite solution to a locally optimal solution, seems to be the most effective, confirming the preliminary findings reported in [90]. To further illustrate the behavior of GRASP and path-relinking, we depict in Figure 8.11 four plots representing the behavior of variant `GPRb` (GRASP with backwards path-relinking) on instance `att` with the variation of the target solution value. As before, 200 runs were performed for each target value decreasing from 126,600 to 126,000 by steps of 200. A similar behavior was observed for all other variants, with or without path-relinking, as well as for other instances and classes of test problems.

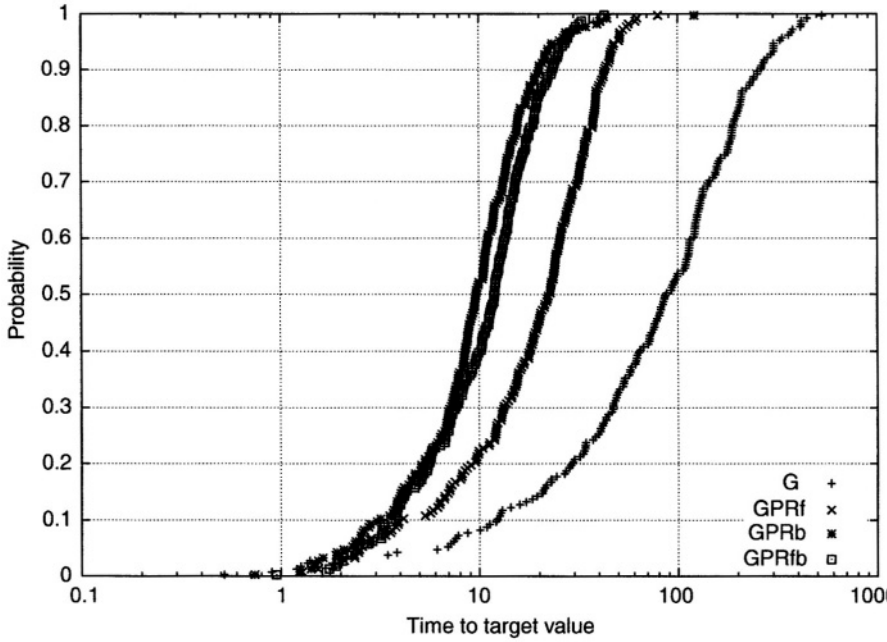

Figure 8.9. Empirical distributions of time to target solution value for GRASP, GRASP with forward path-relinking, GRASP with backwards path-relinking, and GRASP with back and forward path-relinking for instance `att`.

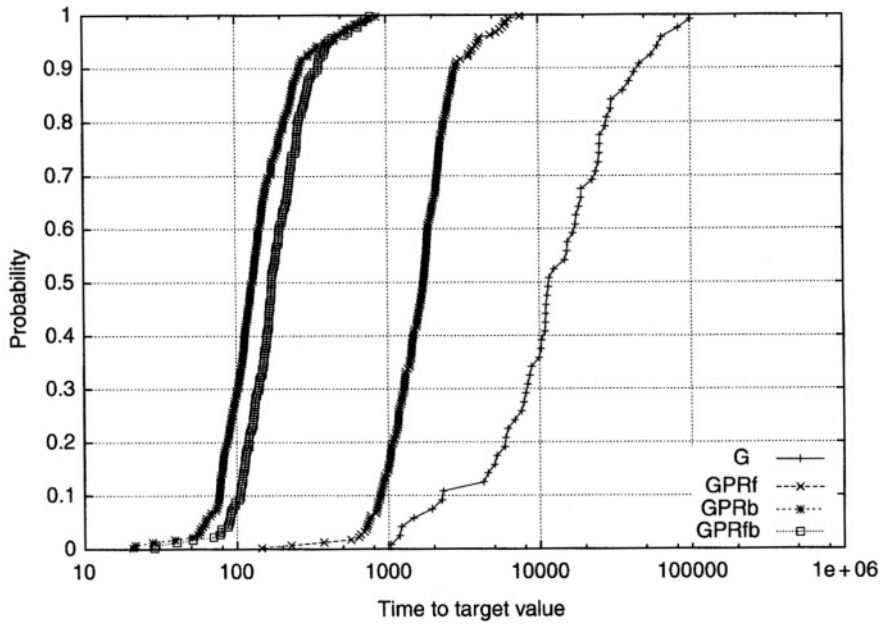

Figure 8.10. Empirical distributions of time to target solution value for GRASP, GRASP with forward path-relinking, GRASP with backwards path-relinking, and GRASP with back and forward path-relinking for instance `fr750a`.

