Scatter search and path-relinking: Fundamentals, advances, and applications

Mauricio G.C. Resende, Celso C. Ribeiro, Fred Glover, and Rafael Martí

Abstract Scatter search is an evolutionary metaheuristic that explores solution spaces by evolving a set of reference points, operating on a small set of solutions while making only limited use of randomization. We give a comprehensive description of the elements and methods that make up its template, including the most recent elements incorporated in successful applications in both global and combinatorial optimization. Path-relinking is an intensification strategy to explore trajectories connecting elite solutions obtained by heuristic methods such as scatter search, tabu search, and GRASP. We describe its mechanics, implementation issues, randomization, the use of pools of high-quality solutions to hybridize path-relinking with other heuristic methods, and evolutionary path-relinking. We also describe the hybridization of path-relinking with genetic algorithms to implement a progressive crossover operator. Some successful applications of scatter search and of path-relinking are also reported.

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

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

Fred Glover
University of Colorado and OptTek Systems, Inc., Boulder, CO 80302 USA, e-mail: glover@colorado.edu

Rafael Martí
Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain, e-mail: rafael.marti@uv.es

1 Introduction

Scatter search (SS) is a metaheuristic that explores solution spaces by evolving a set of reference points. It can be viewed as an evolutionary method that operates on a small set of solutions and makes only limited use of randomization as a proxy for diversification when searching for a globally optimal solution. The scatter search framework is flexible, allowing the development of alternative implementations with varying degrees of sophistication.

The fundamental concepts and principles of the method were first proposed in the 1970s [5], based on formulations dating back to the 1960s for combining decision rules and problem constraints. In contrast to other evolutionary methods like genetic algorithms, scatter search is founded on the premise that systematic designs and methods for creating new solutions afford significant benefits beyond those derived from recourse to randomization. It uses strategies for search diversification and intensification that have proved effective in a variety of settings.

Scatter search orients its explorations systematically relative to a set of reference points that typically consist of good solutions obtained by prior problem solving efforts. The criteria for "good" are not restricted to objective function values, and may apply to sub-collections of solutions rather than to a single solution, as in the case of solutions that differ from each other according to certain specifications.

The scatter search template [7] has served as the main reference for most of the scatter search implementations to date. The dispersion patterns created by these designs have been found useful in several application areas. Section 2 gives a comprehensive description of the elements and methods of this template, based on the formulation given in Laguna and Martí [13]. It includes the most recent elements incorporated in successful applications in both global and combinatorial optimization.

Path-relinking is an intensification strategy to explore trajectories connecting elite solutions obtained by heuristic methods [6]. Path-relinking can be considered an extension of the *combination method* of scatter search. Instead of directly producing a new solution when combining two or more original solutions, path-relinking generates paths between and beyond the selected solutions in the neighborhood space. It should be noted that the combination method in scatter search is a problem-dependent element, which is customized depending on the problem and the solution representation. In particular, in global optimization, where solutions are represented as real vectors, most scatter search applications perform linear combinations between pairs of solutions. Alternatively, in problems where solutions are represented as permutations, such as ordering problems, voting methods have been widely applied. In problems where solutions are represented as binary vectors, such as knapsack problems, probabilistic scores have provided very good results [13]. This way, one can also view path-relinking as a unified combination method for all types of problems and in this way it also generalizes the combination methods. In Section 3, we focus on path-relinking, including its mechanics, implementation issues, randomization, the use of pools of high-quality solutions to hybridize path-relinking with other heuristic methods, and evolutionary path-relinking.

Concluding remarks are made in Section 4, where some successful applications of scatter search and of path-relinking are listed.

2 Scatter search

From an algorithmic point of view we can consider that scatter search basically performs iterations over a set of good solutions called the Reference Set (*RefSet*). It must be noted that the meaning of good is not restricted here to the quality of the solutions, but also considers the diversity that they add to this set of solutions.

Once the initial *RefSet* is created, a global iteration of the method consists of three steps: combine, improve, and update the solutions in the *RefSet*. We first describe the five elements in the template. Next, we explain how they interact.

1. A *Diversification Generation Method* to generate a collection of diverse trial solutions, using one or more arbitrary trial solutions (or seed solutions) as an input.
2. An *Improvement Method* to transform a trial solution into one or more enhanced trial solutions: neither the input nor the output solutions are required to be feasible, though the output solutions are typically feasible. If the input trial solution is not improved as a result of the application of this method, the "enhanced" solution is considered to be the same as the input solution.
3. A *Reference Set Update Method* to build and maintain a reference set consisting of the b "best" solutions found (where the value of b is typically small, e.g., no more than 20), organized to provide efficient access by other parts of the solution procedure. Several alternative criteria may be used to add solutions to the reference set and delete solutions from the reference set.
4. A *Subset Generation Method* to operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions. The most common subset generation method is to generate all pairs of reference solutions (i.e., all subsets of size 2).
5. A *Solution Combination Method* to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solutions.

Figure 1 shows the interaction among these five methods and highlights the central role of the reference set. This basic design starts with the creation of an initial set of solutions P, and then extracts from it the reference set (*RefSet*) of solutions. The darker circles represent improved solutions resulting from the application of the Improvement Method.

The Diversification Generation Method is used to build a large set P of diverse solutions. The size of P (*PSize*) is typically at least ten times the size of *RefSet*. The initial reference set is built according to the Reference Set Update Method. For example, the Reference Set Update Method could consist of selecting b distinct and maximally diverse solutions from P.

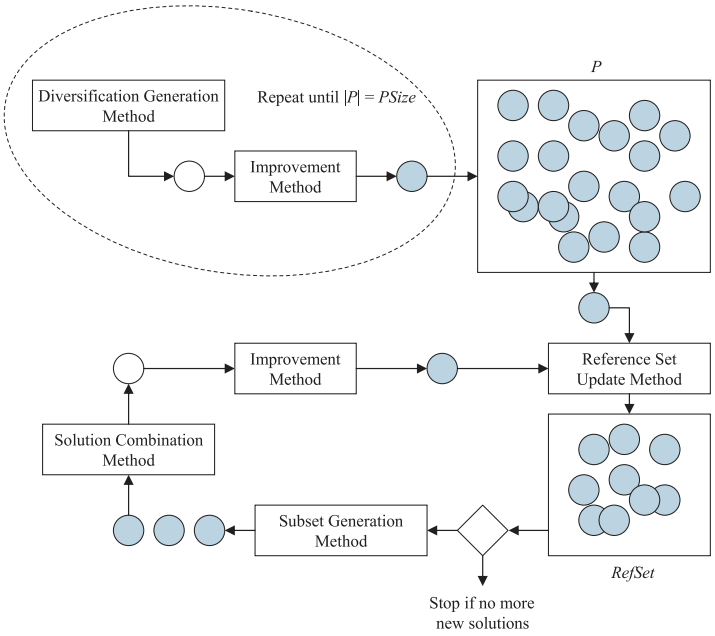


Fig. 1 Scatter search diagram.

