Chapter 2

GRASP WITH PATH-RELINKING: RECENT ADVANCES AND APPLICATIONS

Mauricio G.C. Resende[1] and Celso C. Ribeiro[2]

[1] *Internet and Network Systems Research, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.*
mgcr@research.att.com

[2] *Department of Computer Science, Catholic University of Rio de Janeiro, Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ 22453-900 Brazil*
celso@inf.puc-rio.br

Abstract: Path-relinking is a major enhancement to the basic greedy randomized adaptive search procedure (GRASP), leading to significant improvements in solution time and quality. Path-relinking adds a memory mechanism to GRASP by providing an intensification strategy that explores trajectories connecting GRASP solutions and the best elite solutions previously produced during the search. This paper reviews recent advances and applications of GRASP with path-relinking. A brief review of GRASP is given. This is followed by a description of path-relinking and how it is incorporated into GRASP. Several recent applications of GRASP with path-relinking are reviewed. The paper concludes with a discussion of extensions to this strategy, concerning in particular parallel implementations and applications of path-relinking with other metaheuristics.

Keywords: Metaheuristics, GRASP, path-relinking.

2.1 INTRODUCTION

GRASP (*Greedy Randomized Adaptive Search Procedure*) is a metaheuristic for finding approximate solutions to combinatorial optimization problems formulated as

$$\min f(x) \text{ subject to } x \in X,$$

where $f(\cdot)$ is an objective function to be minimized and X is a discrete set of feasible solutions. It was first introduced by Feo and Resende [14] in a paper describing a probabilistic heuristic for set covering. Since then, GRASP has experienced continued development [15, 43] and has been applied in a wide range of problem areas [18].

2.1.1 Multi-start local search

GRASP can be thought of as a search method that repeatedly applies local search from different starting solutions in X. At each step of local search, the neighborhood $N(x)$ of the current solution x is searched for a solution $y \in N(x)$ such that $f(y) < f(x)$. If such an improving solution is found, it is made the current solution and another step of local search is done. If no improving solution is found, the procedure stops with x as a locally optimal solution.

An obvious initial solution for local search is a solution generated by a greedy algorithm. Greedy algorithms construct a solution one element at a time. For example, a tree is built one edge at a time; a schedule is built one operation at a time; a vertex partition is built one vertex at a time. At each step of a greedy algorithm, a set of candidate elements C contains all elements that can be added to extend the partial solution. Greedy algorithms make use of a greedy function $g(e)$ that measures the incremental cost of adding element $e \in C$ to the current partial solution. For a minimization problem, the element $e^* = \text{argmin}\{g(e) : e \in C\}$ is chosen to be added to the partial solution. The addition of e^* to the partial solution usually restricts the set of candidates elements, which is reflected by the reduction of its cardinality. The procedure ends when a complete solution is built, i.e. when $C = \emptyset$.

The drawback of using a greedy algorithm as an initial solution for local search is that if a deterministic rule is used to break ties, a greedy algorithm will produce a single solution and therefore local search can only be applied once. Even when a probabilistic tie breaking rule is used, the diversity of purely greedy solutions is usually low.

The other extreme is to repeatedly start local search from randomly generated solutions. Though this approach produces a high level of diversity in the starting solutions, the average quality of these random solutions is usually much worse than that of a greedy solution. Furthermore, the time local search takes to converge to a locally optimal solution is, on average, much longer than when a greedy initial solution is used.

GRASP blends greedy and random construction either by using greediness to build a restricted candidate list (RCL) and randomness to select

an element from the list, or by using randomness to build the list and greediness for selection. Candidate elements $e \in C$ are sorted according to their greedy function value $g(e)$. In a cardinality-based RCL, the latter is made up by the k top-ranked elements. In a value-based construction, the RCL consists of the elements in the set $\{e \in C : g_* \leq g(e) \leq g_* + \alpha \cdot (g^* - g_*)\}$, where $g_* = \min\{g(e) : e \in C\}$, $g^* = \max\{g(e) : e \in C\}$, and α is a parameter satisfying $0 \leq \alpha \leq 1$. Since the best value for α is often difficult to determine, it is often assigned a random value for each GRASP iteration.

Algorithm 2.1 shows the pseudo-code for a pure greedy randomized adaptive search procedure. The value of the best solution is stored in f^* and $i_{\max}$ GRASP iterations are done. Each iteration consists of a greedy randomized construction phase, followed by a local search phase, starting from the greedy randomized solution. If the solution resulting from the local search improves the best solution so far, it is stored in x^*.

```
Data  : Number of iterations i_max
Result: Solution x* ∈ X
f* ← ∞;
for i = 1, ..., i_max do
    x ← GreedyRandomizedConstruction();
    x ← LocalSearch(x);
    if f(x) < f* then
        f* ← f(x);
        x* ← x;
    end
end
```

Algorithm 2.1: A basic GRASP for minimization.

Figure 2.1 displays results for an instance of the maximum covering problem [36], showing the distribution of objective function values for the construction phase and the local search phase of a purely random multi-start algorithm (followed by local search) and a GRASP with the parameter α fixed at 0.85. In both plots, the iterates have been sorted by the objective function value of the solution found by local search. The plots show that the GRASP construction achieves a reasonably amount of diversity in terms of solution values, while producing starting solutions for local search that have much better objective function values. The objective function values are situated around 3.5 for the random construction and 9.7 for GRASP construction, while the value obtained by local search are around 9.9. Consequently, the local search times are much smaller for GRASP than for the purely random multi-start algorithm.

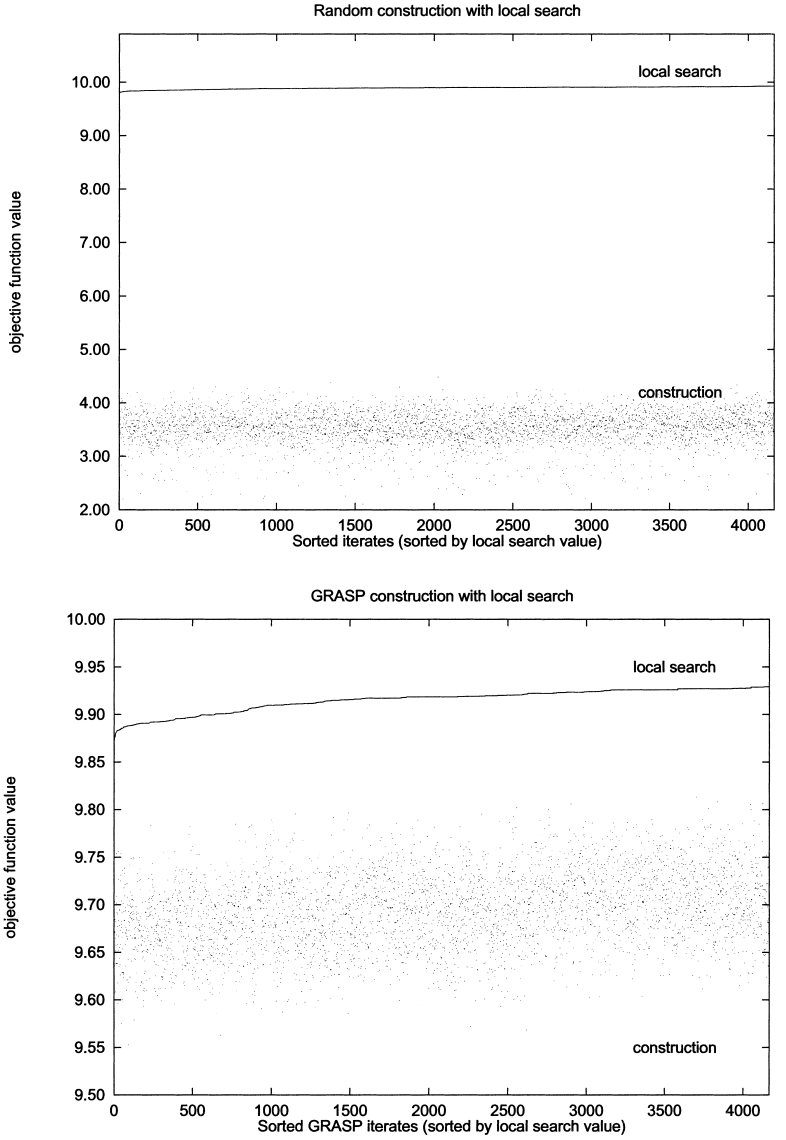

Figure 2.1. Random multi-start vs. GRASP on an instance of maximum covering problem.

Figure 2.2 shows, with results for 100 runs on the same instance of a maximum satisfiability problem, the benefit of using GRASP instead of repeatedly restarting local search with a randomly generated solution and a greedy solution. Two curves compare objective function value (best and average over the 100 runs) for different values of the RCL

parameter α. Two other curves compare solution times (average total time and average local search time) for different values of α. Since this is a maximization problem, $\alpha = 0$ corresponds to random construction, while $\alpha = 1$ corresponds to greedy construction. While the average solution improves as we move from random to greedy, the best solution (what we are really interested in) improves as we move away from random construction, but reaches a maximum before reaching $\alpha = 1$, and then decreases after that. As the mean solution increases, the spread of solutions decreases. The combination of the increase in mean solution value and the presence of enough spread contribute to produce the best solution with $\alpha = .8$. Solution times decrease as one moves from random to greedy and this is mainly due to the decrease in time for local search.