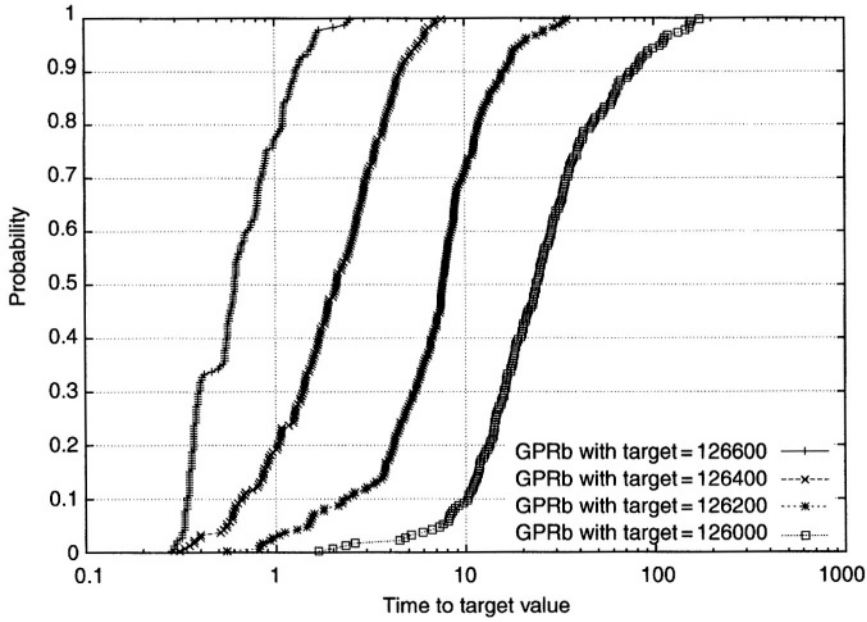

Figure 8.11. Empirical distributions of time to target solution value for GRASP with backwards path-relinking for instance `att` and different target values (`look4`).

Table 8.4. Solution values within fixed time limits over ten runs for instance `att`.

	10 s		100 s	
Variant	Best	Average	Best	Average
`GPR`	126602.883	126694.666	126227.678	126558.293
`GPRf`	126301.118	126578.323	126082.790	126228.798
`GPRb`	125960.336	126281.156	125665.785	125882.605
`GPRfb`	125961.118	126306.736	125646.460	125850.396

As a final experiment, once again we made use of the different GRASP variants for the problem of routing private virtual circuits to illustrate the effect of path-relinking in improving the solutions obtained by a pure GRASP approach, with only the construction and local search phases. This experiment was also performed using the same SGI Challenge computer (with 28,196-MHz MIPS R10000 processors) with 7.6 Gb of memory. For each of ten different seeds, we ran twice each variant for instance `att`, enforcing two different time limits: 10 and 100 s of processing time. The numerical results are reported in Table 8.4. For each variant and for each time limit, we give the average and the best solution values over the ten runs. We first note that both versions with backwards path-relinking performed systematically better, since they found better solutions for both time limits. Variants `GPRb` (GRASP with backwards path-relinking) and `GPRfb` (GRASP with path-relinking in both directions) showed similar behaviors, as it could be anticipated from the empirical probability distributions depicted in Figure 8.9. Variant `GPRb` obtained better results (in terms of both the average and the

best solution values found) within the time limit of 10 s, while variant `GPRfb` performed better for the time limit of 100 s. In the first case, `GPRb` found the best solution among the two variants in seven runs, while `GPRfb` did better for only two runs. However, when the time limit was increased to 100 s, `GPRb` found the best solutions in four runs, while `GPRfb` did better for five runs.

Path-relinking is a quite effective strategy to introduce memory in GRASP, leading to very robust implementations. The results reported above can be further illustrated by those obtained with the hybrid GRASP with path-relinking algorithm for the Steiner problem in graphs described in [90], which in particular improved the best known solutions for 33 out of the 41 still open problems in series i640 of the SteinLib repository [99] on May 1, 2001.

5 EXTENSIONS

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

The use of hashing tables to avoid cycling in conjunction with tabu search was proposed by Woodruff and Zemel [100]. A similar approach was later explored by Ribeiro et al. [88] in their tabu search algorithm for query optimization in relational databases. In the context of GRASP implementations, hashing tables were first used by Martins et al. [66] in their multineighborhood heuristic for the Steiner problem in graphs, to avoid the application of local search to solutions already visited in previous iterations.

Filtering strategies have also been used to speed up the iterations of GRASP, see, e.g., [40,66,78]. In these cases, local search is not applied to all solutions obtained at the end of the construction phase, but instead only to some promising unvisited solutions, defined by a threshold with respect to the incumbent.

Almost all randomization effort in the basic GRASP algorithm involves the construction phase. Local search stops at the first local optimum. On the other hand, strategies such as VNS (Variable Neighborhood Search), proposed by Hansen and Mladenović [54,69], rely almost entirely on the randomization of the local search to escape from local optima. With respect to this issue, GRASP and variable neighborhood strategies may be considered as complementary and potentially capable of leading to effective hybrid methods. A first attempt in this direction was done by Martins et al. [66]. The construction phase of their hybrid heuristic for the Steiner problem in graphs follows the greedy randomized strategy of GRASP, while the local search phase makes use of two different neighborhood structures as a VND procedure [54,69]. Their heuristic was later improved by Ribeiro et al. [90], one of the key components of the new algorithm being another strategy for the exploration of different neighborhoods. Ribeiro and Souza [89] also combined GRASP with VND in a hybrid heuristic for the degree-constrained minimum spanning tree problem. Festa et al. [45] studied different variants and combinations of GRASP and VNS for the MAX-CUT problem, finding and improving the best known solutions for some open instances from the literature.