A typical construction of the initial reference set starts with the selection of the best $b/2$ solutions from P. These solutions are added to *RefSet* and deleted from P. For each solution in *P-RefSet*, the minimum of the distances to the solutions in *RefSet* is computed. Then, the solution with the maximum of these minimum distances is selected. This solution is added to *RefSet* and deleted from P and the minimum distances are updated. (In applying this max-min criterion, or any criterion based on distances, it can be important to scale the problem variables, to avoid a situation where a particular variable or subset of variables dominates the distance measure and distorts the appropriate contribution of the vector components.) The process is repeated $b/2$ times. The resulting reference set has $b/2$ high quality solutions and $b/2$ highly-diverse solutions. Note that with this criterion we are considering as equally important quality and diversity in the original *RefSet*. Alternative designs may include a different composition of the b solutions in this set. For example, we could consider just a single solution selected because of its quality (say the best one in P) and the remaining $b-1$ solutions in the *Refset* could be selected from P because of their diversity. Since the reference set is the heart of a scatter search procedure, its initial composition may result in significant changes during the search process.

The solutions in *RefSet* are ordered according to quality, where the best solution is the first one in the list. The search is then initiated applying the Subset Generation Method. In its simplest (and typical) form it consists of generating all pairs of reference solutions. That is, the method would focus on subsets of size 2 resulting

in $(b^2 - b)/2$ new subsets. The pairs are selected one at a time in lexicographical order and the Solution Combination Method is applied to generate one or more trial solutions. These trial solutions are subjected to the Improvement Method, if one is available. The Reference Set Update Method is applied once again to build the new *RefSet* with the best solutions, according to the objective function value, from the current *RefSet* and the set of trial solutions. A global iteration finishes with the update of the *RefSet*. Note that in subsequent iterations we only combine the pairs of solutions not combined in previous iterations. The basic procedure terminates after all the generated subsets are subjected to the Combination Method and none of the improved trial solutions are admitted to *RefSet* under the rules of the Reference Set Update Method. However, in advanced scatter search designs, the *RefSet* rebuilding is applied at this point and the best $b/2$ solutions are kept in the *RefSet* and the other $b/2$ are selected from P, replacing the worst $b/2$ solutions.

It is interesting to observe similarities and contrasts between scatter search and the original Genetic Algorithm (GA) proposals. Both are instances of what are sometimes called population-based or evolutionary approaches. Both incorporate the idea that a key aspect of producing new elements is to generate some form of combination of existing elements. However, original GA approaches were predicated on the idea of choosing parents randomly to produce offspring, and further on introducing randomization to determine which components of the parents should be combined. By contrast, scatter search is based on deterministic designs in which we implement strategic rules to generate new solutions. These rules do not resort to randomization, as usually happens in GAs. They are based on the structure and properties of the problem being solved, as well as on the search history. Moreover, GAs usually apply general purpose combination methods, such as the well-known crossover operator, while scatter search customizes the combination method for each particular problem. It should be noted, however, that GAs have been progressively incorporating more advanced design elements from more powerful metaheuristics and solution strategies.

2.1 New strategies in global optimization

Egea et al. [2] proposed an evolutionary method for global optimization of complex-process models, which employs some elements of scatter search and path-relinking. Regarding scatter search, the method uses a relatively small population size, partially chosen by a quality criterion from an initial set of diverse solutions. It also performs systematic combinations among the population members. Regarding path relinking, the new solutions are generated within the areas defined by every pair of solutions in the population, introducing a bias to generate new solutions which share more properties with the best population members than with the rest. We mentioned this method here because it introduces new strategies and modifies some standard scatter search designs. Specifically, it employs:

- a small population without memory structures, in which repeated sampling is allowed;
- a new combination method based on wide hyper-rectangles;
- an aggressive population update for a quick convergence; and
- a new search intensification strategy called *the go-beyond*.

Considering its potential applicability to other domains, we describe the *go-beyond* strategy, which consists in exploiting promising directions, extending the combination method.

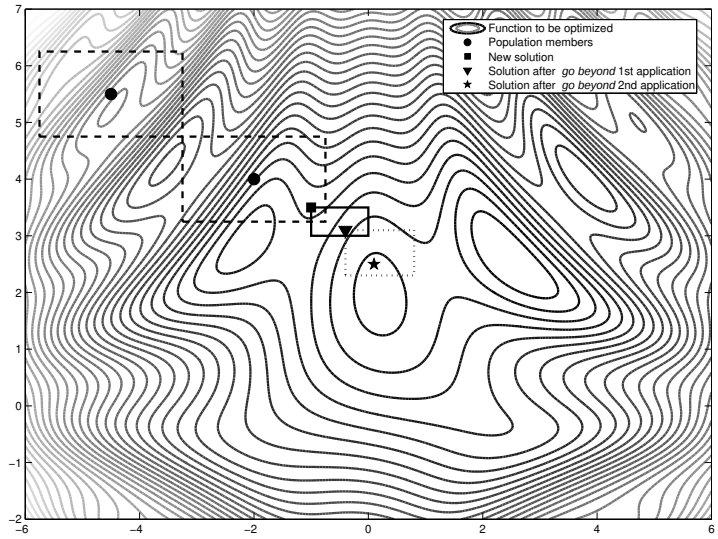
Figure 2 depicts the level curve (contour plots) of the 2-D dimensional unconstrained function $f(x_1,x_2)$ in the range $x_1 \in [-6,6]$,$x_2 \in [-2,7]$, which presents several minima:

$$f(x_1,x_2) = 2 + 0.01(x_2 - x_1^2)^2 + (1-x_1)^2 + 2(2-x_2)^2 + 7\sin(0.5x_1)\sin(0.7x_1x_2)$$