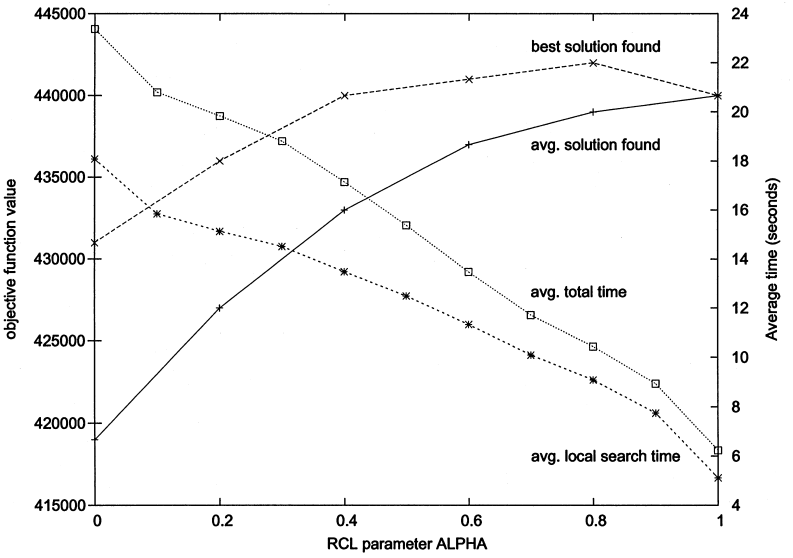

Figure 2.2. Average versus best solution found and total running time versus local search time as a function of the RCL parameter α on 100 runs on an instance of maximum satisfiability.

2.1.2 Memory-based mechanisms

If GRASP iteration i uses the random number generator seed s_i, then the iterations are memoryless, i.e. they produce the same result independently of the order in which they are run. In the remainder of this section, we review how the use of memory was introduced into GRASP.

Memory can be used to avoid doing redundant work. For example, one can store in a hash table all solutions constructed and used as initial

solutions for local search [30]. Every time a new solution is constructed, it will only be used as an initial solution in the local search phase if it is not present in the hash table.

Filtering of constructed solutions [16, 30, 34] avoids applying local search to low-quality solutions, where local search will probably take long to converge to a low-quality local optimum.

Fleurent and Glover [19] introduced a long-term memory mechanism in GRASP construction that makes use of a set of elite solutions found during the GRASP iterations. Their mechanism favors (strongly determined) variables that cannot be changed without eroding the objective or changing significantly other variables and (consistent) variables that receive a particular value in a large portion of the elite solutions.

Prais and Ribeiro [34] introduced another learning mechanism for GRASP construction, which they named *reactive* GRASP. Recall that in a value-based restricted candidate list a parameter α determines the level of randomness or greediness used to make up the RCL. Instead of using a fixed value for α, reactive GRASP selects a value, at random, from a discrete set of values $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Each value α_i has associated with it a probability p_i that it will be selected ($\sum_{i=1}^{m} p_i = 1$). The idea in reactive GRASP is to change these probabilities as the iterations proceed, to favor values that have led to better solutions in previous GRASP iterations.

Laguna and Martí [28] introduced another strategy for using long-term memory consisting of a set of elite solutions. At each GRASP iteration, this strategy combines the GRASP solution with a randomly selected elite solution, using path-relinking [23]. This is the subject of the next section.

2.2 PATH-RELINKING

Path-relinking was originally proposed by Glover [23] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [24–26]. Starting from one or more elite solutions, paths in the solution space leading toward other elite solutions are generated and explored in the search for better solutions. To generate paths, moves are selected to introduce attributes in the current solution that are present in the elite guiding solution. Path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions, by favoring these attributes in the selected moves.

Algorithm 2.2 illustrates the pseudo-code of the path-relinking procedure applied to a pair of solutions x_s (starting solution) and x_t (target solution).

Data : Starting solution x_s and target solution x_t
Result: Best solution x^* in path from x_s to x_t
Compute symmetric difference $\Delta(x_s, x_t)$;
$f^* \leftarrow \min\{f(x_s), f(x_t)\}$;
$x^* \leftarrow \text{argmin}\{f(x_s), f(x_t)\}$;
$x \leftarrow x_s$;
while $\Delta(x, x_t) \neq \emptyset$ **do**
  $m^* \leftarrow \text{argmin}\{f(x \oplus m) : m \in \Delta(x, x_t)\}$;
  $\Delta(x \oplus m^*, x_t) \leftarrow \Delta(x, x_t) \setminus \{m^*\}$;
  $x \leftarrow x \oplus m^*$;
  if $f(x) < f^*$ **then**
    $f^* \leftarrow f(x)$;
    $x^* \leftarrow x$;
  end
end

Algorithm 2.2: Path-relinking.

The procedure starts by computing the symmetric difference $\Delta(x_s, x_t)$ between the two solutions, i.e. the set of moves needed to reach x_t (target solution) from x_s (initial solution). A path of solutions is generated linking x_s and x_t. The best solution x^* in this path is returned by the algorithm. At each step, the procedure examines all moves $m \in \Delta(x, x_t)$ from the current solution x and selects the one which results in the least cost solution, i.e. the one which minimizes $f(x \oplus m)$, where $x \oplus m$ is the solution resulting from applying move m to solution x. The best move m^* is made, producing solution $x \oplus m^*$. The set of available moves is updated. If necessary, the best solution x^* is updated. The procedure terminates when x_t is reached, i.e. when $\Delta(x, x_t) = \emptyset$.

We notice that path-relinking may also be viewed as a constrained local search strategy applied to the initial solution x_s, in which only a limited set of moves can be performed and where uphill moves are allowed. Several alternatives have been considered and combined in recent implementations of path-relinking [1–3, 7, 48, 50, 52]:

- *periodical relinking:* path-relinking is not systematically applied, but instead only periodically;
- *forward relinking:* path-relinking is applied using the worst among x_s and x_t as the initial solution and the other as the target solution;
- *backward relinking:* the roles of x_s and x_t are interchanged, path-relinking is applied using the best among x_s and x_t as the initial solution and the other as the target solution;

- *back and forward relinking:* two different trajectories are explored, the first using x_s as the initial solution and the second using x_t in this role;
- *mixed relinking:* two paths are simultaneously explored, the first emanating from x_s and the second from x_t, until they meet at an intermediary solution equidistant from x_s and x_t;
- *randomized relinking:* instead of selecting the best yet unselected move, randomly select one from among a candidate list with the most promising moves in the path being investigated; and
- *truncated relinking:* the full trajectory between x_s and x_t is not investigated, but instead only part of it.

All these alternatives involve trade-offs between computation time and solution quality. Ribeiro et al. [50] observed that exploring two different trajectories for each pair (x_s, x_t) 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 the relinking procedure starts from the best among x_s and x_t. 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 the pruning of the relinking trajectory before the latter is reached.

2.3 GRASP WITH PATH-RELINKING

Path-relinking is a major enhancement to the basic GRASP procedure, leading to significant improvements in solution time and quality.

The use of path-relinking within a GRASP procedure, as an intensification strategy applied to each locally optimal solution, was first proposed by Laguna and Martí [28]. It was followed by several extensions, improvements, and successful applications [3, 9, 43, 45, 50]. Two basic strategies are used:

- path-relinking is applied to all pairs of elite solutions, either periodically during the GRASP iterations or after all GRASP iterations have been performed as a post-optimization step; 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 only as a post-optimization step. In general, combining intensification with post-optimization results in the best strategy. In the context of intensification, path-relinking is applied to pairs (x, y) of solutions, where x is a locally optimal solution produced by each GRASP iteration after local search and y is one of a few elite solutions randomly chosen from a pool with a limited number `Max_Elite` of elite solutions found along the search. Uniform random selection is a simple strategy to implement. Since the symmetric difference is a measure of the length of the path explored during relinking, a strategy biased toward pool elements y with high symmetric difference with respect to x is usually better than one using uniform random selection [45].

The pool is originally empty. Since we wish to maintain a pool of good but diverse solutions, each locally optimal solution obtained by local search is considered as a candidate to be inserted into the pool if it is sufficiently different from every other solution currently in the pool. If the pool already has `Max_Elite` solutions and the candidate is better than the worst of them, then a simple strategy is to have the former replaces the latter. Another strategy, which tends to increase the diversity of the pool, is to replace the pool element most similar to the candidate among all pool elements with cost worse than the candidate's. If the pool is not full, the candidate is simply inserted.

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

Algorithm 2.3 illustrates such a procedure. Each GRASP iteration has now three main steps:

- *Construction phase:* a greedy randomized construction procedure is used to build a feasible solution;

- *Local search phase:* the solution built in the first phase is progressively improved by a neighborhood search strategy, until a local minimum is found; and

- *Path-relinking phase:* the path-relinking algorithm using any of the strategies described in Section 2.2 is applied to the solution obtained by local search and to a randomly selected solution from the pool. The best solution found along this trajectory is also considered as a candidate for insertion in the pool and the incumbent is updated.

At the end of the GRASP iterations, a post-optimization phase combines the elite solutions in the pool in the search for better solutions..

Data : Number of iterations $i_{\max}$
Result: Solution $x^* \in X$
$P \leftarrow \emptyset$;
$f^* \leftarrow \infty$;
for $i = 1, \ldots, i_{\max}$ **do**
  $x \leftarrow$ `GreedyRandomizedConstruction`();
  $x \leftarrow$ `LocalSearch`(x);
  if $i \geq 2$ **then**
    Choose, at random, pool solutions $\mathcal{Y} \subseteq P$ to relink with x;
    for $y \in \mathcal{Y}$ **do**
      Determine which (x or y) is initial x_s and which is target x_t;
      $x_p \leftarrow$ `PathRelinking`(x_s, x_t);
      Update the elite set P with x_p;
      if $f(x_p) < f^*$ **then**
        $f^* \leftarrow f(x_p)$;
        $x^* \leftarrow x_p$;
      end
    end
  end
end
$P =$ `PostOptimize`$\{P\}$;
$x^* = \text{argmin}\{f(x), x \in P\}$;

Algorithm 2.3: A basic GRASP with path-relinking heuristic for minimization.