GRASP has also been used in conjunction with genetic algorithms. Basically, the greedy randomized strategy used in the construction phase of a GRASP is applied to generate the initial population for a genetic algorithm. We may cite e.g., the genetic algorithm of Ahuja et al. [3] for the quadratic assignment problem, which makes use

of the GRASP proposed by Li et al. [63] to create the initial population of solutions. A similar approach was used by Armony et al. [11], with the initial population made up by both randomly generated solutions and those built by a GRASP.

The hybridization of GRASP with tabu search was first studied by Laguna and González-Velarde [61]. Delmaire et al. [29] 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 3.1, 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.

6 PARALLEL GRASP

Even though parallelism is not yet systematically used to speed up or to improve the effectiveness of metaheuristics, parallel implementations are very robust and abound in the literature; see e.g., Cung et al. [27] for a recent survey.

Most parallel implementations of GRASP follow the *multiple-walk independent thread* strategy, based on the distribution of the iterations over the processors [6,7,40, 63,65,67,70,73,74]. In general, each search thread has to perform `Max_Iterations`/p iterations, where p and `Max_Iterations` are, respectively, the number of processors and the total number of iterations. Each processor has a copy of the sequential algorithm, a copy of the problem data, and an independent seed to generate its own pseudorandom number sequence. To avoid that the processors find the same solutions, each of them must use a different sequence of pseudorandom numbers. A single global variable is required to store the best solution found over all processors. One of the processors acts as the master, reading and distributing problem data, generating the seeds which will be used by the pseudorandom number generators at each processor, distributing the iterations, and collecting the best solution found by each processor. Since the iterations are completely independent and very little information is exchanged, linear speedups are easily obtained provided that no major load imbalance problems occur. The iterations may be evenly distributed over the processors or according with their demands, to improve load balancing.

Martins et al. [67] implemented a parallel GRASP for the Steiner problem in graphs. Parallelization is achieved by the distribution of 512 iterations over the processors, with the value of the RCL parameter α randomly chosen in the interval [0.0,0.3] at each iteration. The algorithm was implemented in C on an IBM SP-2 machine with 32 processors, using the MPI library for communication. The 60 problems from series C, D, and E of the OR-Library [18] have been used for the computational experiments. The parallel implementation obtained 45 optimal solutions over the 60 test instances. The relative deviation with respect to the optimal value was never larger than 4%. Almost-linear speedups observed for 2, 4, 8, and 16 processors with respect to the sequential implementation are illustrated in Figure 8.12.

Path-relinking may also be used in conjunction with parallel implementations of GRASP. In the case of the multiple-walk independent-thread implementation described

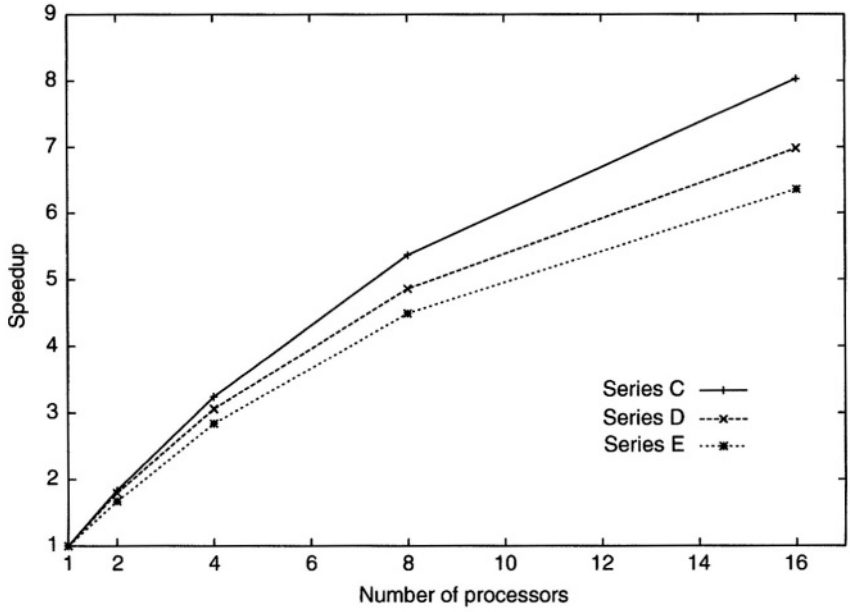

Figure 8.12. Average speedups on 2, 4, 8, and 16 processors.

by Aiex et al. [4] for the 3-index assignment problem, each processor applies path-relinking to pairs of elite solutions stored in a local pool. Computational results using MPI on an SGI Challenge computer with 28 R10000 processors showed linear speedups.