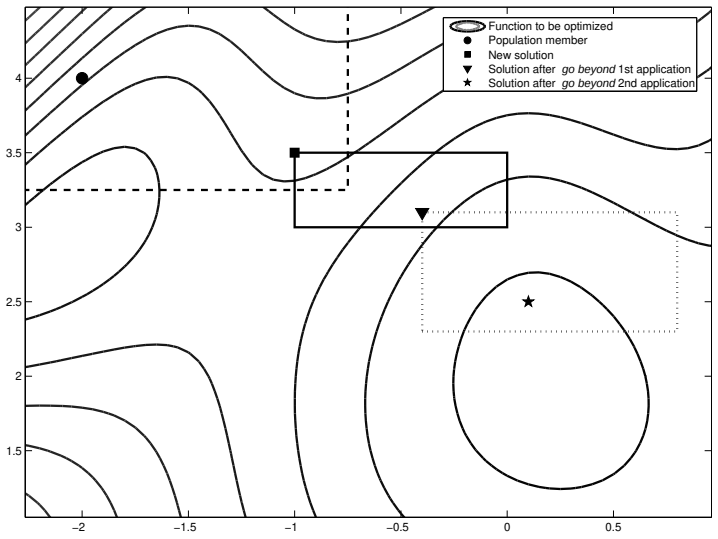
We illustrate in this diagram how the *go-beyond* strategy works. From a pair of *RefSet* solutions x and y (labeled as *population members* in the figure and depicted with black points) a new solution is generated in the corresponding hyper-rectangle, z, and depicted in the figure (labeled as *new solution* and represented with a black square). If z is better than x and y ($f(z) < f(x)$ and $f(z) < f(y)$), then we consider that this is a promising direction and apply the *go-beyond* strategy, *extending* the combination method. In the present problem, this means that we consider a new hyper-rectangle (solid line) defined by the distance between z and y (its closest reference set solution). A new solution (depicted with a triangle) is created in this hyper-rectangle and the process is repeated as long as good solutions are obtained. Figure 2 shows a new solution (starred) created in an area very close to the global minimum.

2.2 New strategies in combinatorial optimization

Martí et al. [16] proposed a scatter search algorithm for the well known Max-Cut problem based on the standard design described in this section. Their method extends the basic scatter search implementation in three different ways. First, it uses a new selection procedure for constructing a reference set from a population of solutions. Traditionally, scatter search implementations have used the criterion of maximizing the minimum distance between the solution under consideration and the solutions already in the reference set. In such a process, diverse solutions are selected one by one from the population P and the distances are updated after each selection. In contrast, Martí et al. [16] propose a method that selects all the diverse solutions at once by solving the *maximum diversity problem* (MDP). Given a set of elements $\mathscr{S}$ and the corresponding distances between the elements of the set, the MDP consists in finding the most diverse subset of $\mathscr{S}$ of a specified size. The diversity of the chosen subset is given by the sum of the distances between each pair



(a) Level curve



(b) Zoom

Fig. 2 The *go-beyond* strategy.

of its elements. The distance between two Max-Cut solutions is defined to be the number of different edges in the cut.

The use of the MDP within scatter search is based on recognizing that the original set of elements is given by $P \setminus \{\text{the } b/2 \text{ best solutions}\}$. The MDP scheme is also used to complete the current *RefSet*, which is already partially populated with the $b/2$ best solutions from P.

The second extension consists of a dynamic adjustment of the depth parameter k associated with the ejection chain mechanism, which is at the core of the search-based improvement method. This local search has an associated parameter that measures the depth of the search in the ejection chain process. The solution representation incorporates the information related to the particular k value used to generate it. In this way, the depth of the ejection chain produced depends on the parameter values associated with the solutions being combined.

The third extension implements a probabilistic selection of the combination methods. The probability of selecting one of three methods proposed in [16] for the Max-Cut problem is proportional to the number of high quality solutions generated by the method in previous iterations. A probability-based mechanism is introduced to select a combination method each time the solutions are combined. The probability of selecting one of the three methods is set to 1/3 at the beginning of the search. The probability values are then updated at the end of each SS iteration in order to favor the combination methods that produce solutions of sufficiently high quality to be included in the reference set.

3 Path-relinking

Path-relinking was originally proposed by Glover [6] as an intensification strategy to explore trajectories connecting elite solutions obtained by tabu search or scatter search [8, 9, 10]. In the remainder of this chapter, we focus on path-relinking, including its mechanics, implementation issues, randomization, the use of pools of high-quality solutions to hybridize path-relinking with other heuristic methods, and evolutionary path-relinking. We conclude the chapter with some computational results illustrating the effect of using path-relinking with other heuristics. For completeness, we have included in this section some material that also appears in the chapter of the handbook on GRASP.

3.1 Mechanics of path-relinking

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

S. Path-relinking is usually carried out between two solutions: one is called the *initial solution*, while the other is the *guiding solution*. One or more paths in the solution space graph connecting these solutions are explored in the search for better solutions. Local search is applied to the best solution in each of these paths, since there is no guarantee that this solution is locally optimal.

Let $s \in S$ be a node on the path between an initial solution and a guiding solution $g \in S$. Not all solutions in the neighborhood $N(s)$ are allowed to follow s on the path from s to g. We restrict the choice to those solutions in $N(s)$ that are more similar to g than s is. This is accomplished by selecting moves from s that introduce attributes contained in the guiding solution g. Therefore, path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions (i.e. the guiding solutions), by favoring these attributes in the selected moves. After an analysis of each potential move, the most common strategy is to select a move that results in the best-quality restricted neighbor of s. The restricted neighbors of s are all solutions in the neighborhood of s that incorporate an attribute of the guiding solution not present in s.

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

Suppose that path-relinking is be applied to a minimization problem between solutions x_1 and x_2 such that $z(x_1) \leq z(x_2)$, where $z(\cdot)$ denotes the objective function. In *forward* path-relinking, the initial and guiding solutions are set to $g = x_1$ and $s = x_2$. Conversely, in *backward* path-relinking, we set $g = x_2$ and $s = x_1$. In *back-and-forward* path-relinking, backward path-relinking is applied first, followed by forward path-relinking. Path-relinking explores the neighborhood of the initial solution more thoroughly than the neighborhood of the guiding solution because, as it moves along the path, the size of the restricted neighborhood decreases. Consequently, backward path-relinking tends to do better than forward path-relinking. Back-and-forward path-relinking does at least as well as either backward or forward path-relinking but takes about twice as long to compute.

In applying *mixed path-relinking* [11, 21] between feasible solutions s and t in S, two paths are started simultaneously, one at s and the other at t. These two paths meet at some solution $r \in S$, thus connecting s and t with a single path. Algorithm 1 describes a mixed path-relinking procedure for a 0-1 minimization problem, such as the set covering problem, where x^s and x^t are binary vectors representing the solutions to be linked.