Aiex [1] and Aiex et al. [4] have shown experimentally that the solution times for finding a target solution value with a GRASP heuristic fit a two-parameter exponential distribution. Figure 2.3 illustrates this result, depicting the superimposed empirical and theoretical distributions observed for one of the cases studied in the computational experiments reported by the authors, which involved 2400 runs of GRASP

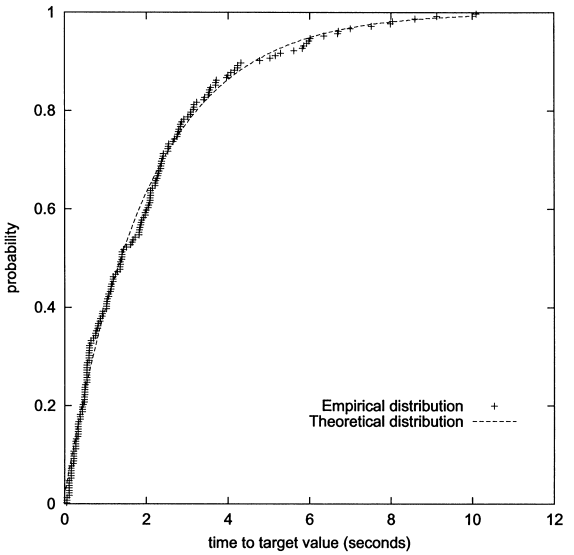

Figure 2.3. Superimposed empirical and theoretical distributions (times to target values measured in seconds on an SGI Challenge computer with 28 processors).

procedures for each of five different problems: maximum independent set [16, 37], quadratic assignment [29, 38], graph planarization [40, 47], maximum weighted satisfiability [39], and maximum covering [36]. The same result still holds when GRASP is implemented in conjunction with a path-relinking procedure [2].

2.4 APPLICATIONS

Path-relinking has been successfully used together with GRASP in a variety of applications, such as the three index assignment problem [1, 3], the problem of routing private circuits in communication networks [41], the 2-path network design problem [48], the p-median problem [44], the Steiner problem in graphs [50], the job-shop scheduling problem [1, 2], the prize-collecting Steiner tree problem [9], the quadratic assignment problem [32], the MAX-CUT problem [17], and the capacitated minimum spanning tree problem [53]. Some of these applications will be reviewed in the remainder of this section.

Before we review some of these applications, we first describe a plot used in several of our papers to experimentally compare different randomized algorithms or different versions of the same randomized algorithm [1, 4]. This plot shows empirical distributions of the random variable *time to target solution value*. To plot the empirical distribution, we fix a solution target value and run each algorithm T independent

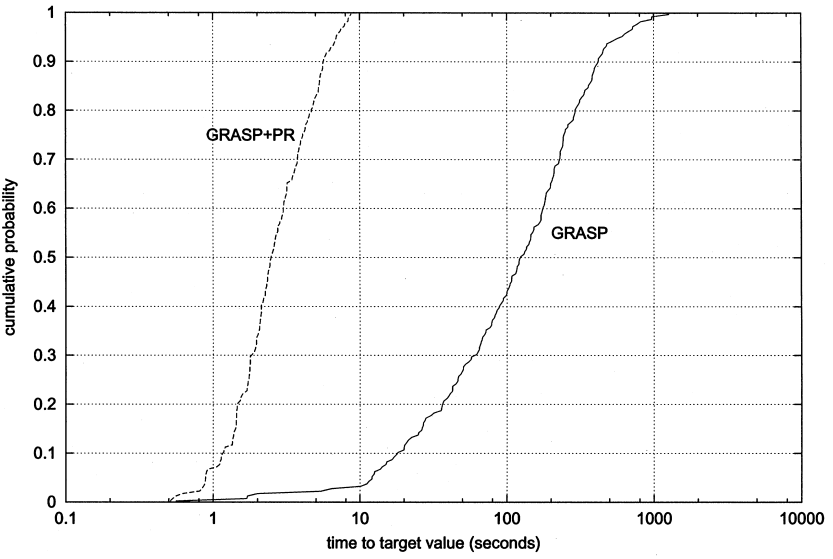

Figure 2.4. Empirical distributions of the random variables *time to target solution* for a pure GRASP and a GRASP with path-relinking for MAX-CUT instance G11 with target value of 552 [17]. Two hundred independent runs of each algorithm were used to make the plots.

times, recording the running time when a solution with cost at least as good as the target value is found. For each algorithm, we associate with the i-th sorted running time (t_i) a probability $p_i = (i - \frac{1}{2})/T$, and plot the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, T$. Figure 2.4 shows one such plot comparing a pure GRASP with a GRASP with path-relinking for MAX-CUT instance G11 with target solution value of 552. The figure shows clearly that GRASP with path-relinking (GRASP+PR) is much faster than pure GRASP to find a solution with weight 552 or more. For instance, the probability of finding such a solution in less than 5 seconds is over 80% with GRASP with path-relinking, while it is about 2% with pure GRASP. Similarly, with probability 50% GRASP with path-relinking finds such a target solution in less than 2.5 seconds, while for pure GRASP, with probability 50% a solution is found in less than 122 seconds.

2.4.1 Private virtual circuit routing

A frame relay service offers virtual private networks to customers by provisioning a set of long-term private virtual circuits (PVCs) between customer endpoints on a large backbone network. During the provisioning of a PVC, routing decisions are made without any knowledge of

future requests. Over time, these decisions can cause inefficiencies in the network and occasional offline rerouting of the PVCs is needed. Resende and Ribeiro [43] formulate the offline PVC routing problem as an integer multi-commodity flow problem with additional constraints and with an objective function that minimizes propagation delays and/or network congestion. They propose variants of a GRASP with path-relinking heuristic for this problem. Experimental results for realistic-size problems show that GRASP benefits greatly from path-relinking and that the proposed heuristics are able to improve the solutions found with standard routing techniques.

Let $G = (V, E)$ be an undirected graph representing the frame relay network. Denote by $V = \{1, \ldots, n\}$ the set of backbone nodes where switches reside, while E is set of trunks (or edges) that connect the backbone nodes, with $|E| = m$. Parallel trunks are allowed. Since G is an undirected graph, flows through each trunk $(i, j) \in E$ have two components to be summed up, one in each direction. However, for modeling purposes, costs and capacities will always be associated only with the ordered pair (i, j) satisfying $i < j$. For each trunk $(i, j) \in E$, denote by b_{ij} its maximum allowed bandwidth (in kbits/second), while c_{ij} denotes the maximum number of PVCs that can be routed through it and d_{ij} is the propagation, or hopping, delay associated with the trunk. Each commodity $k \in K = \{1, \ldots, p\}$ is a PVC to be routed, associated with an origin-destination pair and with a bandwidth requirement (or demand, also known as its effective bandwidth) r_k. The latter takes into account the actual bandwidth required by the customer in the forward and reverse directions, as well as an overbooking factor.

Let $x_{ij}^k = 1$ if and only if edge $(i, j) \in E$ is used to route commodity $k \in K$. The cost function $\phi_{ij}(x_{ij}^1, \ldots, x_{ij}^p, x_{ji}^1, \ldots, x_{ji}^p)$ associated with each trunk $(i, j) \in E$ with $i < j$ is the linear combination of a trunk propagation delay component and a trunk congestion component. The *propagation delay component* is defined as

$$\phi_{ij}^d(x_{ij}^1, \ldots, x_{ij}^p, x_{ji}^1, \ldots, x_{ji}^p) = d_{ij} \cdot \sum_{k \in K} \rho_k (x_{ij}^k + x_{ji}^k), \qquad (2.1)$$

where coefficients ρ_k are used to model two plausible delay functions:

- If $\rho_k = 1$, then this component leads to the minimization of the number of hops weighted by the propagation delay on each trunk.

- If $\rho_k = r_k$, then the minimization takes into account the effective bandwidth routed through each trunk weighted by its propagation delay.

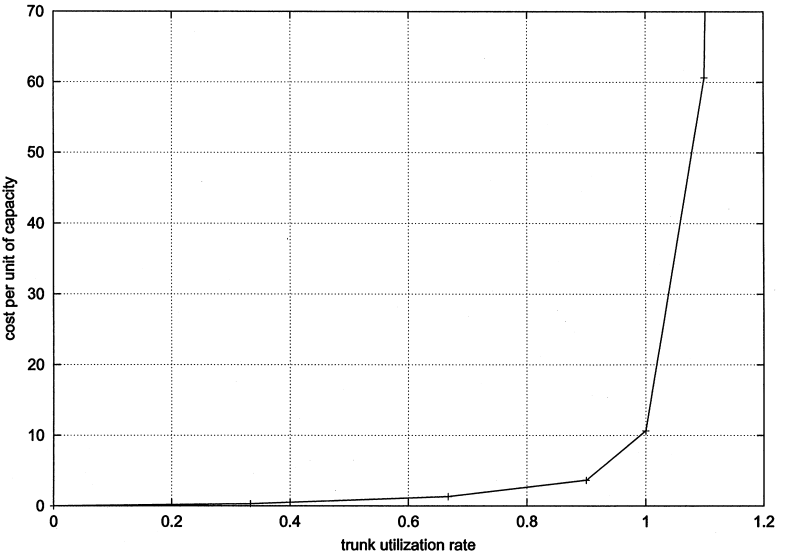

Figure 2.5. Piecewise linear congestion cost component associated with each trunk.