Alvim and Ribeiro [6,7] have shown that multiple-walk independent-thread approaches for the parallelization of GRASP may benefit much from load balancing techniques, whenever heterogeneous processors are used or if the parallel machine is simultaneously shared by several users. In this case, almost-linear speedups may be obtained with a heterogeneous distribution of the iterations over the p processors in $q \geq p$ packets. Each processor starts performing one packet $\lceil \texttt{Max\_Iterations}/q \rceil$ iterations and informs the master when it finishes its packet of iterations. The master stops the execution of each slave processor when there are no more iterations to be performed and collects the best solution found. Faster or less loaded processors will perform more iterations than the others. In the case of the parallel GRASP implemented for the problem of traffic assignment described in [78], this dynamic load balancing strategy allowed reductions in the elapsed times of up to 15% with respect to the times observed for the static strategy, in which the iterations were uniformly distributed over the processors.

The efficiency of multiple-walk independent-thread parallel implementations of metaheuristics, based on running multiple copies of the same sequential algorithm, has been addressed by some authors. A given target value τ for the objective function is broadcasted to all processors which independently execute the sequential algorithm. All processors halt immediately after one of them finds a solution with value at least as good as τ. The speedup is given by the ratio between the times needed to find a solution with value at least as good as τ, using respectively the sequential algorithm and the

parallel implementation with p processors. Some care is needed to ensure that no two iterations start with identical random number generator seeds. These speedups are linear for a number of metaheuristics, including simulated annealing [31,71]; iterated local search algorithms for the traveling salesman problem [33]; tabu search, provided that the search starts from a local optimum [17,94]; and WalkSAT [93] on hard random 3-SAT problems [56]. This observation can be explained if the random variable *time to find a solution within some target value* is exponentially distributed, as indicated by the following proposition [98].

Proposition 8.1. *Let $P_\rho(t)$ be the probability of not having found a given target solution value in t time units with ρ independent processes. If $P_1(t) = e^{-t/\lambda}$ with $\lambda \in \mathbf{R}^+$, corresponding to an exponential distribution, then $P_\rho(t) = e^{-\rho t/\lambda}$.*

This proposition follows from the definition of the exponential distribution. It implies that the probability $1 - e^{-\rho t/\lambda}$ of finding a solution within a given target value in time ρt with a sequential algorithm is equal to the probability of finding a solution at least as good as that in time t using ρ independent parallel processors. Hence, it is possible to achieve linear speedups in the time to find a solution within a target value by multiple independent processors. An analogous proposition can be stated for a two parameter (shifted) exponential distribution.

Proposition 8.2. *Let $P_\rho(t)$ be the probability of not having found a given target solution value in t time units with ρ independent processors. If $P_1(t) = e^{-(t-\mu)/\lambda}$ with $\lambda \in \mathbf{R}^+$ and $\mu \in \mathbf{R}^+$, corresponding to a two parameter exponential distribution, then $P_\rho(t) = e^{-\rho(t-\mu)/\lambda}$.*

Analogously, this proposition follows from the definition of the two-parameter exponential distribution. It implies that the probability of finding a solution within a given target value in time ρt with a sequential algorithm is equal to $1 - e^{-(\rho t-\mu)/\lambda}$, while the probability of finding a solution at least as good as that in time t using ρ independent parallel processors is $1 - e^{-\rho(t-\mu)/\lambda}$. If $\mu = 0$, then both probabilities are equal and correspond to the non-shifted exponential distribution. Furthermore, if $\rho\mu \ll \lambda$, then the two probabilities are approximately equal and it is possible to approximately achieve linear speedups in the time to find a solution within a target value using multiple independent processors.

Aiex et al. [5] have shown experimentally that the solution times for GRASP also have this property, showing that they fit a two-parameter exponential distribution. Figure 8.13 illustrates this result, depicting the superimposed empirical and theoretical distributions observed for one of the cases studied along the computational experiments reported by the authors, which involved 2400 runs of GRASP procedures for each of five different problems: maximum independent set [40,81], quadratic assignment [63,82], graph planarization [85,87], maximum weighted satisfiability [84], and maximum covering [79]. The same result still holds when GRASP is implemented in conjunction with a post-optimization path-relinking procedure [4].

In the case of *multiple-walk cooperative-thread* strategies, the search threads running in parallel exchange and share information collected along the trajectories they investigate. One expects not only to speed up the convergence to the best solution but, also, to find better solutions than independent-thread strategies. The most difficult aspect to be set up is the determination of the nature of the information to be shared or exchanged to improve the search, without taking too much additional memory or time to