The set $\Delta = \{j = 1, \ldots, n \ : \ x_j^s \neq x_j^t\}$ of positions in which x^s and x^t differ is computed in line 2. The cardinality of this set is called the *Hamming distance* between x^s and x^t. The best solution, x^*, among x^t and x^s and its cost, $z^* = z(x^*)$, are determined in lines 3 and 4, respectively. The current path-relinking solution, x, is initialized to x^s is line 5. The loop in lines 6 to 16 progressively determines the next solution in the path connecting x^s and x^t, until the entire path is traversed. For every position $\ell \in \Delta$, we define $x \oplus \ell$ to be the solution obtained from x by complementing the current value of x_ℓ. Line 7 determines the component ℓ^* of Δ for which $x \oplus \ell$

```
1  MixedPathRelinking
2  Δ ← {j = 1,...,n : x^s_j ≠ x^t_j};
3  x* ← argmin{z(x^s), z(x^t)};
4  z* ← min{z(x^s), z(x^t)};
5  x ← x^s;
6  while |Δ| > 1 do
7  |   ℓ* ← argmin{z(x ⊕ ℓ) : ℓ ∈ Δ};
8  |   Δ ← Δ \ {ℓ*};
9  |   x_ℓ ← 1 − x_ℓ;
10 |   if z(x) < z* then
11 |   |   x* ← x;
12 |   |   z* ← z(x);
13 |   end
14 |   x^s ← x^t;
15 |   x^t ← x;
16 end
17 x ← LocalSearch(x);
18 return x ;
```

Algorithm 1: Mixed path-relinking procedure for problems where solutions are represented by binary vectors.

results in the least-cost solution. This component is removed from Δ in line 8 and the current solution is updated in line 9 by complementing the value of its ℓ-th position. If the test in line 10 detects that the new current solution x improves the best solution x^* in the path, then x^* and its cost are updated in lines 11 and 12, respectively. The roles of the starting and target solutions are swapped in lines 14 and 15 to implement the mixed path-relinking strategy. If $|\Delta| = 0$, then the local search is applied to the best solution in the path in line 16 and the locally optimal solution is returned by the procedure.

Like back-and-forward path-relinking, the mixed variant explores both neighborhoods $N(x^s)$ and $N(x^t)$. Unlike back-and-forward path-relinking, it is usually less than twice as long as the backward or forward variants.

In the case of the set covering problem, there always exists a path connecting x^s and x^t. We just need to observe that setting to one all components with value 0 in x^s and value 1 in x^t results in a series of feasible covers leading from x^s to some feasible solution x. Next, by setting to zero those components with value 1 in x and value 0 in x^t results again in a series of feasible covers leading x to x^t. Figure 3 illustrates the application of mixed path-relinking to solutions x^s and x^t for which the Hamming distance is equal to five.

One can expect to see most solutions produced by path-relinking to come from subpaths close to either the initiating or guiding solutions. Resende et al. [18] showed that this occurs in instances of the max-min diversity problem. In that experiment, a back and forward path-relinking scheme was tested. Figure 4 shows the percentage of best solutions found by path-relinking taken over several instances and several applications of path-relinking. The 0-10% range in the figure corre-

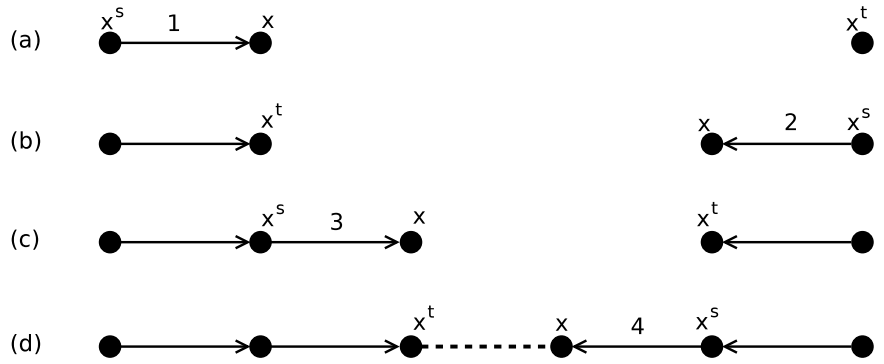


Fig. 3 Mixed path-relinking between two solutions with Hamming distance of five: numbers above the arrows represent the order in which the moves are performed.

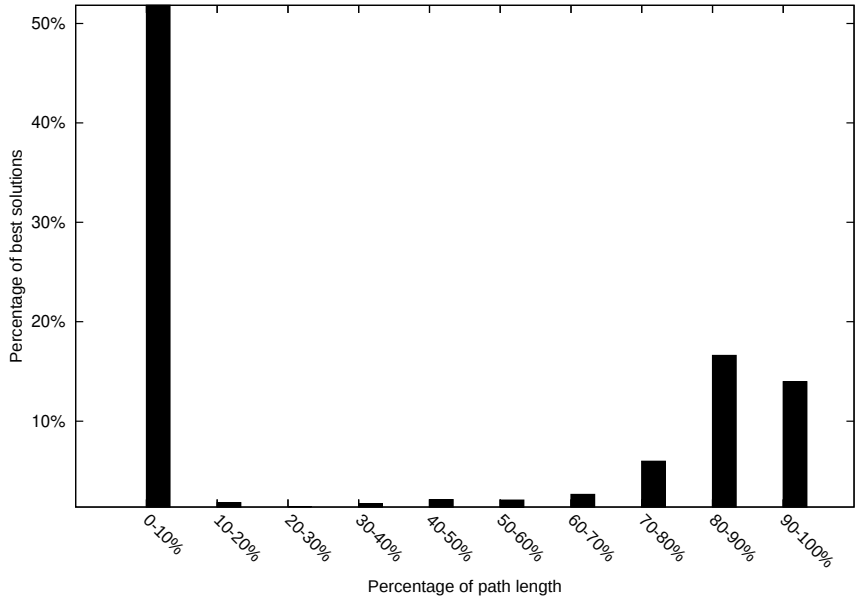


Fig. 4 Percentage of best solutions found at different depths of the path from the initial solution to the guiding solution on instances of the max-min diversity problem.

sponds to subpaths near the initial solutions for the forward path-relinking phase as well as the backward phase, while the 90-100% range are subpaths near the guiding solutions. As the figure indicates, exploring the subpaths near the extremities may produce solutions about as good as those found by exploring the entire path. There is a higher concentration of better solutions close to the initial solutions explored by path-relinking.

