A hybrid genetic algorithm for the phylogeny problem using path-relinking as a progressive crossover strategy

Celso C. Ribeiro[1]⋆ and Dalessandro S. Vianna[2,3]⋆⋆

[1] Universidade Federal Fluminense, Department of Computer Science
Rua Passo da Pátria 156, Niterói, RJ 24210-240, Brazil.
[2] Universidade Candido Mendes, Núcleo de Pesquisa e Desenvolvimento
Rua Anita Pessanha 100, Campos dos Goytacazes, RJ 28040-320, Brazil.
[3] Centro Federal de Educação Tecnológica, Departamento de Informática
Rua Dr. Siqueira 273, Campos dos Goytacazes, RJ 28030-130, Brazil.

`celso@ic.uff.br, dalessandro@ucam-campos.br`

Abstract. A phylogenetic tree relates taxonomic units using their similarities over a set of characteristics. Given a set of taxonomic units and their characteristics, the phylogeny problem under the parsimony criterion consists in finding a phylogenetic tree with a minimum number of evolutionary steps. We developed a hybrid genetic algorithm for the problem of building a phylogenetic tree minimizing parsimony. The algorithm combines local search with a crossover strategy based on path-relinking, an intensification technique originally used in the context of other metaheuristics such as scatter search and GRASP. Computational experiments on benchmark and randomly generated instances show that the proposed algorithm is very robust and outperforms other heuristics in terms of solution quality and running times.

1 Motivation

A central problem in comparative biology is that of establishing ancestrality relations among n species, groups of species, populations of distinct species, or homologous genes in populations of distinct species [6, 34]. These entities can be indistinctly designated as *taxons*. Ancestrality relations are represented by a rooted tree (leaves represent the taxons under analysis, while internal nodes represent hypothetic ancestrals) which is called a *phylogenetic tree* or a *phylogeny*. Taxons under analysis are called *operational taxons*, while taxons associated to internal nodes of a phylogeny are called *hypothetical taxons*. Each taxon has a

⋆ Celso C. Ribeiro is partially supported by CNPq research grants 301.694/2007-9 and 485.328/2007-0, and by FAPERJ research grant E-152.522/2006.

⋆⋆ Dalessandro S. Vianna is funded by FAPERJ research grant E-26/110.514/2007 and by CNPq research grants 308.586/2006-9 and 472.749/2006-4.

set of m *characteristics*. An instance of the phylogeny problem with binary characteristics is defined by an $m \times n$ 0-1 matrix, in which each element corresponds to the state (existence or absence) of a given character for a given taxon.

Different evaluation criteria exist to infer the quality of a phylogenetic tree. Each state change along a branch of a phylogenetic tree is counted as one evolutionary step. The parsimony criterion states that the best phylogeny is the more parsimonious, i.e., the one that can be explained by the minimum number of evolutionary steps [8, 17, 34]. Parsimony is often legitimated as the most appropriate evaluation criterion for phylogenies, since evolutionary changes occur with very small probabilities [22, 33].

Luckow and Pimentel [21] were the first to compare different mainframe programs for computing phylogenetic trees under the parsimony criterion. Later, Platnick [23] compared phylogeny inference systems running on microcomputers.

Andreatta et al. [4] proposed an object oriented framework for the development of local search heuristics and metaheuristics for combinatorial optimization problems. This framework was used for the development of a family of local search heuristics for the phylogeny problem under the parsimony criterion. It made possible a fair comparison of heuristics running under the same conditions on a set of eight benchmarks problems [5].

Ribeiro and Vianna [31] proposed a GRASP heuristic that found the best known solutions to date for benchmark instances. This heuristic makes use of the k-SPR neighborhood. A move in the latter may be seen as the combination of k consecutive moves in the SPR (*Subtree Pruning and Regrafting*) neighborhood.

Path-relinking was originally proposed as an intensification strategy to explore trajectories connecting elite solutions obtained by tabu search or scatter search [12–15]. It was also successfully used to introduce memory mechanisms in implementations of GRASP [11, 25, 26]. Ribeiro and Vianna [30] reported preliminary results obtained with an innovative hybridization strategy of a genetic algorithm with path-relinking, in which the latter was used to implement a progressive crossover operation. Later, Cotta [7] used path-relinking within an implementation of scatter search for solving the phylogeny problem under a distance-based optimization criterion.

In this work, we extend and improve the hybridization of path-relinking within a genetic algorithm for approximately solving the phylogeny problem under the parsimony criterion. The remaining of this paper is organized as follows. The hybrid genetic algorithm is described in detail in Section 2. The crossover operation based on path-relinking is presented in Section 3. Computational results illustrating the effectiveness of the proposed approach are reported in Section 4. Concluding remarks are drawn in the last section.

2 Hybrid genetic algorithm

The hybrid genetic algorithm described in this work incorporates two optimization strategies. The first is the local search heuristic described in Section 2.3, which is applied to some of the offsprings resulting from crossover. Applying local search to some offsprings with a small probability contributes to improve solution quality, without leading to freezing the population.

The second optimization strategy concerns the progressive crossover operation based on path-relinking [30]. Its detailed description is delayed to Section 3.

The fitness of each individual (or solution) is defined as the inverse of the parsimony value of its associated phylogenetic tree. A number of computational experiments have been carried out to define the best policies for selection and population evolution. Several selection rules were implemented: random or stochastic universal sampling, roulette-wheel or stochastic sampling with replacement, tournament - random variation, tournament - roulette variation, deterministic sampling, stochastic remainder sampling, stochastic remainder sampling with replacement, stochastic sampling without replacement, and ranking [16, 18]. Different population evolution policies were implemented and tested: simple genetic algorithm (SGA), steady state genetic algorithm (SSGA), and GAP parameter [16]. Best results were obtained using the selection method known as remainder sampling with replacement (see Section 2.2 for details) and SGA as the population evolution strategy (all individuals replaced by their offsprings).

Since the main goal of this work consists in reporting and comparing the results obtained by the hybridization of path-relinking within a genetic algorithm to implement a progressive crossover operation, we omitted the numerical results leading to the above choices to be able to focus on the crossover strategy. The main phases of the hybrid genetic algorithm are detailed below.