Let $y_{ij} = \sum_{k \in K} r_k(x_{ij}^k + x_{ji}^k)$ be the total flow through trunk $(i,j) \in E$ with $i < j$. The *trunk congestion component* depends on the utilization rates $u_{ij} = y_{ij}/b_{ij}$ of each trunk $(i,j) \in E$ with $i < j$. It is taken as the piecewise linear function proposed by Fortz and Thorup [20] and depicted in Figure 2.5, which increasingly penalizes flows approaching or violating the capacity limits:

$$\phi_{ij}^b(x_{ij}^1, \ldots, x_{ij}^p, x_{ji}^1, \ldots, x_{ji}^p) = \\ = b_{ij} \cdot \begin{cases} u_{ij}, & u_{ij} \in [0, 1/3) \\ 3 \cdot u_{ij} - 2/3, & u_{ij} \in [1/3, 2/3), \\ 10 \cdot u_{ij} - 16/3, & u_{ij} \in [2/3, 9/10), \\ 70 \cdot u_{ij} - 178/3, & u_{ij} \in [9/10, 1), \\ 500 \cdot u_{ij} - 1468/3, & u_{ij} \in [1, 11/10), \\ 5000 \cdot u_{ij} - 16318/3, & u_{ij} \in [11/10, \infty). \end{cases} \tag{2.2}$$

For PVC routing, Resende and Ribeiro used the cost function

$$\phi_{ij}(x_{ij}^1, \ldots, x_{ij}^p, x_{ji}^1, \ldots, x_{ji}^p) = \\ = (1-\delta) \cdot \phi_{ij}^d(x_{ij}^1, \ldots, x_{ij}^p, x_{ji}^1, \ldots, x_{ji}^p) + \delta \cdot \phi_{ij}^b(x_{ij}^1, \ldots, x_{ij}^p, x_{ji}^1, \ldots, x_{ji}^p) \tag{2.3}$$

associated with each trunk $(i,j) \in E$ with $i < j$, where weights $(1-\delta)$ and δ correspond respectively to the propagation delay and the network congestion components, with $\delta \in [0,1]$.

In the construction phase of GRASP, the routes are determined, one at a time. A new PVC is selected to be routed in each iteration. To reduce the computation times, we used a combination of the strategies usually employed by GRASP and heuristic-biased stochastic sampling. We create a restricted candidate list with a fixed number of elements n_c. At each iteration, it is formed by the n_c unrouted PVC pairs with the largest demands. An element ℓ is selected at random from this list with probability $\pi(\ell) = r_\ell / \sum_{k \in \mathrm{RCL}} r_k$.

Once a PVC $\ell \in K$ is selected, it is routed on a shortest path from its origin to its destination. The bandwidth capacity constraints are relaxed and handled via the penalty function introduced by the trunk congestion component (2.2) of the edge weights. The constraints on the limit of PVCs routed through each trunk are explicitly taken into account by forbidding routing through trunks already using its maximum number of PVCs. The weight $\Delta\phi_{ij}$ of each edge $(i,j) \in E$ is given by the increment of the cost function value $\phi_{ij}(x_{ij}^1, \ldots, x_{ij}^p, x_{ji}^1, \ldots, x_{ji}^p)$, associated with routing r_ℓ additional units of demand through edge (i,j).

More precisely, let $\underline{K} \subseteq K$ be the set of previously routed PVCs and $\underline{K}_{ij} \subseteq \underline{K}$ be the subset of PVCs that are routed through trunk $(i,j) \in E$. Likewise, let $\overline{K} = \underline{K} \cup \{\ell\} \subseteq K$ be the new set of routed PVCs and $\overline{K}_{ij} = \underline{K}_{ij} \cup \{\ell\} \subseteq \overline{K}$ be the new subset of PVCs that are routed through trunk (i,j). Then, define $\underline{x}_{ij}^h = 1$ if PVC $h \in \underline{K}$ is routed through trunk $(i,j) \in E$ from i to j, $\underline{x}_{ij}^h = 0$ otherwise. Similarly, define $\overline{x}_{ij}^h = 1$ if PVC $h \in \overline{K}$ is routed through trunk $(i,j) \in E$ from i to j, $\overline{x}_{ij}^h = 0$ otherwise. According to (2.3), the cost associated with each edge $(i,j) \in E$ in the current solution is given by $\phi_{ij}(\underline{x}_{ij}^1, \ldots, \underline{x}_{ij}^p, \underline{x}_{ji}^1, \ldots, \underline{x}_{ji}^p)$. In the same manner, the cost associated with each edge $(i,j) \in E$ after routing PVC ℓ will be $\phi_{ij}(\overline{x}_{ij}^1, \ldots, \overline{x}_{ij}^p, \overline{x}_{ji}^1, \ldots, \overline{x}_{ji}^p)$. Then, the incremental edge weight $\Delta\phi_{ij}$ associated with routing PVC $\ell \in K$ through edge $(i,j) \in E$, used in the shortest path computations, is given by

$$\Delta\phi_{ij} = \phi_{ij}(\overline{x}_{ij}^1, \ldots, \overline{x}_{ij}^p, \overline{x}_{ji}^1, \ldots, \overline{x}_{ji}^p) - \phi_{ij}(\underline{x}_{ij}^1, \ldots, \underline{x}_{ij}^p, \underline{x}_{ji}^1, \ldots, \underline{x}_{ji}^p). \tag{2.4}$$

The enforcement of the constraints that limit the number of PVCs routed through each trunk may lead to unroutable demand pairs. In this case, the current solution is discarded and a new construction phase starts.

Each solution built in the first phase may be viewed as a set of routes, one for each PVC. The local search procedure seeks to improve each

route in the current solution. For each PVC $k \in K$, start by removing r_k units of flow from each edge in its current route. Next, compute incremental edge weights $\Delta\phi_{ij}$ associated with routing this demand through each trunk $(i, j) \in E$ according to (2.4). A tentative new shortest path route is computed using the incremental edge weights. If the new route improves the solution, it replaces the current route of PVC k. This is continued until no improving route can be found.

In the proposed path-relinking strategy, 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 $K_{x_1,x_2} \subseteq K$ of PVCs routed through different routes in x_1 and x_2. Without loss of generality, suppose that path-relinking starts from any elite solution z in the pool and uses the locally optimal solution y as the guiding solution.

The best solution $\overline{y}$ along the new path to be constructed is initialized with z. For each PVC $k \in K_{y,z}$, the same shortest path computations described for construction and local search are used to evaluate the cost of the new solution obtained by rerouting the demand associated with PVC k through the route used in the guiding solution y instead of that used in the current solution originated from z. The best move is selected and removed from $K_{y,z}$. The new solution obtained by rerouting the above selected PVC is computed, the incumbent $\overline{y}$ is updated, and a new iteration resumes. These steps are repeated, until the guiding solution y is reached. The incumbent $\overline{y}$ is returned as the best solution found by path-relinking and inserted into the pool if it is better than the worst solution currently in the pool.

Figure 2.6 illustrates the comparison of the four algorithms: pure GRASP (`GRASP`), GRASP with forward path-relinking (`GRASP+PRf`, in which a locally optimal solution is used as the initial solution), GRASP with backward path-relinking (`GRASP+PRb`, in which an elite solution is used as the initial solution), and GRASP with backward and forward path-relinking (`GRASP+PRfb`, in which path-relinking is performed in both directions) on PVC routing instance `fr750a` (60 nodes, 498 arcs, and 750 commodities). For a given computation time, the probability of finding a solution at least as good as the target value increases from `GRASP` to `GRASP+PRf`, from `GRASP+PRf` to `GRASP+PRfb`, and from `GRASP+PRfb` to `GRASP+PRb`. For example, there is 9.25% probability for `GRASP+PRfb` to find a target solution in less than 100 seconds, while this probability increases to 28.75% for `GRASP+PRb`. For `GRASP`, there is a 8.33% probability of finding a target solution within 2000 seconds, while for `GRASP+PRf` this probability increases to 65.25%. `GRASP+PRb` finds a target solution in at most 129 seconds with 50% probability. For the

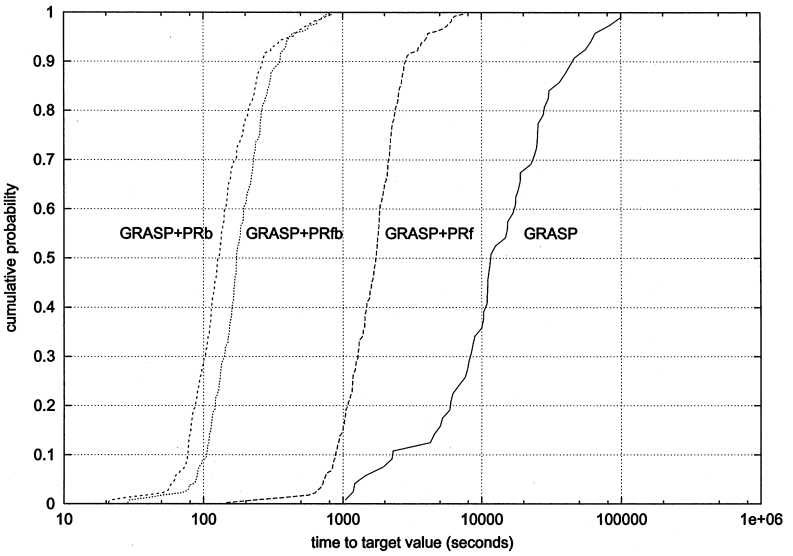

Figure 2.6. Empirical distributions of time to target solution for GRASP, GRASP with forward path-relinking, GRASP with backward path-relinking, and GRASP with back and forward path-relinking for private virtual circuit routing instance `fr750a`. Two hundred independent runs were done for each algorithm. Target solution value used was 479000.

same probability, this time increases to 172, 1727, and 10933 seconds, respectively, for variants `GRASP+PRfb`, `GRASP+PRf`, and `GRASP`.