As shown in Algorithm 2, it is simple to adapt path-relinking to explore only the neighborhoods close to the extremes. Let ρ be a real parameter such that $0 < \rho \leq 1$ that defines the portion of the path to be explored. Instead of carrying out the main loop while $|\Delta| > 1$ as in the mixed path-relinking of Algorithm 1, the main loop is applied while $|\Delta| > \rho \cdot \delta_t$, where δ_t is the cardinality of the initial set Δ.

```
1  TruncatedMixedPathRelinking
2  Δ ← {j = 1,...,n : x^s_j ≠ x^t_j};
3  δ_t ← |Δ|;
4  x* ← argmin{z(x^s), z(x^t)};
5  z* ← min{z(x^s), z(x^t)};
6  x ← x^s;
7  while |Δ| > ρ · δ_t do
8  |   ℓ* ← argmin{z(x ⊕ ℓ) : ℓ ∈ Δ};
9  |   Δ ← Δ \ {ℓ*};
10 |   x_ℓ ← 1 − x_ℓ;
11 |   if z(x) < z* then
12 |   |   x* ← x;
13 |   |   z* ← z(x);
14 |   end
15 |   x^s ← x^t;
16 |   x^t ← x;
17 end
18 x ← LocalSearch(x);
19 return x ;
```

Algorithm 2: Truncated mixed path-relinking procedure for problems where solutions are represented by binary vectors.

3.2 Minimum distance required for path-relinking

We assume that we want to connect solutions s and t with path-relinking. If the distance $|\Delta(s,t)|$ between s and t, i.e. the number of components in which s and t differs, is equal to one, then the path directly connects the two solutions and no solution, other than s and t, is visited.

If we assume that s and t are both locally optimal, we know that $z(s) \leq z(r)$ for all $r \in N(s)$ and $z(t) \leq z(r)$ for all $r \in N(t)$. If $|\Delta(s,t)| = 2$, then any path is of the type $s \rightarrow r \rightarrow t$, where $r \in N(s) \cap N(t)$, and consequently r cannot be better than either s or t. Likewise, if $|\Delta(s,t)| = 3$ then any path is of the type $s \rightarrow r_s \rightarrow r_t \rightarrow t$, where $r_s \in N(s)$ and $r_t \in N(t)$, and consequently neither r_s nor r_t can be better than both s and t.

Therefore, things only get interesting for $|\Delta(s,t)| > 3$. For those cases, any path is of the type $s \rightarrow r_s \rightarrow w_1 \rightarrow \cdots \rightarrow w_p \rightarrow r_t \rightarrow t$, where $w_1, \ldots, w_p$ are candidates

to be better than both s and t. Therefore, we do not consider relinking a pair of solutions s,t unless $|\Delta(s,t)| \geq 4$.

3.3 Randomization in path-relinking

Consider again a problem whose solution can be represented as a binary vector of size n, such as the set covering problem, the satisfiability problem, or the max-cut problem. Let us denote the set of solutions spanned by the common elements of solutions s and t as

$$\mathscr{S}(s,t) := \{w \in \{0,1\}^n : w_i = s_i = t_i, i \notin \Delta(s,t)\} \setminus \{s,t\}, \tag{1}$$

with $|\mathscr{S}(s,t)| = 2^{|\Delta(s,t)|} - 2$. The underlying assumption of path-relinking is that there exist good-quality solutions in $\mathscr{S}(s,t)$, since this space consists of all solutions which contain the common elements of two good solutions s and t. Taking into consideration that the size of this space is exponentially large, we normally adopt a greedy search where a path of solutions

$$s = w_0, w_1, \ldots, w_{|\Delta(s,t)|} = t,$$

is constructed, such that $|\Delta(w_i, w_{i+1})| = 1,\ \ i = 0, \ldots, |\Delta(s,t)| - 1$, and the best solution from this path is chosen. However, by adopting the greedy strategy, we limit ourselves to exploring a single path from a set of exponentially many paths. By adding randomization to path-relinking, *greedy randomized adaptive* path-relinking (GRAPR) [3] is not constrained to explore a single path.

The pseudo-code for GRAPR for a minimization problem is shown in Algorithm 3. The main difference with respect to Algorithm 1 are lines 6, and 8–11. Instead of selecting the move that results in the best solution as is the case in standard path-relinking, a restricted candidate list (*RCL*) is constructed with the moves that result in solutions with costs in an interval that depends on the value of the best move, the value of the worst move, and a random parameter α. From this set, one move is selected at random to produce the next step in the path.

GRAPR is useful when path-relinking is applied more than once between the same pair of solutions as it can occur in evolutionary path-relinking (discussed in Subsection 3.5).

3.4 Hybridization with a pool of elite solutions

Path-relinking is a major enhancement to metaheuristics that generate a sequence of locally optimal feasible solutions. These metaheuristics include, but are not limited to, GRASP, variable neighborhood search, tabu search, scatter search, and simulated annealing, To hybridize path-relinking with these metaheuristics, one usually makes

```
1  GreedyRandomizedAdaptivePathRelinking
2  Δ ← {j = 1,...,n : x_j^s ≠ x_j^t};
3  x* ← argmin{z(x^s), z(x^t)};
4  z* ← min{z(x^s), z(x^t)};
5  x ← x^s;
6  Select α ∈ [0,1] ⊂ ℝ at random;
7  while |Δ| > 1 do
8  |   z⁻ ← min{z(x ⊕ ℓ) : ℓ ∈ Δ};
9  |   z⁺ ← max{z(x ⊕ ℓ) : ℓ ∈ Δ};
10 |   RCL ← {ℓ ∈ Δ : z(x ⊕ ℓ) ≤ z⁻ + α(z⁺ − z⁻)};
11 |   Select ℓ* ∈ RCL at random;
12 |   Δ ← Δ \ {ℓ*};
13 |   x_ℓ ← 1 − x_ℓ;
14 |   if z(x) < z* then
15 |   |   x* ← x;
16 |   |   z* ← z(x);
17 |   end
18 |   x^s ← x^t;
19 |   x^t ← x;
20 end
21 x ← LocalSearch(x);
22 return x ;
```

Algorithm 3: Greedy randomized adaptive path-relinking with a mixed variant of path-relinking.

use of an *elite set*, i.e. a diverse pool of high-quality solutions found during the search. The elite set starts empty and is limited in size. Each locally optimal solution produced by the metaheuristic is relinked with one or more solutions from the elite set. Each solution produced by path-relinking is a candidate for inclusion in the elite set where it can replace an elite solution of worse value.