2.1 Initial population

The genetic algorithm is initialized with a population formed by $SizePop = 100$ individuals generated by the randomized version `GStep_wR` of the greedy `GStep` heuristic [5]. Numerical results have shown that this algorithm finds better solutions than others, although at the cost of slightly higher computation times.

Whenever a taxon i is inserted into a partial phylogeny under construction by algorithm `GStep`, all branches of the latter are evaluated. The chosen branch is that minimizing the insertion cost of taxon i (i.e., the most parsimonious). In its randomized version `GStep_wR`, the branch were taxon i will be inserted is randomly selected from among all those with cost at most 10% higher than the most parsimonious incremental value.

2.2 Selection by stochastic remainder sampling with replacement

The selection policy is based on a temporary population, which is built once for all for each generation with copies of the individuals in the population of this generation.

The temporary population is generated as follows. We first compute the average fitness $f_{med} = (1/SizePop) \sum_{i=1}^{SizePop} f_i$ of the current population, with the fitness f_i of each individual i being the inverse of the parsimony value of its associated phylogenetic tree, for $i = 1, \dots, SizePop$. The fitness f_i of each individual of the population is divided by the average fitness f_{med} and $\lfloor f_i/f_{med} \rfloor$ copies of this individual are inserted into the temporary population. Therefore, only individuals whose fitness is higher than average are candidates for selection. Furthermore, copies of individuals with higher fitness will be more numerous in the temporary population, making them more amenable for selection.

At this point, the temporary population has $\sum_{i=1}^{SizePop} \lfloor f_i/f_{med} \rfloor$ individuals. The solutions needed to complete the temporary population are generated by the application of the roulette-wheel method exactly $SizePop - \sum_{i=1}^{SizePop} \lfloor f_i/f_{med} \rfloor$ times. At each time, an individual $i = 1, \ldots, SizePop$ with $f_i/f_{med} > 1$ is randomly selected from the current population, with a probability proportional to the fractional part of f_i/f_{med}, and a copy of it is added to the temporary population.

Next, two individuals are randomly selected from the temporary population to be submitted to crossover. Their unique offspring is added to the population of the new generation. This step is repeated until a new population with $SizePop$ offsprings is obtained.

2.3 Local search and mutations

The local search heuristic explores the SPR neighborhood [35, 36], also described in [5]. First, a subtree of the current phylogeny is disconnected. Next, it is reconnected to the body of the original tree in a different position. Figure 1 illustrates an example of a move in this neighborhood, in accordance with the steps below:

- Step 1: An edge $q = (c, f)$ of the current phylogeny is selected and removed. The subtree containing node c is the *base subtree*, while that containing node f is the *pending subtree*.
- Step 2: Node c is destroyed in the base subtree and its two adjacent nodes are directly connected by a new edge (a, g) which results from collapsing the original edges (a, c) and (c, g). Next, an edge $r = (b, d)$ of the base subtree is selected for reconnecting the pending subtree.
- Step 3: A new node h is created and edge $r = (b, d)$ is replaced by two edges (b, h) and (h, d) in the base subtree.
- Step 4: The pending subtree is reconnected to the base subtree through node h created in the previous step.

A phylogenetic tree has $O(n)$ potential subtrees and each of them can be reconnected by $O(n)$ possible edges. Therefore, each solution has $O(n^2)$ neighbors in the SPR neighborhood. The local search heuristic follows a best-improving strategy: all neighbors are evaluated and the best improving move is selected. The search stops when no improving move exist.

Mutations are randomly generated moves in the same SPR neighborhood. A node is randomly selected to act as the root of the pending subtree and a branch of the base subtree is randomly selected for reconnecting the pending subtree.

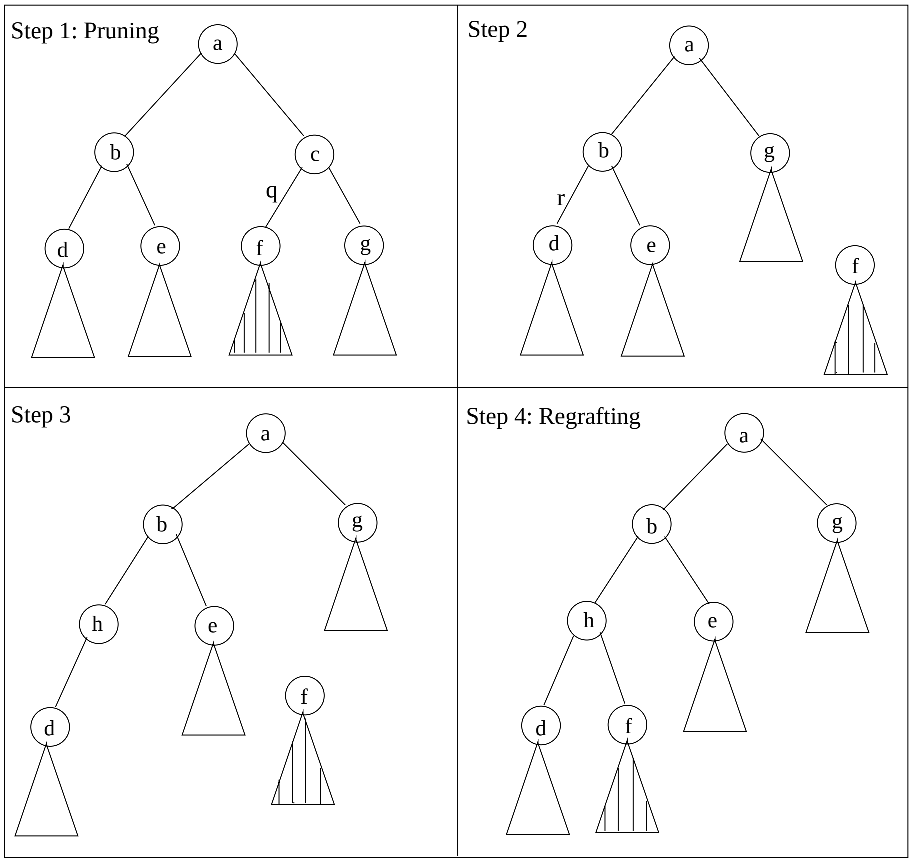

Fig. 1. Illustration of a move in neighborhood SPR.

2.4 Population evolution

The SGA rule revealed itself as the best appropriate for population evolution. Each population is completely replaced at the next generation.