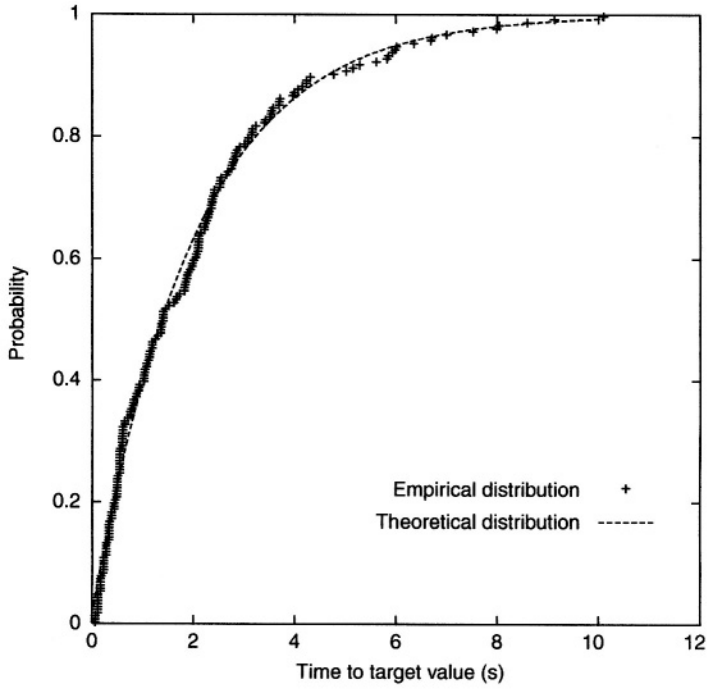

Figure 8.13. Superimposed empirical and theoretical distributions (times to target values measured in seconds on an SGI Challenge computer with 28 processors).

be collected. Cooperative-thread strategies may be implemented using path-relinking, by combining elite solutions stored in a central pool with the local optima found by each processor at the end of each GRASP iteration. Canuto et al. [22] used path-relinking to implement a parallel GRASP for the prize-collecting Steiner tree problem. Their strategy is truly cooperative, since pairs of elite solutions from a centralized unique central pool are distributed to the processors which perform path-relinking in parallel. Computational results obtained with an MPI implementation running on a cluster of 32,400-MHz Pentium II processors showed linear speedups, further illustrating the effectiveness of path-relinking procedures used in conjunction with GRASP to improve the quality of the solutions found by the latter.

7 APPLICATIONS

The first application of GRASP described in the literature concerns the set covering problem [38]. The reader is referred to Festa and Resende [44] for an annotated bibliography of GRASP and its applications. We conclude this chapter by summarizing below some references focusing the main applications of GRASP to problems in different areas:

- routing [9,12,16,24,59];
- logic [30,74,80,83];
- covering and partition [8,10,38,47,53];

- location [1,29,57,96,97];
- minimum Steiner tree [23,65–67,90];
- optimization in graphs [2,40,60,72,79,85,87];
- assignment [37,46,63,64,68,70,73,75,78];
- timetabling, scheduling, and manufacturing [13–15,19,28,32,34–36,41,42,58,91, 92,101];
- transportation [9,34,37];
- power systems [20];
- telecommunications [2,11,57,64,78,79,86];
- graph and map drawing [43,62,85,87]; and
- VLSI [8], among other areas of application.

8 CONCLUDING REMARKS

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

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

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

These and other extensions make up a set of tools that can be added to simpler heuristics to find better-quality solutions. To illustrate the effect of additional extensions on solution quality, Figure 8.14 shows some results obtained for the prize-collecting Steiner tree problem, as discussed in [22]. We consider the 40 instances of series C. The lower curve represents the results obtained exclusively with the primal-dual constructive algorithm (GW) of Goemans and Williamson [52]. The second curve shows the quality of the solutions produced with an additional local search (GW + LS), corresponding to the first iteration of GRASP. The third curve is associated with the results obtained after 500 iterations of GRASP with path-relinking (GRASP + PR). Finally, the top curve shows the results found by the complete algorithm, using variable neighborhood search as a post-optimization procedure (GRASP + PR + VNS). For a given relative deviation with respect to the optimal value, each curve indicates the number of instances for which the corresponding algorithm found a solution within that quality range. For example, we observe that the number of optimal solutions found goes from six, using only the constructive algorithm, to a total of 36, using the complete

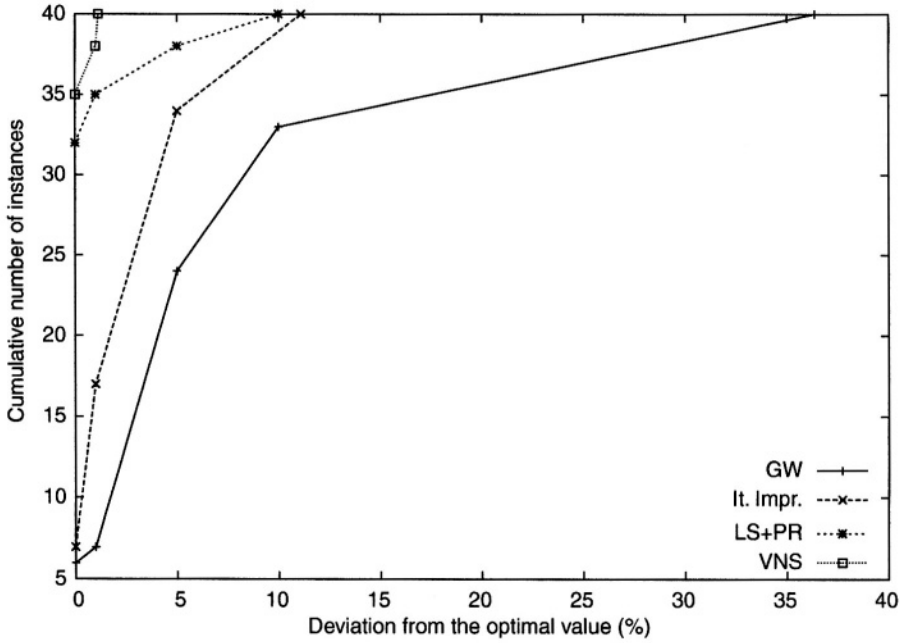

Figure 8.14. Performance of GW and successive variants of local search for Series C problems.

algorithm described in [22]. 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 see the contribution made by each additional extension.