The pool of elite solutions is initially empty. Each locally optimal solution produced by the metaheuristic and each solution resulting from path-relinking is considered as a candidate to be inserted into the pool. If the pool is not yet full, the candidate is simply added to the pool if it differs from all pool members. If the pool is full and the candidate is better than the incumbent, then it replaces an element of the pool. In case the candidate is better than the worst element of the pool but not better than the best element, then it replaces some element of the pool if it is sufficiently different from every other solution currently in the pool. To balance the impact on pool quality and diversity, the element selected to be replaced is the one that is most similar to the entering solution among those elite solutions of quality no better than the entering solution [20].

Given a local optimum s_1 produced by the metaheuristic, we need to select at random from the pool a solution s_2 to be connected with s_1 via path-relinking. In principle, any solution in the pool could be selected. However, one should avoid solutions that are too similar to s_1, because relinking two solutions that are similar

limits the scope of the path-relinking search. If the solutions are represented by binary vectors, one should favor pairs of solutions for which the Hamming distance between them is high. A strategy introduced in [20] is to select a pool element at random with probability proportional to the Hamming distance between the pool element and the local optimum s_1. Since the number of paths between two solutions grows exponentially with their Hamming distance, this strategy favors pool elements that have a large number of paths connecting them to and from s_1.

```
1  HEUR+PR
2  Initialize elite set P ← ∅;
3  while stopping criterion not satisfied do
4  |   x ← HeuristicLocalOptimal();
5  |   if P = ∅ then insert x into P;
6  |   else
7  |   |   x^s ← x;
8  |   |   Choose, at random, a pool solution x^t ∈ P;
9  |   |   x ← PathRelinking(x^s, x^t);
10 |   |   Update the elite set P with x ;
11 |   end
12 end
13 return P ;
```

Algorithm 4: Hybridization of path-relinking with a heuristic that generates local optima.

Algorithm 4 illustrates the pseudo-code of a hybrid heuristic that uses path-relinking for minimization. In line 2, the pool of elite solutions P is initially empty. The loop in lines 3 to 12 makes up an iteration of the hybrid algorithm. In line 4, x is a locally optimal solution generated by procedure `HeuristicLocalOptimal()`. If the elite set is empty, then x is inserted into the pool in line 5. Otherwise, x becomes the initiating solution in lines 7 and a guiding solution is selected at random from the pool in line 8. The initiating and guiding solutions are relinked in line 9 and the resulting solution is tested for inclusion into the elite set in line 10. The hybrid procedure returns the set of elite solutions which includes the best solution found during the search.

3.5 Evolutionary path-relinking

Path-relinking can also be applied between elite set solutions to search for new high-quality solutions and to improve the average quality of the elite set. This can be done in a post-optimization phase, after the main heuristic stops, or periodically, when the main heuristic is still being applied [1, 18, 20].

We describe two schemes called *evolutionary path-relinking* for this purpose. Both schemes take as input the elite set and return either the same elite set or one with an improved average cost.

The first scheme, described by Resende and Werneck [20], works with a population that evolves over a number of generations. The initial population is the input elite set. In the k-th generation the procedure builds the k-th population, which is initially empty. Path-relinking is applied between all pairs of solutions in population $k-1$. Each solution output from the path-relinking operation is a candidate for inclusion in population k. The usual rules for inclusion into an elite set are adopted in evolutionary path-relinking. If population k is not yet full, the solution is accepted if it differs from all solutions in the population. After population k is full, the solution is accepted if either it is better than the best solution in the population or it is better than the worst and is sufficiently different from all solutions in the population. Once a solution is accepted for inclusion into population k, it replaces the solution in population k that does not have a better cost and that is most similar to it. The procedure halts when the best solution in population k does not have better cost than the best solution in population $k-1$.

A variation of the above scheme is described by Resende et al. [18]. In that scheme, while there exists a pair of solutions in the elite set for which path-relinking has not yet been applied, the two solutions are combined with path-relinking and the resulting solution is tested for membership in the elite set. If it is accepted, it then replaces the elite solution most similar to it among all solutions having worse cost.

Since some elite solutions may remain in the elite set over several applications of evolutionary path-relinking, greedy randomized adaptive path-relinking [3] can be used in evolutionary path-relinking to avoid repeated explorations of the same paths in the solution space in different applications of the procedure.

GRASP with evolutionary path-relinking and scatter search are evolutionary methods based on evolving a small set of selected solutions (elite set in the former and reference set in the latter). We can, therefore, observe similarities between them. In some implementations of scatter search, GRASP is used to populate the reference set. Note, however, that other constructive methods can be used as well. Similarly, path-relinking can be used to combine solutions in scatter search, but we can use any other combination method. From an algorithmic point of view, we may find two main differences between these methods. The first one is that in scatter search we do not apply path-relinking to the solutions obtained with GRASP, but rather, we only apply path-relinking as a combination method between solutions already in the reference set. The second difference is that in scatter search when none of the new solutions obtained with combinations are admitted to the reference set (elite set), it is rebuilt, removing some of its solutions, as specified in the reference set update method. In GRASP with evolutionary path-relinking we do not remove solutions from the elite set, but rather, we reapply GRASP and use the same rules for inclusion in the elite set.

3.6 Progressive crossover: Hybridization with genetic algorithms

Path-relinking was first applied in the context of a genetic algorithm by Ribeiro and Vianna [22] in order to implement a progressive crossover operator. In this innovative application, the hybridization strategy was applied to a phylogeny problem.

The original proposal was extended and improved in [23]. In this case, a bidirectional (or back and forward) path-relinking strategy is used: given two parent solutions s_1 and s_2, one path is computed leading from s_1 to s_2 and another leading from s_2 to s_1. The best solution along them is returned as the offspring resulting from crossover. This mechanism is an extension of the traditional crossover operator: instead of producing only one offspring, defined by one single combination of two parents, it investigates many solutions that share characteristics of the selected parents. The solution found by path-relinking corresponds to the best offspring that could be obtained by applying the standard crossover to the parents.