Figure 2 presents the pseudo-code of the main steps of the hybrid genetic algorithm `GA+PR+LS`. The fitness f^* of the best solution found is initialized in line 1. The greedy randomized heuristic `GStep_wR` described in Section 2.1 is applied $SizePop$ times in line 2 to create the initial population. The loop in lines 3-16 evolves the population until a given stopping criterion is met. The temporary population associated with the current generation is built in line 4, as described in Section 2.2. The loop in lines 5-15 ensures that the new population entirely replaces the current one, since exactly $SizePop$ offsprings are generated.

Two individuals $parent_1$ and $parent_2$ are randomly selected from the temporary population in line 6. They are submitted to crossover by path-relinking in line 7, as described later in Section 3. Local search is applied to $offspring$ in line

8 with probability $ProbLS$. A mutation is applied in line 9 to $offspring$ with probability $ProbMut$. Parameters $ProbLS$ and $ProbMut$ were empirically set at 15% and 3%, respectively. If the new solution $offspring$ improves the best solution found, then the latter is updated in lines 10-13. The new individual $offspring$ is inserted into the new population in line 14.

```
Procedure GA+PR+LS
01. $f^* \leftarrow 0$;
02. Apply heuristic GStep_wR exactly $SizePop$ times to build the initial population;
03. While stopping criterion not met do
04.     Build the temporary population associated with the current generation;
05.     For $i = 1$ to $SizePop$ do
06.         Randomly select two individuals $parent_1$ and $parent_2$ for crossover;
07.         Apply the path-relinking crossover to $parent_1$ and $parent_2$, obtaining
            a new individual $offspring$;
08.         Apply local search to $offspring$ with probability $ProbLS$;
09.         Apply mutation to $offspring$ with probability $ProbMut$;
10.         If the fitness of $offspring$ is larger than $f^*$ then
11.             Replace $f^*$ by the fitness of $offspring$;
12.             Set $s^* \leftarrow offspring$;
13.         End-if
14.         Insert $offspring$ into the new generation population;
15.     End-for
16. End-while
End-GA+PR+LS
```

Fig. 2. Pseudo-code of the hybrid genetic algorithm with path-relinking crossover.

3 Progressive crossover by path-relinking

Path-relinking is an intensification strategy to explore paths connecting elite solutions obtained by metaheuristics. 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. To generate the paths, moves applied to the initial solution introduce attributes contained in the guiding solution. Therefore, path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions (i.e. the guiding elite solutions), by favoring these attributes in the selected moves. Extensions, improvements, and successful applications of path-relinking appeared in [1, 10, 25, 28, 29, 32]. Surveys reporting on advanced path-relinking strategies can be found in [26, 27].

Path-relinking was first applied in the context of a genetic algorithm to implement a progressive crossover operation by Ribeiro and Vianna [30]. In this work, we extend our original proposal and develop a better implementation. We follow a bidirectional (or back and forward) strategy: 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 operation: instead of producing only one offspring, it investigates many solutions that share the same 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 same parents.

Let s_1 and s_2 be, respectively, the initial and the guiding phylogenies. Let N_1 be the root of the phylogenetic tree s_1 and N_2 the root of s_2. By applying path-relinking to s_1 and s_2, the set of operational taxons in the left subtree of N_1 will be made equal to the set of operational taxons in the left subtree of N_2. Consequently, the operational taxons in the right subtree of N_1 will become the same appearing in the right subtree of N_2. This process is initiated at the roots of trees s_1 and s_2 and top-down propagated until their leaves are reached.

Figure 3 presents the pseudo-code of the crossover by path-relinking operation applied to solutions s_1 and s_2. The fitness $\bar{f}$ of the best solution found is initialized in line 1. The loop in lines 2-34 ensures that path-relinking is applied twice, once in each direction as defined in line 3. First, path-relinking builds a path from the initial solution $s = s_1$ to the guiding solution $s_{guiding} = s_2$. Next, path-relinking builds a path from the initial solution $s = s_2$ to the guiding solution $s_{guiding} = s_1$. Two stacks Q_1 and Q_2 containing the roots of s_1 and s_2, respectively, are initialized in line 4. The loop in lines 5-33 guarantees that all nodes of the initial solution s_0 will be analyzed and made in correspondence with those of the guiding solution $s_{guiding}$, beginning with their roots and continuing until their leaves are reached. Two nodes N_1 and N_2 are removed from stacks Q_1 and Q_2, respectively, in line 6. The left and right subtrees of nodes N_1 and N_2 are identified in lines 7 and 8, respectively.