These results suggest that variant `GRASP+PRb`, which performs path-relinking backward from an elite solution to a locally optimal solution, is the most effective.

Another experiment comparing the four variants was done on PVC routing instance `att` (90 nodes, 274 trunks, 272 commodities). Ten independent runs of each algorithm were done for 100 seconds on a 196 MHz SGI Challenge computer. Table 2.1 summarizes these results. For each variant, this table lists the best and average solution values found after 10 seconds and after 100 seconds. The results point to `GRASP+PRb` and `GRASP+PRfb` as the two best heuristics. It is interesting to note that even if given 100 seconds, `GRASP` finds solutions of worse quality than those found by `GRASP+PRb` and `GRASP+PRfb` in only 10 seconds.

2.4.2 2-path network design

Let $G = (V, E)$ be a connected undirected graph, where V is the set of nodes and E is the set of edges. A k-path between nodes $s, t \in V$ is a sequence of at most k edges connecting them. Given a non-negative

Table 2.1. Comparison of GRASP, GRASP with forward path-relinking, GRASP with backward path-relinking, and GRASP with back and forward path-relinking for private virtual circuit routing instance `att`. Ten independent runs of 100 seconds were done for each algorithm.

10 runs	10 seconds		100 seconds	
Variant	best	average	best	average
`GRASP`	126603	126695	126228	126558
`GRASP+PRf`	126301	126578	126083	126229
`GRASP+PRb`	125960	126281	125666	125883
`GRASP+PRfb`	125961	126307	125646	125850

weight function $w : E \rightarrow R_+$ associated with the edges of G and a set D of pairs of origin-destination nodes, the *2-path network design problem* (2PNDP) consists in finding a minimum weighted subset of edges $E' \subseteq E$ containing a 2-path between every origin-destination pair. Applications can be found in the design of communication networks, in which paths with few edges are sought to enforce high reliability and small delays. Dahl and Johannessen [13] proved that the decision version of 2PNDP is NP-complete.

Rosseti [52] and Ribeiro and Rosseti [48] described sequential and parallel implementations of GRASP with path relinking for the 2-path network design. The construction of a new solution begins by the initialization of modified edge weights with the original edge weights. Each iteration of the construction phase starts by the random selection of an origin-destination pair still in D. A shortest 2-path between the extremities of this pair is computed, using the modified edge weights. The weights of the edges in this 2-path are set to zero until the end of the construction procedure, the origin-destination pair is removed from D, and a new iteration resumes. The construction phase stops when 2-paths have been computed for all origin-destination pairs.

The local search phase seeks to improve each solution built in the construction phase. Each solution may be viewed as a set of 2-paths, one for each origin-destination pair in D. To introduce some diversity by driving different applications of the local search to different local optima, the origin-destination pairs are investigated at each GRASP iteration in a circular order defined by a different random permutation of their original indices. Each 2-path in the current solution is tentatively eliminated. The weights of the edges used by other 2-paths are temporarily set to zero, while those which are not used by other 2-paths in the current solution are restored to their original values. A new shortest 2-path be-

tween the extremities of the origin-destination pair under investigation is computed, using the modified weights. If the new 2-path improves the current solution, then the latter is modified; otherwise the previous 2-path is restored. The search stops if the current solution was not improved after a sequence of $|D|$ iterations along which all 2-paths have been investigated. Otherwise, the next 2-path in the current solution is investigated for substitution and a new iteration resumes.

Each GRASP iteration performs an intensification phase using path-relinking, in which the newly generated solution obtained at the end of the local search phase is combined with a randomly selected solution from the pool of elite solutions. Path-relinking starts by determining all origin-destination pairs whose associated 2-paths are different in the two solutions. These computations amount to determining the set of moves which should be applied to the initial solution to reach the guiding one. Each move is characterized by a pair of 2-paths, one to be inserted and the other to be eliminated from the current solution. At each path-relinking iteration the best yet unselected move is applied to the current solution and the best solution found along the path connecting the two solutions is updated. The incumbent best solution found along the path-relinking step is inserted into the pool if it is better than the worst solution currently in the pool.

Several strategies for the implementation of the path-relinking step have been investigated in [49, 52]: pure GRASP (`GRASP`), GRASP with forward path-relinking (`GRASP+PRf`, in which a locally optimal solution is used as the initial solution), GRASP with backward path-relinking (`GRASP+PRb`, in which an elite solution is used as the initial solution), GRASP with backward and forward path-relinking (`GRASP+PRfb`, in which path-relinking is performed twice, once in each direction), and GRASP with mixed path-relinking (`GRASP+ PRm`, in which two paths in opposite directions are simultaneously explored).

The results displayed in Table 2.2 illustrate the behavior of these five variants on randomly generated instances [52] on complete graphs with 100, 200, 300, 400, and 500 nodes. For each instance, we give the best and average solution values found over ten independent runs of each algorithm. For each problem size, the processing time is limited at that observed for 200 iterations of the pure GRASP procedure on the first instance in the group. Algorithms `GRASP+PRfb` and `GRASP+PRm` performed better than the other variants, as far as together they found the best solutions and the best average solutions for all instances in the table. GRASP with backward path-relinking usually performs better than the forward path-relinking variant, due to the fact that it starts

Table 2.2. Results for ten runs of each algorithm on randomly generated instances of 2-path network design problems with limited processing time.

	GRASP		GRASP+PRf		GRASP+PRb		GRASP+PRfb		GRASP+PRm	
$\|V\|$	best	avg.	best	avg.	best	avg.	best	avg.	best	avg.
100	779	784.3	760	772.8	763	769.3	749	762.7	755	765.3
	762	769.6	730	749.4	735	746.0	729	741.7	736	745.7
	773	779.2	762	769.3	756	766.1	757	763.6	754	765.1
	746	752.0	732	738.4	723	736.7	719	730.4	717	732.2
	756	762.3	742	749.7	739	746.5	737	742.9	728	743.7
200	1606	1614.7	1571	1584.4	1540	1568.0	1526	1562.0	1538	1564.3
	1601	1608.8	1557	1572.8	1559	1567.9	1537	1558.9	1545	1563.3
	1564	1578.2	1523	1541.9	1516	1531.9	1508	1519.9	1509	1528.7
	1578	1585.6	1531	1553.3	1518	1538.1	1510	1532.2	1513	1534.7
	1577	1599.6	1567	1575.4	1543	1563.5	1529	1556.3	1531	1556.1
300	2459	2481.9	2408	2425.0	2377	2401.3	2355	2399.2	2366	2393.6
	2520	2527.7	2453	2469.7	2419	2449.1	2413	2438.9	2405	2439.4
	2448	2463.5	2381	2403.1	2339	2373.8	2356	2375.3	2338	2370.3
	2462	2482.1	2413	2436.2	2373	2409.3	2369	2400.9	2350	2401.0
	2450	2458.8	2364	2402.5	2328	2368.6	2347	2373.9	2322	2365.4
400	3355	3363.8	3267	3285.5	3238	3257.0	3221	3239.4	3231	3252.2
	3393	3417.5	3324	3338.2	3283	3306.8	3220	3292.2	3271	3301.4
	3388	3394.4	3311	3322.4	3268	3291.9	3227	3275.1	3257	3273.2
	3396	3406.0	3316	3326.5	3249	3292.0	3256	3284.8	3246	3287.9
	3416	3429.3	3335	3365.5	3267	3327.7	3270	3313.9	3259	3323.5
500	4338	4350.1	4209	4247.1	4176	4207.6	4152	4196.1	4175	4206.2
	4353	4369.6	4261	4278.6	4180	4233.7	4166	4219.6	4175	4226.3
	4347	4360.7	4239	4257.8	4187	4224.8	4170	4201.9	4187	4217.9
	4317	4333.8	4222	4238.6	4157	4197.4	4156	4182.2	4159	4197.1
	4362	4370.4	4263	4292.0	4203	4294.0	4211	4236.8	4200	4240.2

from an elite solution that is often better than the current local optimum, fully exploring the neighborhood of the former.

The results observed for variant `GRASP+PRm` are very encouraging: this algorithm found better solutions than the other variants for 40% of the instances.

To further illustrate and compare these five variants, we display in Figure 2.7 a plot of the empirical probability distribution of the time to target solution value for each algorithm, computed from 200 independent runs. These plots show that the probability of finding a solution at least as good as a target value increases from `GRASP` to `GRASP+PRf` to `GRASP+PRb` to `GRASP+PRfb`, and finally to `GRASP+PRm`. These results confirms an observation first noticed by Ribeiro et al. [50] and later by Resende and Ribeiro [42], suggesting that the backward strategy performs

a major role in successful implementations of path-relinking. Moreover, they also indicate that the mixed path-relinking strategy proposed by Rosseti [52] is very effective.