The experiments reported in [23] make use of the results obtained on one randomly generated instance (TST17) of the phylogeny problem to assess the evolution of the solutions found by three different genetic algorithm in one hour (3,600 seconds) of computations: the random-keys genetic algorithm RKGA [22], the proposed genetic algorithm GA+PR using path-relinking to implement the progressive crossover operator, and the simpler genetic algorithm GAUni using uniform crossover. Figure 5 presents the solution value at the end of each generation for each of the 100 individuals in the population. Since the original random-keys genetic algorithm RKGA made use of elitism, the solution values are restricted to a smaller interval ranging between 2500 and 2620. The solution values obtained by the two other algorithms show more variability. The solutions found by algorithm GA+PR are better than those obtained by RKGA and GAUni, illustrating the contribution of the strategy based on path-relinking to implement the crossover operator.

Path-relinking was also applied by Zhang and Lai [25] following the strategy proposed in [22] in the implementation of a genetic algorithm for the multiple-level warehouse layout problem. Their approach also makes use of path-relinking when the genetic algorithm seems to be trapped in a locally optimal solution. Once again, path-relinking was used by Vallada and Ruiz [24] as a progressive crossover operator within a genetic algorithm for the minimum tardiness permutation flowshop problem. It was also applied as an intensification strategy after a number of generations without improvement to the best solution. The selected individuals are marked in order to not be selected again during the application of path-relinking. Path-relinking was also hybridized with a genetic algorithm as a post-optimization procedure [17]. In this work, the solutions in the final population produced by the genetic algorithm are progressively combined and refined.

3.7 Hybridization of path-relinking with other heuristics

The basic implementation of GRASP is memoryless because it does not make use of information collected in previous iterations. 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 [12]. It was followed by several extensions, improvements, and successful applications [19]. Each local minimum produced by the GRASP is combined with a randomly selected elite solution. The resulting solution is a candidate for inclusion into the elite set. Evolutionary path-relinking can be applied periodically to improve the quality of the elite set.

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

4 Applications and concluding remarks

There are three main sources where successful applications of scatter search and path-relinking can be found. First, Chapter 8 of the monograph on scatter search by Laguna and Martí [13], identifies 14 applications, including neural networks, multi and mono-objective routing problems, graph drawing, scheduling, and coloring problems. A second source of successful implementations of both methodologies is a special issue of EJOR [14] in which they are classified into the following seven categories: Foundations, Nonlinear Optimization, Optimization in Graphs, Parallel Optimization, Prediction and Clustering, Routing and Scheduling. There is also a third source, which is frequently updated with current applications: the web site `http://www.uv.es/rmarti/scattersearch` on scatter search and path-relinking publications, in which more than 100 implementations are collected.

References

1. R.M. Aiex, P.M. Pardalos, M.G.C. Resende, and G. Toraldo. GRASP with path-relinking for three-index assignment. *INFORMS J. on Computing*, 17:224–247, 2005.
2. J. Egea, R. Martí, and J. Banga. An evolutionary method for complex-process optimization. *Computers and Operations Research*, 2009. `doi:10.1016/j.cor.2009.05.003`.

3. H. Faria Jr., S. Binato, M.G.C. Resende, and D.J. Falcão. Transmission network design by a greedy randomized adaptive path relinking approach. *IEEE Transactions on Power Systems*, 20:43–49, 2005.
4. P. Festa, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. GRASP with path-relinking for the weighted MAXSAT problem. *ACM J. of Experimental Algorithmics*, 11:1–16, 2006.
5. F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, pages 156–166, 1977.
6. F. Glover. Tabu search and adaptive memory programing – Advances, applications and challenges. In R.S. Barr, R.V. Helgason, and J.L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer Academic Publishers, 1996.
7. F. Glover. A template for scatter search and path relinking. In J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution, Third European Conference, AE97*, volume 1363 of *Lecture Notes in Computer Science*, pages 13–54. Springer, 1998.
8. F. Glover. Multi-start and strategic oscillation methods – Principles to exploit adaptive memory. In M. Laguna and J.L. Gonzáles-Velarde, editors, *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pages 1–24. Kluwer Academic Publishers, 2000.
9. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
10. F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.
11. F. Glover, M. Laguna, and R. Martí. Scatter search and path relinking: Foundations and advanced designs. In Godfrey C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, volume 141 of *Studies in Fuzziness and Soft Computing*, pages 87–100. Springer, 2004.
12. M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, 11:44–52, 1999.
13. M. Laguna and R. Martí. *Scatter search: Methodology and implementations in C*. Operations Research/Computer Science Interfaces Series. Kluwer Academic Publishers, Boston, 2003.
14. R. Martí, (Editor). Feature cluster on scatter search methods for optimization. *European J. on Operational Research*, 169(2):351–698, 2006.
15. C.A. Oliveira, P.M. Pardalos, and M.G.C. Resende. GRASP with path-relinking for the quadratic assignment problem. In C.C. Ribeiro and S.L. Martins, editors, *Proceedings of III Workshop on Efficient and Experimental Algorithms*, volume 3059, pages 356–368. Springer, 2004.
16. Martí R., A. Duarte, and M. Laguna. Advanced scatter search for the max-cut problem. *INFORMS Journal on Computing*, 21:26–38, 2009.
17. M. Ranjbar, F. Kianfar, and S. Shadrokh. Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. *Applied Mathematics and Computation*, 196:879–888, 2008.
18. M.G.C. Resende, R. Martí, M. Gallego, and A. Duarte. GRASP and path relinking for the max-min diversity problem. *Computers and Operations Research*, 2008. Published online 28 May 2008, doi:10.1016/j.cor.2008.05.011.
19. M.G.C. Resende and C.C. Ribeiro. GRASP with path-relinking: Recent advances and applications. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: Progress as Real Problem Solvers*, pages 29–63. Springer, 2005.
20. M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the *p*-median problem. *J. of Heuristics*, 10:59–88, 2004.
21. C.C. Ribeiro and I. Rosseti. A parallel GRASP heuristic for the 2-path network design problem. In *Proceedings of Euro-Par 2002*, volume 2400 of *Lecture Notes in Computer Science*, pages 922–926. Springer-Verlag, Paderborn, 2002.
22. C.C. Ribeiro and D.S. Vianna. A genetic algorithm for the phylogeny problem using an optimized crossover strategy based on path-relinking. In *Anais do II Workshop Brasileiro de Bioinformtica*, pages 97–102, Macaé, 2003.

23. C.C. Ribeiro and D.S. Vianna. A hybrid genetic algorithm for the phylogeny problem using path-relinking as a progressive crossover strategy. *International Transactions in Operational Research*, 16(5), 2009. To appear.
24. E. Vallada and R. Ruiz. New genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, 2009. in press, doi:10.1016/j.omega.2009.04.002.
25. G.Q. Zhang and K.K. Lai. Combining path relinking and genetic algorithms for the multiple-level warehouse layout problem. *European Journal of Operational Research*, 169:413–425, 2006.