Line 9 sets *comp*1 to the number of operational taxons common to the left subtrees of N_1 and N_2, plus the number of those common to their right subtrees. The value of *comp*1 represents the number of operational taxons that appear in the same subtrees in N_1 and N_2 (left-left or right-right) and, therefore, do not have to be moved from one subtree of N_1 to the other if path-relinking compares the left subtrees of N_1 and N_2 and the right subtrees of N_1 and N_2. Similarly, line 10 sets *comp*2 to the number of operational taxons common to the left subtree of N_1 and to the right subtree of N_2, plus the number of those common to the right subtree of N_1 and to the left subtree of N_2. The value of *comp*2 represents the number of operational taxons that appear in different subtrees in N_1 and N_2 (left-right or right-left) and, therefore, do not have to be moved from one subtree of N_1 to the other if path-relinking compares the left subtree of N_1 with the right of N_2 and the right subtree of N_1 with the left of N_2. The comparison of *comp*1 and *comp*2 and the possible interchange of the positions of the left and right subtrees of N_2 in line 11 is used to speedup path-relinking, by reducing the number of moves to be applied to the initial solution s until the guiding solution

```
Procedure Crossover($s_1, s_2$)
01. $\bar{f} \leftarrow 0$;
02. For $i = 1, 2$ do
03.     If $i = 1$ then $s \leftarrow s_1$; $s_{guiding} \leftarrow s_2$ else $s \leftarrow s_2$; $s_{guiding} \leftarrow s_1$;
04.     Initialize the stacks $Q_1$ and $Q_2$ with the roots of $s$ and $s_{guiding}$, respectively;
05.     While $Q_1 \neq \emptyset$ do
06.         Remove the top nodes $N_1$ from $Q_1$ and $N_2$ from $Q_2$;
07.         Let $LN_1$ and $RN_1$ be the left and right subtrees of $N_1$, respectively;
08.         Let $LN_2$ and $RN_2$ be the left and right subtrees of $N_2$, respectively;
09.         Set comp1 to the number of operational taxons common to $LN_1$ and $LN_2$
            plus the number of operational taxons common to $RN_1$ and $RN_2$;
10.         Set comp2 to the number of operational taxons common to $LN_1$ and $RN_2$
            plus the number of operational taxons common to $RN_1$ and $LN_2$;
11.         If $comp2 > comp1$ then interchange subtrees $LN_2$ and $RN_2$;
12.         Set $W$ as the set of operational taxons that appear either in $RN_1$ and
            $LN_2$, or in $LN_1$ and $RN_2$;
13.         While $W \neq \emptyset$ do
14.             Set $\Delta \leftarrow \infty$;
15.             For each operational taxon $t \in W$ do
16.                 Temporarily remove taxon $t$ from its current subtree in $N_1$;
17.                 Investigate all possible branches to where taxon $t$ can be moved
                    in the other subtree of $N_1$ and let $q$ denote that minimizing
                    the variation $\delta(t)$ in the parsimony of the current solution $s$;
18.                 If $\delta(t) < \Delta$ then set $t' \leftarrow t$, $q' \leftarrow q$, and $\Delta \leftarrow \delta(t)$;
19.                 Restore taxon $t$ to its original subtree in $N_1$ from where
                    it was temporarily removed;
20.             End-for;
21.             Obtain a new solution $s'$ by moving taxon $t'$ from its current subtree
                of $N_1$ to branch $q'$ of the other;
22.             Set $W \leftarrow W - \{t'\}$;
23.             If the fitness of $s'$ is greater than $\bar{f}$ then
24.                 Set $\bar{s} \leftarrow s'$;
25.                 Set $\bar{f}$ as the fitness of $s'$;
26.             End-if
27.             $s \leftarrow s'$;
28.         End-while
29.         If $N_1$ is not a leaf then
30.             Insert the left and right children of $N_1$ into stack $Q_1$;
31.             Insert the left and right children of $N_2$ into stack $Q_2$;
32.         End-if
33.     End-while
34. End-for
35. Return $\bar{s}$;
End-Crossover
```

Fig. 3. Crossover procedure using path-relinking.

$s_{guiding}$ is reached. If $comp1$ is larger than or equal to $comp2$, then subtrees that are already in the same position (left-left and right-right) in the current and in the guiding solutions will drive the path-relinking operation. Otherwise, it is more efficient to use subtrees in different positions (left-right and right-left) in the current and in the guiding solutions to drive path-relinking. This can be easily implemented by interchanging the positions of the left and right subtrees of N_2. Therefore, the algorithm always compares the left subtree of N_1 with that of N_2, and the right subtree of N_1 with that of N_2.

Set W computed in line 12 contains all operational taxons appearing either in LN_1 and RN_2, or in RN_1 and LN_2. These are badly positioned operational taxons that have to be moved from one subtree of N_1 to the other (left-right or right-left). The loop in lines 13-28 moves all operational taxons in W from one subtree of N_1 to the other. Variable Δ in line 14 is initialized to handle the computation in lines 15-20 of the taxon t' that leads to the smallest variation in the parsimony of the current solution s if it is moved from one subtree of N_1 to the other. Each badly positioned operational taxon t is temporarily removed from its current subtree of N_1 in line 16. In line 17, all possible positions (branches) where taxon t can be inserted in the other subtree of N_1 as a new leaf are investigated and the corresponding variation in the parsimony of the current solution s is evaluated. Let q be the position leading to the smallest variation $\delta(t)$. If moving the operational taxon t from its current subtree in N_1 to position q in the other subtree improves the best move already identified for this path-relinking iteration, than the latter is updated in line 18. Taxon t is restored in line 19 to its original subtree in N_1 from where it was temporarily removed, before a new badly positioned taxon be investigated. The best move is applied in line 20, with the operational taxon t' being removed from its current subtree of N_1 and inserted into position q' of the other subtree. Let s' be the new solution. The operational taxon t is removed from W since it is now in the same subtrees of N_1 and N_2 (left-left or right-right). In line 22, the fitness of the new solution s' is compared with that of the best solution $\bar{s}$ found in the previous path-relinking steps. In case of improvement, $\bar{s}$ and $\bar{f}$ are updated in lines 24 and 25, respectively. The current solution s is updated in line 27. Lines 29-32 implement the top-down propagation of the path-relinking operation. If N_1 is not a leaf (and, consequently, so is not N_2), then the left and right children of N_1 and N_2 are inserted into stacks Q_1 and Q_2, respectively, and a new step resumes.

Figure 4 illustrates the application of crossover by path-relinking to phylogenies s and $s_{guiding}$. The phylogenetic tree s in Figure 4-i will be progressively modified until it becomes equal to $s_{guiding}$. The nodes marked in s and $s_{guiding}$ are those corresponding to N_1 and N_2 at each step, respectively. The final situation is described in Figure 4-vi, where both phylogenies coincide.

Path-relinking was also applied by Zhang and Lai [38] as a crossover operator following the strategy proposed in [30] 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