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

BIBLIOGRAPHY

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

[2] J. Abello, P.M. Pardalos and M.G.C. Resende (1999) On maximum clique problems in very large graphs. In: J. Abello and J. Vitter (eds.), *External Memory Algorithms and Visualization,* volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, pp. 199–130.

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

[4] R.M. Aiex, M.G.C. Resende, P.M. Pardalos and G. Toraldo (2000) GRASP with path-relinking for the three-index assignment problem. Technical report, AT&T Labs–Research.

[5] R,M. Aiex, M.G.C. Resende and C.C. Ribeiro (2002) Probability distribution of solution time in GRASP: an experimental investigation. *Journal of Heuristics,* **8**, 343–373.

[6] A.C. Alvim (1998) Parallelization strategies for the metaheuristic GRASP. Master's thesis, Department of Computer Science, Catholic University of Rio de Janeiro, Brazil (in Portuguese).

[7] A.C. Alvim and C.C. Ribeiro (1998) Load balancing for the parallelization of the GRASP metaheuristic. In: *Proceedings of the X Brazilian Symposium on Computer Architecture,* Búzios, pp. 279–282 (in Portuguese).

[8] S. Areibi and A. Vannelli (1997) A GRASP clustering technique for circuit partitioning. In: J. Gu and P.M. Pardalos (eds.), *Satisfiability Problems,* Volume 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, pp. 711–724.

[9] M.F. Argüello, J.F. Bard and G. Yu (1997) A GRASP for aircraft routing in response to groundings and delays. *Journal of Combinatorial Optimization,* **1**, 211–228.

[10] M.F. Argüello, T.A. Feo and O. Goldschmidt (1996) Randomized methods for the number partitioning problem. *Computers and Operations Research,* **23**, 103–111.

[11] M. Armony, J.C. Klincewicz, H. Luss and M.B. Rosenwein (2000) Design of stacked self-healing rings using a genetic algorithm. *Journal of Heuristics,* **6**, 85–105.

[12] J.B. Atkinson (1998) A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *Journal of the Operational Research Society,* **49**, 700–708.

[13] J.F. Bard and T.A. Feo (1989) Operations sequencing in discrete parts manufacturing. *Management Science,* **35**, 249–255.

[14] J.F. Bard and T.A. Feo (1991) An algorithm for the manufacturing equipment selection problem. *IIE Transactions,* **23**, 83–92.

[15] J.F. Bard, T.A. Feo and S. Holland (1996) A GRASP for scheduling printed wiring board assembly. *IIE Transactions,* **28**, 155–165.

[16] J.F. Bard, L. Huang, P. Jaillet and M. Dror (1998) A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Science,* **32**, 189–203.

[17] R. Battiti and G. Tecchiolli (1992) Parallel biased search for combinatorial optimization: Genetic algorithms and tabu. *Microprocessors and Microsystems,* **16**, 351–367.

[18] J.E. Beasley (1990) OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society,* **41**, 1069–1072.

[19] S. Binato, W.J. Hery, D, Loewenstern and M.G.C. Resende (2002) A GRASP for job shop scheduling. In: C.C. Ribeiro and P. Hansen (eds.), *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, pp. 59–79.

[20] S. Binato and G.C. Oliveira (2002) A reactive GRASP for transmission network expansion planning. In: C.C. Ribeiro and P. Hansen (eds.), *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, pp. 81–100.

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

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

[23] S.A. Canuto, C.C. Ribeiro and M.G.C. Resende (1999) Local search with perturbations for the prize-collecting Steiner tree problem. In: *Extended Abstracts of the Third Metaheuristics International Conference.* Angra dos Reis, pp. 115–119.

[24] C. Carreto and B. Baker (2002) A GRASP interactive approach to the vehicle routing problem with backhauls. In: C.C. Ribeiro and P. Hansen (eds.), *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, pp. 185–199.

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

[26] I. Charon and O. Hudry (2002) The noising methods: a survey. In: C.C. Ribeiro and C.C. Ribeiro (eds.), *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, pp. 245–261.

[27] V.-D. Cung, S.L. Martins, C.C. Ribeiro and C. Roucairol (2002) Strategies for the parallel implementation of metaheuristics. In: C.C. Ribeiro and C.C. Ribeiro (eds.), *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, pp. 263–308.

[28] P. De, J.B. Ghosj and C.E. Wells (1994) Solving a generalized model for con due date assignment and sequencing. *International Journal of Production Economics,* **34**, 179–185.

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

[30] A.S. Deshpande and E. Triantaphyllou (1998) 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.

[31] N. Dodd (1990) Slow annealing versus multiple fast annealing runs: an empirical investigation. *Parallel Computing,* **16**, 269–272.

[32] A. Drexl and F. Salewski (1997) Distribution requirements and compactness constraints in school timetabling. *European Journal of Operational Research,* **102**, 193–214.