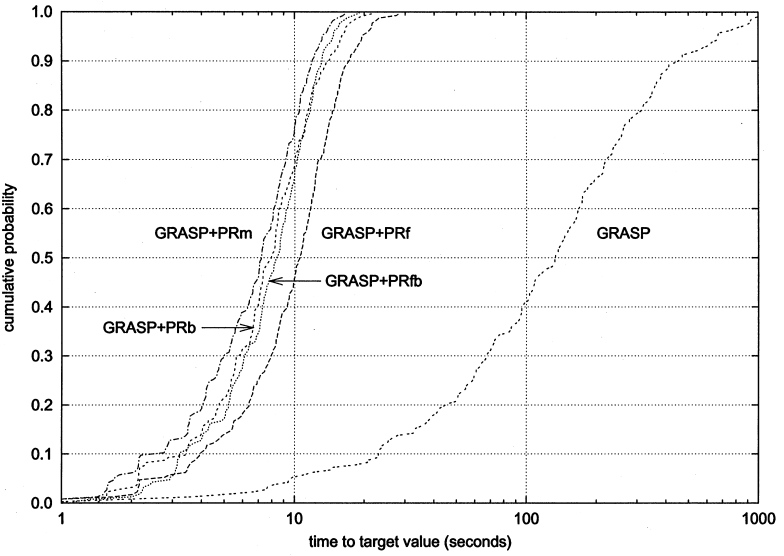

Figure 2.7. Empirical distributions of time to target solution for GRASP, GRASP with forward path-relinking, GRASP with backward path-relinking, GRASP with back and forward path-relinking, and GRASP with mixed path-relinking for a 2-path network design instance with 80 nodes. Two hundred independent runs were done for each algorithm. Target solution value used was 588.

2.4.3 p-median problem

In the p-median problem, we are given a set F of m potential facilities, a set U of n users (or customers), a distance function $d : U \times F \rightarrow \mathbb{R}$, and a constant $p \leq m$, and want to determine which p facilities to open so as to minimize the sum of the distances from each user to its closest open facility. It is a well-known NP-hard problem [27].

Resende and Werneck [45] describe a GRASP with path-relinking for the p-median problem. Empirical results on instances from the literature show that the algorithm is robust and that it performs at least as well as other methods, and often better in terms of both running time and solution quality. In all cases the solutions obtained by the GRASP with path-relinking were within 0.1% of the best known upper bounds. For a large number of instances new best known solutions were produced by the new algorithm.

The standard greedy algorithm for the p-median problem [10, 55] starts with an empty solution and adds facilities one at a time, choosing the most profitable in each iteration (the one whose insertion causes the greatest drop in solution cost). The construction procedure proposed in [45] is similar to the greedy algorithm, but instead of selecting the best among all possible options, it only considers $q < m$ possible insertions (chosen uniformly at random) in each iteration. The most profitable among those is selected. The running time of the algorithm is $O(m + pqn)$. The idea is to make q small enough so as to significantly reduce the running time of the algorithm (when compared to the pure greedy one) and to ensure a fair degree of randomization. In tests, the value $q = \lceil \log_2(m/p) \rceil$ was determined to be suitable.

The standard local search procedure for the p-median problem, originally proposed by Teitz and Bart [54], is based on swapping facilities. Given an initial solution S, the procedure determines, for each facility $f \notin S$, which facility $g \in S$ (if any) would improve the solution the most if f and g were interchanged (i.e., if f were opened and g closed). If there is one such improving move, f and g are interchanged. The procedure continues until no improving interchange can be made, in which case a local minimum will have been found. The complexity of this swap-based local search is $O(pmn)$ per iteration. Whitaker [55] proposed an efficient implementation of this method, which he called *fast interchange*, for which the bound on the running time of each iteration is reduced to $O(mn)$. Resende and Werneck [35] have recently proposed an alternative implementation. Although it has the same worst-case complexity as Whitaker's, it can be substantially faster in practice. The speedup (of up to three orders of magnitude) results from the use of information gathered in early iterations of the algorithm to reduce the amount of computation performed in later stages. Though this implementation can require a greater amount of memory, with the use of some programming techniques (e.g. sparse matrix representation and cache), the additional memory requirements can be minimized.

Intensification (via path-relinking) occurs in two different stages. First, every GRASP iteration contains an intensification step, in which the newly generated solution is combined with a solution from the pool. Then, in the post-optimization phase, solutions in the pool are combined among themselves.

Let S_1 and S_2 be two valid solutions, interpreted as sets of (open) facilities. The path-relinking procedure starts with one of the solutions (say, S_1) and gradually transforms it into the other (S_2) by swapping in elements from $S_2 \setminus S_1$ and swapping out elements from $S_1 \setminus S_2$. The total number of swaps made is $|S_2 \setminus S_1|$, which is equal to $|S_1 \setminus S_2|$. The choice

of which swap to make in each stage is greedy: the most profitable (or least costly) move is made.

The outcome of path-relinking is the best local minimum in the path. A local minimum in this context is a solution that is both succeeded (immediately) and preceded (either immediately or through a series of same-value solutions) in the path by strictly worse solutions. If the path has no local minima, one of the original solutions (S_1 or S_2) is returned with equal probability. When there is an improving solution in the path, this criterion matches the traditional one exactly: it simply returns the best element in the path. It is different only when the standard path-relinking is unsuccessful, in which case it tries to increase diversity by selecting a solution other than the extremes of the path.

Note that path-relinking is very similar to the local search procedure described earlier, with two main differences. First, the number of allowed moves is restricted: only elements in $S_2 \setminus S_1$ can be inserted, and only those in $S_1 \setminus S_2$ can be removed. Second, non-improving moves are allowed. These differences are easily incorporated into the basic implementation of the local search procedure.

The intensification procedure is augmented by performing a full local search on the solution produced by path-relinking. Because this solution is usually very close to a local optimum, this application tends to be much faster than on a solution generated by the randomized constructive algorithm. A side effect of applying local search at this point is increased diversity, since one is free to use facilities that did not belong to any of the original solutions.

The plots in Figure 2.8 compare GRASP with path-relinking and pure GRASP on the 1400-facility, 1400-user TSPLIB instance fl1400. The plot on the left shows quality of the best solution found as a fraction of the average value of the first solution for GRASP with path-relinking and pure GRASP for $p = 500$. Times are given as multiples of the average time required to perform one multi-start iteration. Smaller values are better. The plot on the right shows ratios between partial solutions found with and without path-relinking for different values of p. Ratios smaller than 1.000 favor the use of path-relinking. The plots show that GRASP benefits from path-relinking, in particular for large values of p.

2.4.4 Three index assignment problem

The three-index assignment problem (AP3) was introduced by Pierskalla [33] as an extension of the classical two-dimensional assignment problem. Consider a complete tripartite graph $K_{n,n,n} = (I \cup J \cup K, (I \times J) \cup (I \times K) \cup (J \times K))$,where I, J, and K are disjoint sets of size n.

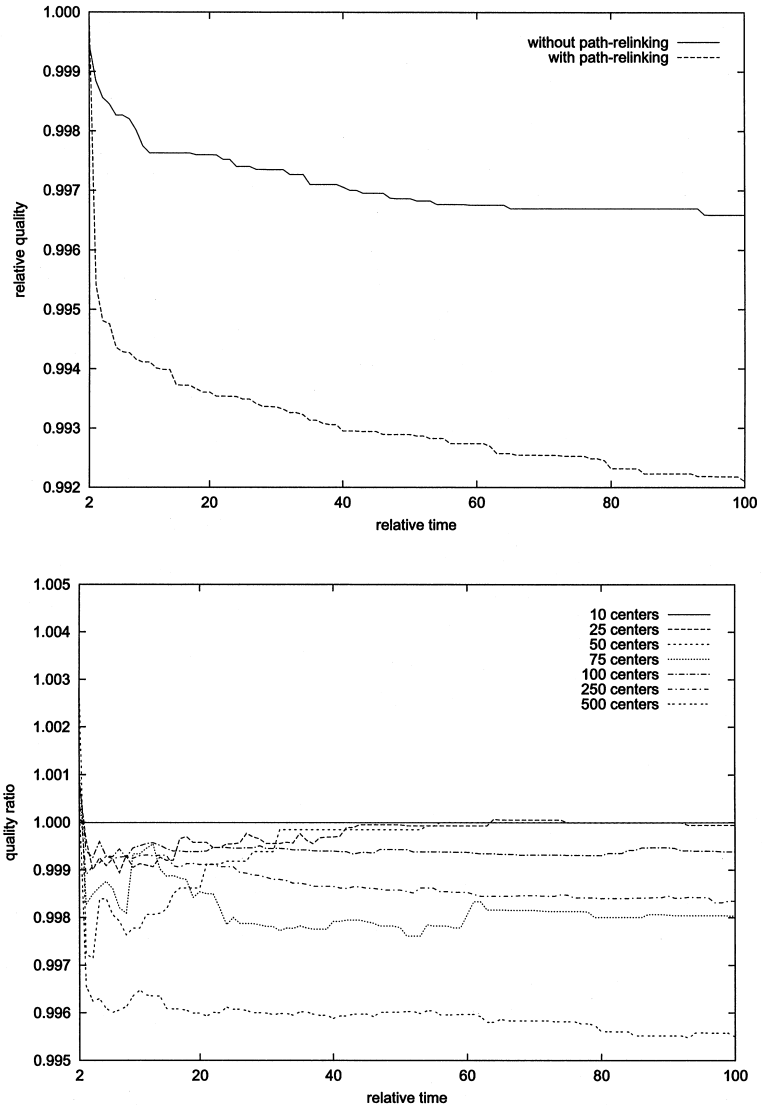

Figure 2.8. GRASP with path-relinking versus pure GRASP on TSPLIB instance fl1400.