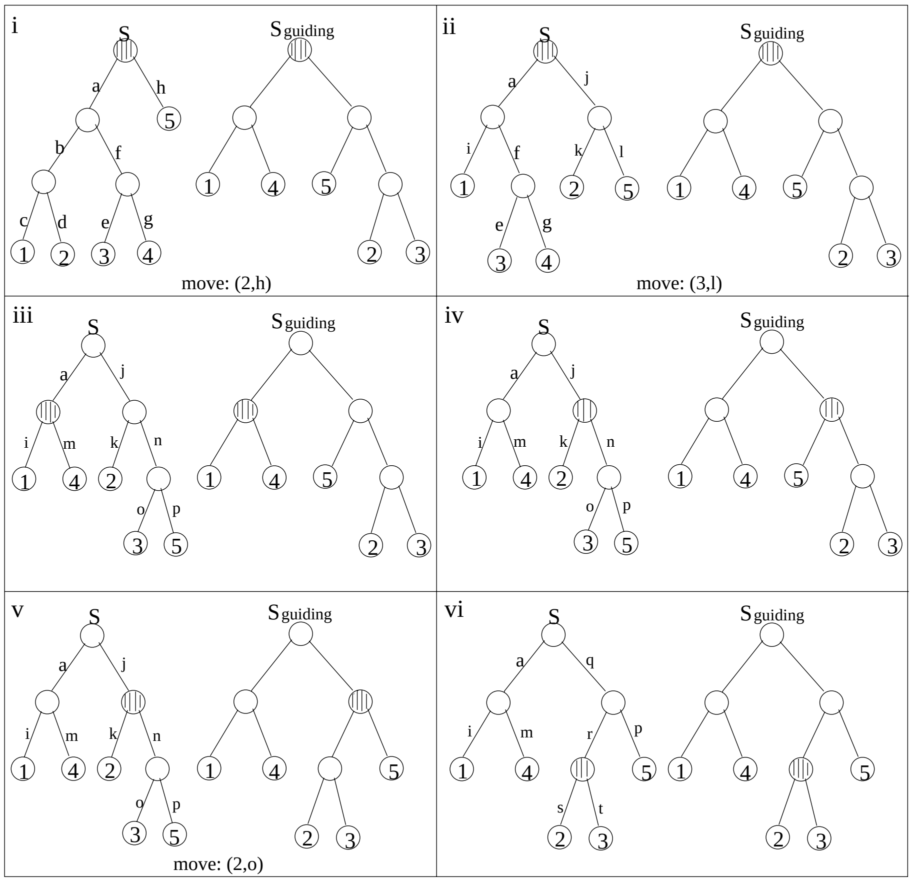

Fig. 4. Example of the application of path-relinking to two phylogenies s and $s_{guiding}$.

Ruiz [37] 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 in the best solution. The selected individuals are marked in order to not be selected again for the application of path-relinking. In another context, path-relinking was also hybridized with a genetic algorithm as a post-optimization procedure [24], in which the solutions obtained in the last generation of the genetic algorithm are progressively combined and refined.

4 Computational results

Algorithm `GA+PR+LS` was implemented in C using version 6.0 of the Microsoft Visual C++ compiler. All computational experiments were performed on a 2.0 GHz Pentium IV processor with 512 Mbytes of RAM.

Two sets of test instances have been used. The first corresponds to the eight benchmark instances [21, 23] also considered in [5, 31]. The second is formed by 20 randomly generated instances. The instance generator receives as parameters the number of taxons and the number of binary characteristics. The number of taxons ranges from 45 to 75 and the number of characteristics from 61 to 159.

Algorithm `GA+PR+LS` was compared with the `GRASP+VND` heuristic in [31] and with two versions of the original random-keys genetic algorithm presented in [30]: without local search (`RKGA`) and with local search (`RKGA+LS`). To further illustrate its behavior, we also compared algorithm `GA+PR+LS` with its simplified version `GA+PR`, in which the offsprings are never submitted to local search.

The comparison also involved another variant of the genetic algorithm, using a uniform crossover implemented as follows. First, the phylogenetic trees associated with the two parents $parent_1$ and $parent_2$ are traversed in preorder, generating two vectors formed by the operational taxons in the order in which their leaves are visited. These two vectors are submitted to a one-point crossover and two offspring vectors result. These offspring vectors may be infeasible, with missing and repeated taxons. Infeasibilities are repaired as follows. First, one of the duplicated taxons is randomly selected to be removed. Next, one of the missing taxons is also randomly selected to replace the removed duplicated taxon. These steps are repeated until each of the two offspring vectors contain exactly one copy of the n different operational taxons. Finally, each of the original two parents is traversed in preorder and the taxons in its leaves are progressively replaced by those appearing in one of the repaired offspring vectors. The other strategies and operators are the same used in `GA+PR+LS`. Once again, two versions are considered: without local search (`GAUni`) and with (`GAUni+LS`) local search.

In the first experiment, ten runs of each algorithm were performed on each instance, with the computation times limited to 1,000 seconds. Table 1 gives the number of taxons (n) and the number of binary characteristics (m) for each instance, followed by the average and best parsimony values obtained by each algorithm over ten runs. Results marked in bold are the best obtained for each instance. Algorithm `GA+PR+LS` obtained the best average values for seven benchmark instances. It also obtained the best average values for 17 out of the 20 randomly generated instances. Algorithm `GA+PR+LS` found the best solutions for all benchmark instances and for 14 out of the 20 randomly generated instances.

Time-to-target plots (tttplots) display on the ordinate axis the probability that an algorithm will find a solution at least as good as a given target value within a given running time, shown on the abscissa axis. They were used by Feo et al. [9] and have been advocated by Hoos and Stützle [19, 20] as a way to characterize the running times of stochastic algorithms for combinatorial optimization. Aiex et al. [2] suggested and largely explored the use of tttplots to evaluate and compare different randomized algorithms running on the same instance. Their use has been growing ever since and they have been extensively applied in computational studies of sequential and parallel randomized algorithms [25, 26, 28]. The foundations of the construction of time-to-target plots, together with their interpretation and applications, were surveyed by Aiex et al. [3].

In the second experiment, we used tttplots to compare the hybrid algorithm `GA+PR+LS` with its simpler version `GA+PR` and with the other algorithms involved in the first experiment. Since similar results have been observed for all instances, they are summarized by those obtained for instance SCHU. We performed 250 independent runs of each algorithm. Each run finishes when a solution value less than or equal to a given target is found or after three hours of computations if the target was not attained. The target value was set to 760, which is the value of the best previously known solution for this instance. The empirical probability distributions of the time-to-target random variables are plotted in Figure 5 for each algorithm. These plots show that algorithm `GA+PR+LS` systematically finds better solutions than the others in smaller computation times. Furthermore, algorithm `GA+PR+LS` is also more robust, presenting a much smaller variability in the computation times over different runs.

The final experiment makes use of the results obtained for the random instance TST17 to assess the evolution of the solutions found by each algorithm along one hour (3,600 seconds) of computations. For each algorithm `RKGA`, `GA+PR`, and `GAUni`, Figure 6 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 it obtained are confined in a smaller interval ranging from 2500 to 2620. The solution values obtained by the two other algorithms present a larger variability. The solutions progressively 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 operation.