[33] H.T. Eikelder, M. Verhoeven, T. Vossen and E. Aarts (1996) A probabilistic analysis of local search. In: I. Osman and J. Kelly (eds.), *Metaheuristics: Theory and Applications.* Kluwer Academic Publishers, pp. 605–618.

[34] T.A. Feo and J.F. Bard (1989) Flight scheduling and maintenance base planning. *Management Science,* **35**, 1415–1432.

[35] T.A. Feo and J.F. Bard (1989) The cutting path and tool selection problem in computer-aided process planning. *Journal of Manufacturing Systems,* **8**, 17–26.

[36] T.A. Feo, J.F. Bard and S. Holland (1995) Facility-wide planning and scheduling of printed wiring board assembly. *Operations Research,* **43**, 219–230.

[37] T.A. Feo and J.L. González-Velarde (1995) The intermodal trailer assignment problem: models, algorithms, and heuristics. *Transportation Science,* **29**, 330–341.

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

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

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

[41] T.A. Feo, K. Sarathy and J. McGahan (1996) A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers and Operations Research,* **23**, 881–895.

[42] T.A. Feo, K. Venkatraman and J.F. Bard (1991) A GRASP for a difficult single machine scheduling problem. *Computers and Operations Research,* **18**, 635–643.

[43] E. Fernández and R. Martí (1999) GRASP for seam drawing in mosaicking of aerial photographic maps. *Journal of Heuristics,* **5**, 181–197.

[44] P. Festa and M.G.C. Resende (2002) GRASP: an annotated bibliography. In: C.C. Ribeiro and P. Hansen (eds.), *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, pp. 325–367.

[45] P. Festa, M.G.C. Resende, P. Pardalos and C.C. Ribeiro (2001) GRASP and VNS for Max-Cut. In: *Extended Abstracts of the Fourth Metaheuristics International Conference.* Porto, pp. 371–376.

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

[47] J.B. Ghosh (1996) Computatinal aspects of the maximum diversity problem. *Operations Research Letters,* **19**, 175–181.

[48] F. Glover (1996) Tabu search and adaptive memory programming—advances, applications and challenges. In: R.S. Barr, R.V. Helgason and J.L. Kennington (eds.), *Interfaces in Computer Science and Operations Research.* Kluwer, pp. 1–75.

[49] F. Glover (2000) Multi-start and strategic oscillation methods—principles to exploit adaptive memory. In: M. Laguna and J.L. Gonzáles-Velarde (eds.), *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research.* Kluwer, pp. 1–24.

[50] F. Glover and M. Laguna (1997) *Tabu Search.* Kluwer.

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

[52] M.X. Goemans and D.P. Williamson (1996) The primal dual method for approximation algorithms and its application to network design problems. In: D. Hochbaum (ed.), *Approximation Algorithms for NP-hard Problems.* PWS Publishing Co., pp. 144–191.

[53] P.L. Hammer and D.J. Rader, Jr. (2001) Maximally disjoint solutions of the set covering problem. *Journal of Heuristics,* **7**, 131–144.

[54] P. Hansen and N. Mladenović (2002) Developments of variable neighborhood search. In: C.C. Ribeiro and P. Hansen (eds.), *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, pp. 415–439.

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

[56] H. Hoos and T. Stützle (1999) Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence,* **112**, 213–232.

[57] J.G. Klincewicz (1992) Avoiding local optima in the p–hub location problem using tabu search and GRASP. *Annals of Operations Research,* **40**, 283–302.

[58] J.G. Klincewicz and A. Rajan (1994) Using GRASP to solve the component grouping problem. *Naval Research Logistics,* **41**, 893–912.

[59] G. Kontoravdis and J.F. Bard (1995) A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing,* **7**, 10–23.

[60] M. Laguna, T.A. Feo and H.C. Elrod (1994) A greedy randomized adaptive search procedure for the two–partition problem. *Operations Research,* **42**, 677–687.

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

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

[63] Y. Li, P.M. Pardalos and M.G.C. Resende (1994) A greedy randomized adaptive search procedure for the quadratic assignment problem. In: P.M. Pardalos and H. Wolkowicz (eds.), *Quadratic Assignment and Related Problems,* volume 16 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, pp. 237–261.

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

[65] S.L. Martins, P.M. Pardalos, M.G.C. Resende and C.C. Ribeiro (1999) Greedy randomized adaptive search procedures for the steiner problem in graphs. In: P.M. Pardalos, S. Rajasekaran and J. Rolim (eds.), *Randomization Methods in*

Algorithmic Design, volume 43 *of DIMACS Series on Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, pp. 133–145.

[66] S.L. Martins, M.G.C. Resende, C.C. Ribeiro and P. Pardalos (2000) A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization,* **17**, 267–283.

[67] S.L. Martins, C.C. Ribeiro and M.C. Souza (1998) A parallel GRASP for the Steiner problem in graphs. In: A. Ferreira and J. Rolim (eds.), *Proceedings of IRREGULAR'98—5th International Symposium on Solving Irregularly Structured Problems in Parallel,* volume 1457 of *Lecture Notes in Computer Science.* Springer-Verlag, pp. 285–297.