If a cost $c_{i,j,k}$ is associated with each triplet $(i, j, k) \in I \times J \times K$, then the AP3 consists of finding a subset $A \in I \times J \times K$ of n triplets such that every element of $I \cup J \cup K$ occurs in exactly one triplet of A, and the sum of the costs of the chosen triplets is minimized. The AP3 is

NP-hard [21, 22]. A permutation-based formulation for AP3 is

$$\min_{p,q\in\pi_N} \sum_{i=1}^{n} c_{ip(i)q(i)},$$

where π_N denotes the set of all permutations of the set of integers $N = \{1, 2, \ldots, n\}$.

Aiex [1] and Aiex et al. [3] describe a GRASP with path-relinking for AP3. Computational results show clearly that this GRASP for AP3 benefits from path-relinking and compares well with previously proposed heuristics for this problem. GRASP with path-relinking was able to improve the solution quality of heuristics proposed by Balas and Saltzman [6], Burkard et al. [8], and Crama and Spieksma [11] on all instances proposed in those papers.

The GRASP construction phase builds a feasible solution S by selecting n triplets, one at a time. The solution S is initially empty and the set C of candidate triplets is initially the set of all triplets. To select the pth triplet ($p = 1, \ldots, n-1$) to be added to the solution, a restricted candidate list C' is defined to include all triplets (i, j, k) in the candidate set C having cost $c_{ijk} \leq \underline{c} + \alpha(\overline{c} - \underline{c})$, where

$$\underline{c} = \min\{c_{ijk} \mid (i, j, k) \in C\} \text{ and } \overline{c} = \max\{c_{ijk} \mid (i, j, k) \in C\}.$$

Triplet $(i_p, j_p, k_p) \in C'$ is chosen at random and is added to the solution, i.e. $S = S \cup \{(i_p, j_p, k_p)\}$. Once (i_p, j_p, k_p) is selected, the set of candidate triplets must be adjusted to take into account that (i_p, j_p, k_p) is part of the solution. Any triplet (i, j, k) such that $i = i_p$ or $j = j_p$ or $k = k_p$ is removed from the current set of candidate triplets. After $n - 1$ triplets have been selected, the set C of candidate triplets contains one last triplet which is added to S, thus completing the construction phase.

In the local search procedure, the current solution is improved by searching its neighborhood for a better solution. The solution of the AP3 can be represented by a pair of permutations (p, q). For a solution $p, q \in \pi_N$, the 2-exchange neighborhood is $N_2(p, q) = \{p', q' \mid d(p, p') + d(q, q') = 2\}$, where $d(s, s') = |\{i \mid s(i) \neq s'(i)\}|$.

In the local search, each cost of a neighborhood solution is compared with the cost of the current solution. If the cost of the neighbor is lower, then the solution is updated, the search is halted, and a search in the new neighborhood is initialized. The local search ends when no neighbor of the current solution has a lower cost than the current solution.

Path-relinking is done between an initial solution

$$S = \{(1, j_1^S, k_1^S), (2, j_2^S, k_2^S), \ldots, (n, j_n^S, k_n^S)\}$$

and a guiding solution

$$T = \{(1, j_1^T, k_1^T), (2, j_2^T, k_2^T), \ldots, (n, j_n^T, k_n^T)\}.$$

Let the symmetric difference between S and T be defined by the following two sets of indices:

$$\delta J = \{i = 1, \ldots, n \mid j_i^S \neq j_i^T\}$$

and

$$\delta K = \{i = 1, \ldots, n \mid k_i^S \neq k_i^T\}.$$

An intermediate solution of the path is visited at each step of path-relinking. Two elementary types of moves can be carried out. In a type-one move, triplets

$$\{(i_1, j_1, k_1), (i_2, j_2, k_2)\}$$

are replaced by triplets

$$\{(i_1, j_2, k_1), (i_2, j_1, k_2)\}.$$

In a type-two move, triplets

$$\{(i_1, j_1, k_1), (i_2, j_2, k_2)\}$$

are replaced by

$$\{(i_1, j_1, k_2), (i_2, j_2, k_1)\}.$$

Set δJ guides type-one moves, while δK guides type-two moves. For all $i \in \delta J$, let q be such that $j_q^T = j_i^S$. A type-one move replaces triplets

$$\{(i, j_i^S, k_i^S), (q, j_q^S, k_q^S)\}$$

by

$$\{(i, j_q^S, k_i^S), (q, j_i^S, k_q^S)\}.$$

For all $i \in \delta K$, let q be such that $k_q^T = k_i^S$. A type-two move replaces triplets

$$\{(i, j_i^S, k_i^S), (q, j_q^S, k_q^S)\}$$

by

$$\{(i, j_i^S, k_q^S), (q, j_q^S, k_i^S)\}.$$

At each step, the move that produces the least costly solution is selected and the corresponding index is deleted from either δJ or δK. This process continues until there are only two move indices left in one of the sets δJ or δK. At this stage, any of these two moves results in the

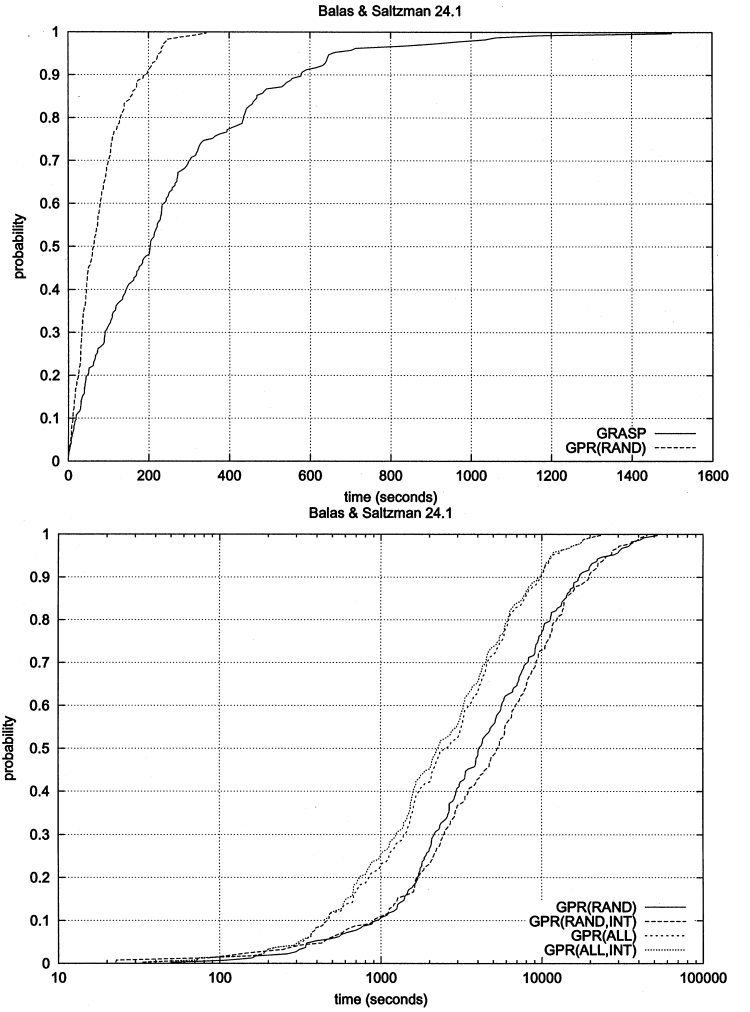

Figure 2.9. The plot of the left shows empirical probability distributions of time to target value for GRASP and GRASP with path-relinking (random selection of one guiding solution from elite set). The plot of the right shows empirical probability distributions of time to target value for different variants of GRASP with path-relinking.

guiding solution and, therefore, are not carried out. The best solution found in the path is returned by the procedure.

The plots in Figure 2.9 illustrate how GRASP with path-relinking compares with pure GRASP and how different variants of GRASP with path-relinking compare. The variants of GRASP with path-relinking tested were: random selection of one guiding solution [GPR(RAND)]; random selection of one guiding solution and periodic relinking of all elements in pool [GPR(RAND,INT)]; selection of all pool elements as guid-

ing solutions [GPR(ALL)]; and selection of all pool elements as guiding solutions with periodic relinking of all elements in pool [GPR(ALL,INT)]. The algorithms were run 200 times (using different initial seeds for the random number generator) on instance 24.1 of Balas and Saltzman [6], stopping when a solution value better than a given target value was found. The experiment comparing pure GRASP with GRASP with path-relinking used a target value of 17, while the one comparing the different variants of GRASP with path-relinking used a more difficult target value of 7. The plot on the left shows the benefit of using path-relinking in GRASP. The plot on the right shows that the variants using path-relinking with all elite solutions have a higher probability of finding a target solution in a given amount of time than the variants that use path-relinking with a single randomly selected elite solution. The use of periodic intensification does not appear to influence the distributions as much.

2.5 CONCLUSIONS AND EXTENSIONS

This paper reviewed recent advances and applications of GRASP with path-relinking. By providing a short discussion of each component of GRASP with path-relinking and showing examples of how such heuristics can be implemented for combinatorial optimization problems such as PVC routing, 2-path network design, 3-index assignment, and p-median, we hope this paper will serve as a guide for the reader to put together other GRASP with path-relinking heuristics.

Path-relinking is a major enhancement to the basic greedy randomized adaptive search procedure (GRASP), leading to significant improvements in both solution time and quality. It adds an effective memory mechanism to GRASP by providing an intensification strategy that explores trajectories connecting GRASP solutions and the best elite solutions previously produced during the search. The numerical results summarized for the four problems listed above clearly illustrate the benefits obtained by the combination of GRASP with path relinking.