Similarly, for each algorithm `RKGA+LS`, `GA+PR+LS`, and `GAUni+LS` using local search, Figure 7 presents the solution value at the end of each generation for each of the 100 individuals in the population. These algorithms found better solutions than their counterparts without local search. The solutions obtained by each of them are clearly separated in two clouds: one formed by offsprings to

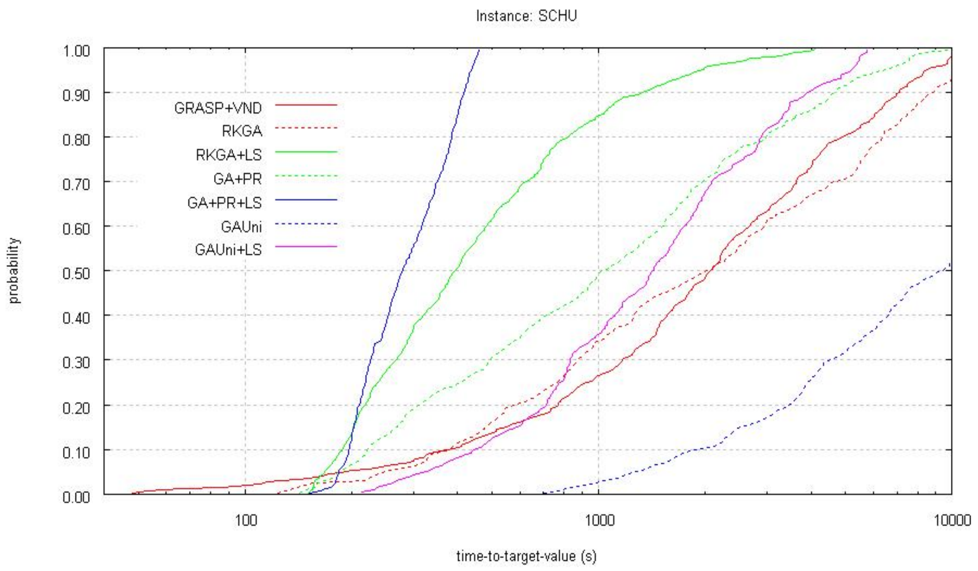

Fig. 5. Empirical probability distributions of time-to-target (instance SCHU).

which local search was not applied (higher values), and the other corresponding to the solutions obtained by local search (smaller values).

Figure 8 superimposes the results report for algorithms `GA+PR+LS` and `RKGA+LS` in the previous figure. The solution values obtained by algorithm `GA+PR+LS` are plotted in green, while those obtained by `RKGA+LS` are given in red.

5 Conclusion

In this work, we compared different genetic algorithm implementations and a GRASP with VND heuristic for the phylogeny problem. In particular, we assessed the results obtained by a hybridization of a genetic algorithm with a path-relinking scheme to implement the crossover operator.

The resulting hybrid genetic algorithm `GA+PR+LS` also makes use of a local search procedure to improve some offsprings resulting from crossover. Computational results have shown that the hybrid algorithm `GA+PR+LS` systematically found better solutions in smaller computation times. It matched the best solutions previously reported in the literature for all benchmark instances. It also found better solutions than the other algorithms for most random test instances. Furthermore, it is more robust and presents much smaller variability in the computation times.

The numerical results clearly illustrate the contribution of the path-relinking scheme in the implementation of an effective crossover operator. Path-relinking was also very appropriate for handling the complex solution coding data structure associated with phylogenetic trees.

Acknowledgements

The authors are grateful to one anonymous referee who incited and encouraged them to prepare an improved and extended version of the original manuscript.

References

1. R.M. Aiex, P.M. Pardalos, M.G.C. Resende, and G. Toraldo. GRASP with path-relinking for three-index assignment. *INFORMS Journal on Computing*, 17:224–247, 2005.
2. 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.
3. R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. TTTPLOTS: A perl program to create time-to-target plots. *Optimization Letters*, 1:355–366, 2007.
4. A.A. Andreatta, S.E.R. Carvalho, and C.C. Ribeiro. A framework for local search heuristics for combinatorial optimization problems. In S. Voss and D. Woodruff, editors, *Optimization Software Class Libraries*, pages 59–79. Kluwer, 2002.
5. A.A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002.
6. F.J. Ayala. The myth of Eve: Molecular biology and human origins. *Science*, 270:1930–1939, 1995.
7. C. Cotta. Scatter search with path relinking for phylogenetic inference. *European Journal of Operational Research*, 169:520–532, 2006.
8. A. Edwards and L. Cavalli-Sforza. Reconstruction of evolutionary trees. *Phenetic and Phylogenetic Classification*, 6:67–76, 1964.
9. 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.
10. P. Festa, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. GRASP with path-relinking for the weighted MAXSAT problem. *ACM Journal of Experimental Algorithmics*, 11:1–16, 2006.
11. P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer, 2002.
12. 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.
13. 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, 2000.
14. F. Glover and M. Laguna. *Tabu Search*. Kluwer, 1997.
15. F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

16. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, Reading, 1989.
17. W. Hennig. *Phylogenetic systematics.* University of Illinois Press, Urbana, 1966.
18. J. Holland. *Adaptation in natural and artificial systems.* MIT Press, 1992.
19. H.H. Hoos and T. Stützle. Evaluation Las Vegas algorithms - Pitfalls and remedies. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 238–245, 1998.
20. H.H. Hoos and T. Stützle. On the empirical evaluation of Las Vegas algorithms â-Position paper. Technical report, Computer Science Department, University of British Columbia, 1998.
21. M. Luckow and R.A. Pimentel. An empirical comparison of numerical Wagner computer programs. *Cladistics*, 1:47–66, 1985.
22. D. Penny, L.R. Foulds, and M.D. Hendy. Testing the theory of evolution by comparing phylogenetic trees constructed from five different protein sequences. *Nature*, 247:197–200, 1982.
23. N.I. Platnick. An empirical comparison of microcomputer parsimony programs. *Cladistics*, 3:121–144, 1987.
24. 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.
25. 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, 2003.
26. 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.
27. M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures: Advances and applications. In M. Gendreau and J.Y. Potvin, editors, *Handbook of Metaheuristics.* Kluwer, 2nd edition, to appear.
28. C.C. Ribeiro and I. Rosseti. Efficient parallel cooperative implementations of GRASP heuristics. *Parallel Computing*, 33:21–35, 2007.
29. 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.
30. 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 Bioinformática*, pages 97–102, Macaé, 2003.
31. C.C. Ribeiro and D.S. Vianna. A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. *International Transactions in Operational Research*, 12:325–338, 2005.
32. M. Scaparra and R. Church. A GRASP and path relinking heuristic for rural road network development. *Journal of Heuristics*, 11:89–108, 2005.
33. E. Sober. Parsimony, likelihood and the principle of the common cause. *Philosophy of Science*, 54:465–469, 1987.
34. D.L. Swofford. *Wagner procedure program.* Illinois Natural History Survey, Champaign, 1982.
35. D.L. Swofford and G. Olsen. Phylogeny reconstruction. In D. M. Hillis and C. Moritz, editors, *Molecular Systematics*, pages 411–501. Sinauer, 1990.
36. D.L. Swofford, G. Olsen, P.J. Waddell, and D.M. Hillis. Phylogeny inference. In D. M. Hillis, C. Moritz, and B. K. Mable, editors, *Molecular Systematics*, pages 407–514. Sinauer, 2nd edition, 1996.