[68] T. Mavridou, P.M. Pardalos, L.S. Pitsoulis and M.G.C. Resende (1998) A GRASP for the biquadratic assignment problem. *European Journal of Operational Research,* **105**, 613–621.

[69] N. Mladenović and P. Hansen (1997) Variable neighborhood search. *Computers and Operations Research,* **24**, 1097–1100.

[70] R.A. Murphey, P.M. Pardalos and L.S. Pitsoulis (1998) A parallel GRASP for the data association multidimensional assignment problem. In: P.M. Pardalos (ed.), *Parallel Processing of Discrete Problems,* volume 106 of *The IMA Volumes in Mathematics and Its Applications,* Springer-Verlag, pp. 159–180.

[71] L. Osborne and B. Gillett (1991) A comparison of two simulated annealing algorithms applied to the directed Steiner problem on networks. *ORSA Journal on Computing,* **3**, 213–225.

[72] P.M. Pardalos, T. Qian and M.G.C. Resende (1999) A greedy randomized adaptive search procedure for the feedback vertex set problem. *Journal of Combinatorial Optimization,* **2**, 399–412.

[73] P.M. Pardalos, L.S. Pitsoulis and M.G.C. Resende (1995) A parallel GRASP implementation for the quadratic assignment problem. In: A. Ferreira and J. Rolim (eds.), *Parallel Algorithms for Irregularly Structured Problems—Irregular'94.* Kluwer Academic Publishers, pp. 115–133.

[74] P.M. Pardalos, L.S. Pitsoulis and M.G.C. Resende (1996) A parallel GRASP for MAX-SAT problems. *Lecture Notes in Computer Science,* **1184**, 575–585.

[75] L.S. Pitsoulis, P.M. Pardalos and D.W. Hearn (2001) Approximate solutions to the turbine balancing problem. *European Journal of Operational Research,* **130**, 147–155.

[76] M. Prais and C.C. Ribeiro (1999) Parameter variation in GRASP implementations. In: *Extended Abstracts of the Third Metaheuristics International Conference.* Angra dos Reis, pp. 375–380.

[77] M. Prais and C.C. Ribeiro (2000) Parameter variation in GRASP procedures. *Investigación Operativa,* **9**, 1–20.

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

[79] M.G.C. Resende (1998) Computing approximate solutions of the maximum covering problem using GRASP. *Journal of Heuristics,* **4**, 161–171.

[80] M.G.C. Resende and T.A. Feo (1996) A GRASP for satisfiability. In: D.S. Johnson and M.A. Trick (eds.), *Cliques, Coloring, and Satisfiability: The Second DIMACS Implementation Challenge,* volume 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, pp. 499–520.

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

[82] M.G.C. Resende, P.M. Pardalos and Y. Li (1996) Algorithm 754: Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software,* **22,** 104–118.

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

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

[85] M.G.C. Resende and C.C. Ribeiro (1997) A GRASP for graph planarization. *Networks,* **29**, 173–189.

[86] M.G.C. Resende and C.C. Ribeiro (2001) A GRASP with path-relinking for private virtual circuit routing. Technical report, AT&T Labs Research.

[87] C.C. Ribeiro and M.G.C. Resende (1999) Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP. *ACM Transactions on Mathematical Software,* **25**, 342–352.

[88] C.C. Ribeiro, C.D. Ribeiro and R.S. Lanzelotte (1997) Query optimization in distributed relational databases. *Journal of Heuristics,* **3**, 5–23.

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

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

[91] R.Z. Ríos-Mercado and J.F. Bard (1998) Heuristics for the flow line problem with setup costs. *European Journal of Operational Research,* 76–98.

[92] R.Z. Ríos-Mercado and J.F. Bard (1999) An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup costs. *Journal of Heuristics,* **5**, 57–74.

[93] B. Selman, H. Kautz and B. Cohen (1994) Noise strategies for improving local search. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence.* Seattle, MIT Press, pp. 337–343.

[94] E. Taillard (1991) Robust taboo search for the quadratic assignment problem. *Parallel Computing,* **7**, 443–455.

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

[96] T.L. Urban (1998) Solution procedures for the dynamic facility layout problem. *Annals of Operations Research,* 323–342.

[97] T.L. Urban, W.-C. Chiang and R.A. Russel (2000) The integrated machine allocation and layout problem. *International Journal of Production Research,* 2913–2930.

[98] M.G.A. Verhoeven and E.H.L. Aarts (1995) Parallel local search. *Journal of Heuristics,* **1**, 43–65.

[99] S. Voss, A. Martin and T. Koch (2001) Steinlib testdata library. Online document at http://elib.zib.de/steinlib/steinlib.html, last visited on May 1.

[100] D.L. Woodruff and E. Zemel (1993) Hashing vectors for tabu search. *Annals of Operations Research,* **41,** 123–137.

[101] J. Yen, M. Carlsson, M. Chang, J.M. Garcia and H. Nguyen (2000) Constraint solving for inkjet print mask design. *Journal of Imaging Science and Technology,* **44**, 391–397.