In *evolutionary path-relinking* used in the post-optimization intensification phase, a new generation of elite solutions is generated from the current population in the pool of elite solutions by applying path-relinking between all pairs of solutions in this population. Solutions obtained by each path-relinking operation are tested for inclusion in the population of the next generation following the usual rules used in pool management. This strategy was successfully used for the Steiner problem in graphs by Ribeiro et al. [50], for the p-median problem by Resende

and Werneck [44], and for the uncapacitated facility location problem by Resende and Werneck [46].

Path-relinking may also be used as a solution extractor for population methods. In particular, path-relinking was recently successfully applied as a generalized crossover strategy to generate optimized offsprings in the context of a genetic algorithm for the phylogeny problem [51].

The fact that the computation time to find a target solution value using GRASP with path-relinking fits a two-parameter exponential distribution (cf. Section 2.3, see [1, 2, 4]) has a major consequence in parallel implementations of GRASP with path-relinking: linear speedups proportional to the number of processors can be easily observed in *parallel independent strategies*. Additionally, path-relinking offers a very effective mechanism for the implementation of *parallel cooperative strategies* [12]. In this case, inter-processor cooperation is enforced by a master processor which stores and handles a common pool of elite solutions which is shared by all slave processors performing GRASP with path-relinking. Careful implementations making appropriate use of the computer resources may lead to even larger speedups and to very robust parallel algorithms, see e.g. [31, 48, 49, 52]. Results obtained for the 2-path network design problem are illustrated in Figure 2.10, showing the speedup obtained by the cooperative strategy with respect to the independent one on a cluster of eight processors. Much larger improvements can be obtained with more processors.

Finally, we notice that path-relinking can also be successfully used in conjunction with implementations of other metaheuristics such as VNS and ant colonies, as recently reported e.g. in [5, 17].

ACKNOWLEDGMENTS

Most of this work is part of their dissertations and was jointly done with the following currently and former M.Sc. and Ph.D. students from the Catholic University of Rio de Janeiro, Brazil, which are all gratefully acknowledged: R.M. Aiex, S.A. Canuto, S.L. Martins, M. Prais, I. Rosseti, M.C. Souza, E. Uchoa, D.S. Vianna, and R.F. Werneck.

REFERENCES

[1] R.M. Aiex. *Uma investigação experimental da distribuição de probabilidade de tempo de solução em heurísticas GRASP e sua aplicação na análise de implementações paralelas.* PhD thesis, Department

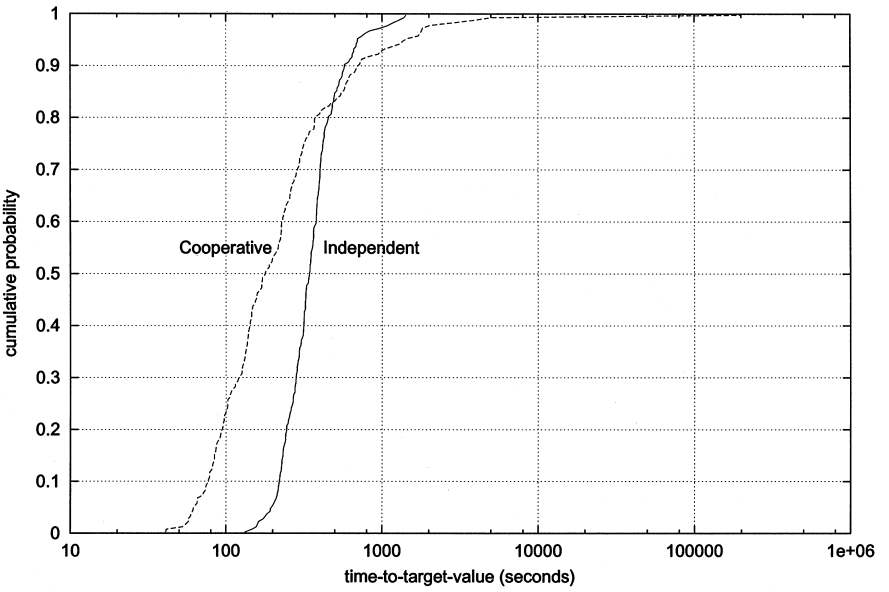

Figure 2.10. Probability distributions of time-to-target-value on an instance of the 2-path network design problem for cooperative and independent parallel implementations of GRASP with path-relinking on a Linux cluster with eight processors.

of Computer Science, Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, 2002.

[2] 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.

[3] R.M. Aiex, M.G.C. Resende, P.M. Pardalos, and G. Toraldo. GRASP with path relinking for the three-index assignment problem. Technical report, AT&T Labs Research, Florham Park, NJ 07733, 2000. To appear in *INFORMS J. on Computing.*

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

[5] D.J. Aloise, D. Aloise, C.T.M. Rocha, C.C. Ribeiro, José C. Ribeiro Filho, and Luiz S.S. Moura. Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics*, to appear.

[6] E. Balas and M.J. Saltzman. An algorithm for the three-index assignment problem. *Oper. Res.*, 39:150–161, 1991.

[7] 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.

[8] R.E. Burkard, R. Rudolf, and G.J. Woeginger. Three-dimensional axial assignment problems with decomposible cost coefficients. *Discrete Applied Mathematics*, 65:123–139, 1996.

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

[10] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: An analytical study of exact and approximate algorithms. *Management Science*, 23:789–810, 1977.

[11] Y. Crama and F.C.R. Spieksma. Approximation algorithms for three-dimensional assignment problems with triangle inequalities. *European Journal of Operational Research*, 60:273–279, 1992.

[12] V.-D. Cung, S.L. Martins, C.C. Ribeiro, and C. Roucairol. Strategies for the parallel implementation of metaheuristics. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publishers, 2002.

[13] G. Dahl and B. Johannessen. The 2-path network problem. *Networks*, 43:190–199, 2004.

[14] 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.

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

[16] 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.

[17] 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.

[18] P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and surveys on metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

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

[20] B. Fortz and M. Thorup. Internet traffic engineering by optimizing ospf weights. In *Proc. IEEE INFOCOM 2000 – The Conference on Computer Communications*, pages 519–528, 2000.

[21] A.M. Frieze. Complexity of a 3-dimensional assignment problem. *European Journal of Operational Research*, 13:161–164, 1983.

[22] M.R. Garey and D.S. Johnson. *Computers and intractability - A guide to the theory of NP-completeness.* W.H. Freeman and Company, 1979.

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

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

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

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

[27] O. Kariv and L. Hakimi. An algorithmic approach to nework location problems, Part II: The p-medians. *SIAM Journal of Applied Mathematics*, 37:539–560, 1979.

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

[29] 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.

[30] 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. Rajasekaran, 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.

[31] S.L. Martins, C.C. Ribeiro, and I. Rosseti. Applications and parallel implementations of metaheuristics in network design and routing. *Lecture Notes in Computer Science*, 3285:205–213, 2004.

[32] C.A. Oliveira, P.M. Pardalos, and M.G.C. Resende. GRASP with path-relinking for the QAP. In Toshihide Ibaraki and Yasunari Yoshitomi, editors, *Proceedings of the Fifth Metaheuristics International Conference*, pages 57–1 – 57–6, 2003.

[33] W.P. Pierskalla. The tri-subsitution method for the three-multidimensional assignment problem. *CORS J.*, 5:71–81, 1967.

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

[35] M. G. C. Resende and R. F. Werneck. On the implementation of a swap-based local search procedure for the p-median problem. In R. E. Ladner, editor, *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments (ALENEX'03)*, pages 119–127. SIAM, 2003.

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

[37] 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 Transactions on Mathematical Software*, 24:386–394, 1998.

[38] 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.

[39] 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.

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

[41] M.G.C. Resende and C.C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41:104–114, 2003.

[42] M.G.C. Resende and C.C. Ribeiro. GRASP and path-relinking: Recent advances and applications. In Toshihide Ibaraki and Yasunari Yoshitomi, editors, *Proceedings of the Fifth Metaheuristics International Conference*, pages T6–1 – T6–6, 2003.

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

[44] M.G.C. Resende and R.F. Werneck. A GRASP with path-relinking for the p-median problem. Technical Report TD-5E53XL, AT&T Labs Research, 2002.

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

[46] M.G.C. Resende and R.F. Werneck. A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, to appear.

[47] 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*, 25:341–352, 1999.

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

[49] C.C. Ribeiro and I. Rosseti. Efficient parallel cooperative implementations of GRASP heuristics. Technical report, Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, 2005.

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

[51] C.C. Ribeiro and D.S. Vianna. A genetic algorithm for the phylogeny problem using an optimized crossover strategy based on path-relinking. *Revista Tecnologia da Informação*, 3(2):67–70, 2003.

[52] I. Rosseti. *Heurísticas para o problema de síntese de redes a 2-caminhos*. PhD thesis, Department of Computer Science, Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, July 2003.

[53] M.C. Souza, C. Duhamel, and C.C. Ribeiro. A GRASP with path-relinking heuristic for the capacitated minimum spanning tree problem. In M.G.C. Resende and J. Souza, editors, *Metaheuristics: Computer Decision Making*, pages 627–658. Kluwer Academic Publishers, 2003.

[54] M. B. Teitz and P. Bart. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, 16:955–961, 1968.

[55] R. Whitaker. A fast algorithm for the greedy interchange of large-scale clustering and median location prolems. *INFOR*, 21:95–108, 1983.