37. E. Vallada and R. Ruiz. New genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem, 2006. Universidad Politécnica de Valencia.
38. 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.

			Average solution values							Best solution values						
Instance	n	m	`GRASP+VND`	`RKGA`	`RKGA+LS`	`GA+PR`	`GA+PR+LS`	`GAUni`	`GAUni+LS`	`GRASP+VND`	`RKGA`	`RKGA+BL`	`GA+PR`	`GA+PR+LS`	`GAUni`	`GAUni+LS`
GRIS	47	93	**172.0**	**172.0**	**172.0**	**172.0**	**172.0**	**172.0**	**172.0**	**172**	**172**	**172**	**172**	**172**	**172**	**172**
ANGI	49	59	**216.0**	**216.0**	**216.0**	216.4	**216.0**	**216.0**	**216.0**	**216**	**216**	**216**	**216**	**216**	**216**	**216**
TENU	56	179	**682.0**	**682.0**	**682.0**	**682.0**	**682.0**	**682.0**	**682.0**	**682**	**682**	**682**	**682**	**682**	**682**	**682**
ETHE	58	86	**372.0**	**372.0**	372.4	372.6	**372.0**	**372.0**	**372.0**	**372**	**372**	**372**	**372**	**372**	**372**	**372**
ROPA	75	82	326.0	326.2	**325,6**	326.6	325.8	**325.6**	**325.6**	326	326	**325**	326	**325**	**325**	**325**
GOLO	77	97	498.2	498.2	497.2	496.6	**496.2**	497.0	497.6	**496**	**496**	**496**	**496**	**496**	497	497
SCHU	113	146	761.0	763.2	759.8	760.6	**759.4**	761.0	761.0	**759**	**759**	**759**	760	**759**	760	760
CARP	117	110	552.4	552.4	549.2	549.6	**548.6**	550.6	550.2	550	549	**548**	549	**548**	550	549
TST01	45	61	550.8	550.0	549.0	549.8	**548.6**	551.0	550.8	549	549	**547**	548	**547**	549	549
TST02	47	151	1367.2	1369.2	1365.0	1366.8	**1362.6**	1366.2	1366.0	**1361**	1367	1362	1363	**1361**	1364	1364
TST03	49	111	843.8	845.6	**838.8**	841.4	841.0	844.2	842.0	843	843	**836**	837	837	842	841
TST04	50	97	599.2	596.6	596.8	596.8	**592.6**	600.0	596.0	598	593	593	592	**590**	598	592
TST05	52	75	796.8	797.0	795.4	796.2	**793.4**	798.8	798.8	793	794	793	**792**	**792**	798	798
TST06	54	65	606.6	605.6	**602.6**	606.2	603.8	607.0	605.6	605	603	**600**	603	603	605	604
TST07	56	143	1288.6	1289.4	1285.0	1284.4	**1279.4**	1288.2	1285.2	1283	1286	1282	1278	**1274**	1286	1281
TST08	57	119	873.0	871.4	868.2	872.0	**866.4**	873.4	871.2	868	869	867	869	**862**	868	871
TST09	59	93	1159.2	1155.0	1153.6	1157.2	**1151.4**	1158.8	1156.6	1156	1153	**1147**	1151	1150	1157	1155
TST10	60	71	732.0	735.2	727.6	727.8	**725.8**	732.2	731.0	730	732	726	725	**722**	731	726
TST11	62	63	554.8	554.6	**548.2**	554.0	550.0	555.2	553.2	554	550	**546**	551	547	555	551
TST12	64	147	1242.0	1238.4	1230.2	1237.8	**1228.6**	1239.0	1236.6	1237	1233	**1224**	1231	1225	1234	1231
TST13	65	113	1532.4	1537.0	1531.4	1533.4	**1526.8**	1539.6	1533.2	1529	1536	1528	1528	**1524**	1535	1529
TST14	67	99	1182.0	1183.0	1177.2	1178.8	**1173.0**	1185.2	1180.4	1180	1179	1176	1173	**1171**	1181	1175
TST15	69	77	774.8	776.8	767.4	766.4	**760.8**	774.4	770.8	769	772	762	762	**758**	772	768
TST16	70	69	550.6	545.6	545.0	547.2	**542.0**	552.0	549.0	547	545	540	543	**537**	550	547
TST17	71	159	2479.0	2476.4	2472.2	2478.8	**2471.2**	2483.8	2479.8	2475	2470	**2468**	2473	2469	2483	2476
TST18	73	117	1554.6	1566.0	1546.2	1552.8	**1536.6**	1558.0	1555.0	1548	1548	1533	1546	**1531**	1556	1552
TST19	74	95	1041.0	1039.8	1035.0	1036.6	**1027.4**	1042.0	1036.8	1035	1034	1034	1033	**1024**	1040	1032
TST20	75	79	685.4	680.6	678.4	683.4	**673.8**	685.6	682.0	682	679	674	681	**671**	682	680

Table 1. Comparative results (ten runs of each algorithm).