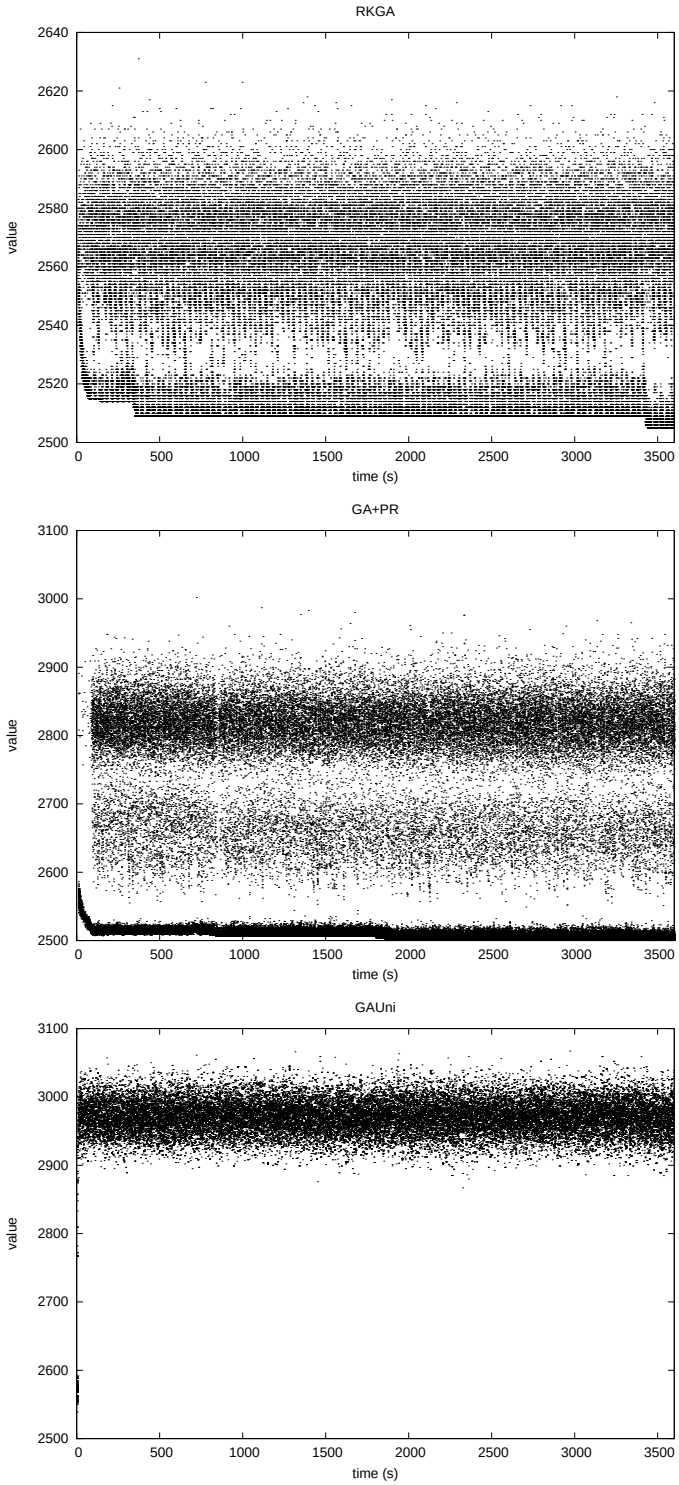


Fig. 5 Solutions obtained by genetic algorithms for random instance TST17 for 3,600 seconds of computations.

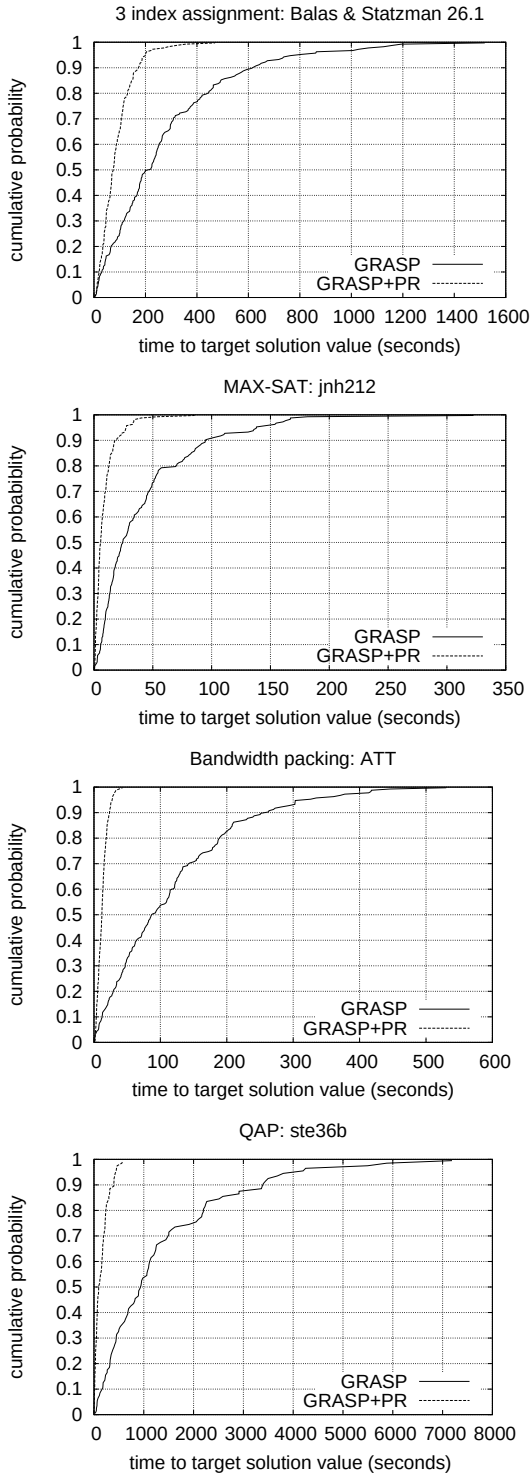


Fig. 6 Time to target plots comparing running times of pure GRASP and GRASP with path-relinking on four instances of distinct problem types: three index assignment [1], maximum satisfiability [4], bandwidth packing [4], and quadratic assignment [15].