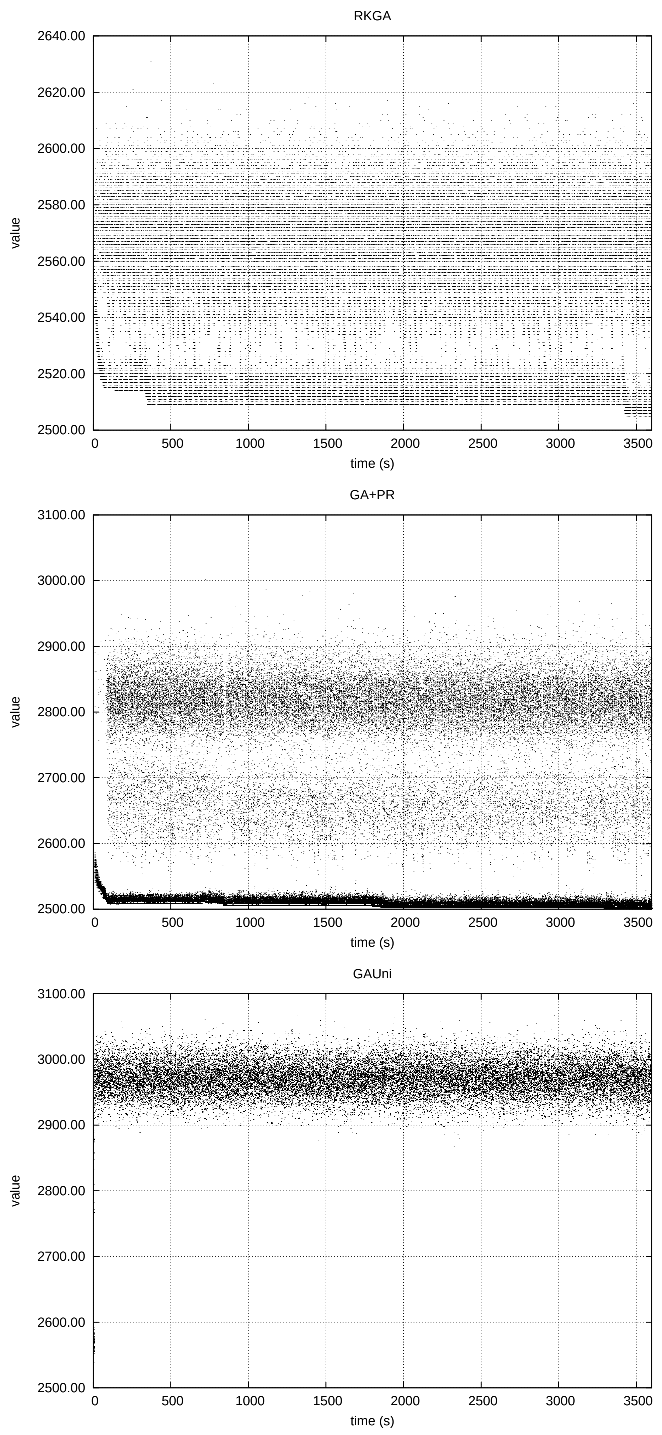

Fig. 6. Solutions obtained for instance TST17 along 3,600 seconds by the algorithms without local search.

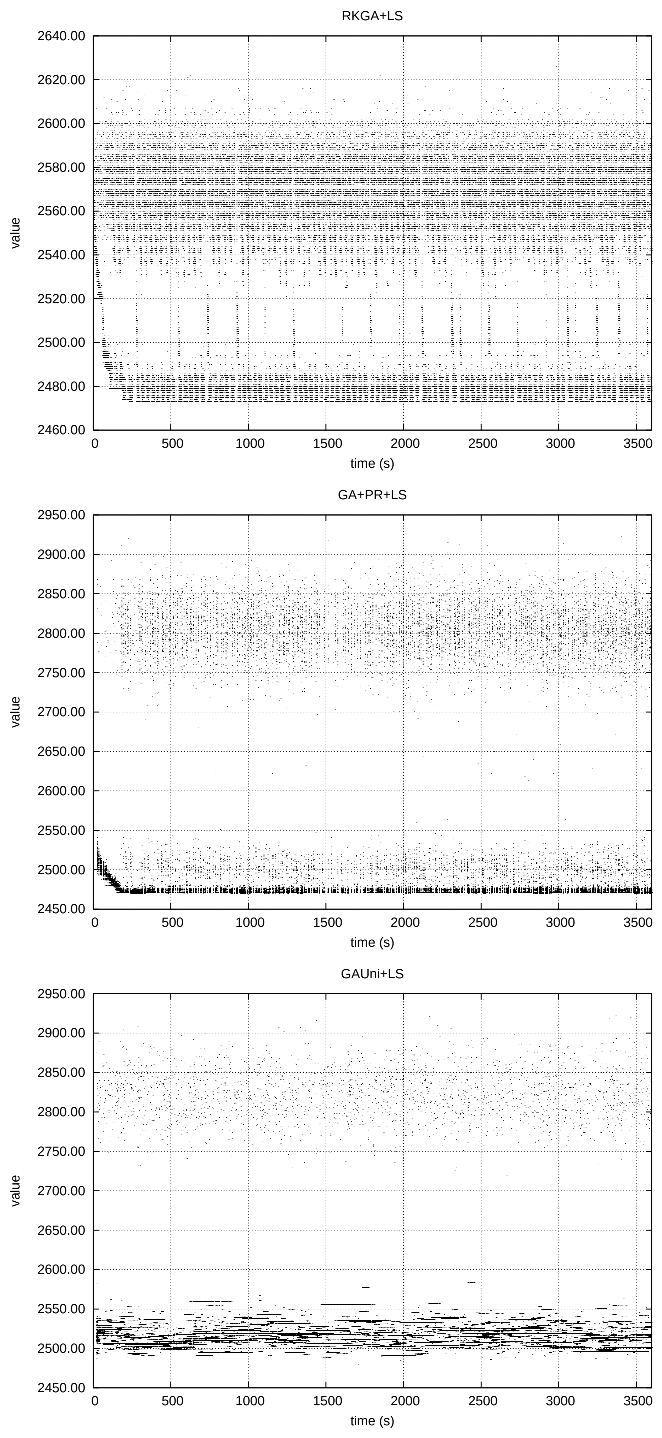

Fig. 7. Solutions obtained for instance TST17 along 3,600 seconds by the algorithms with local search.

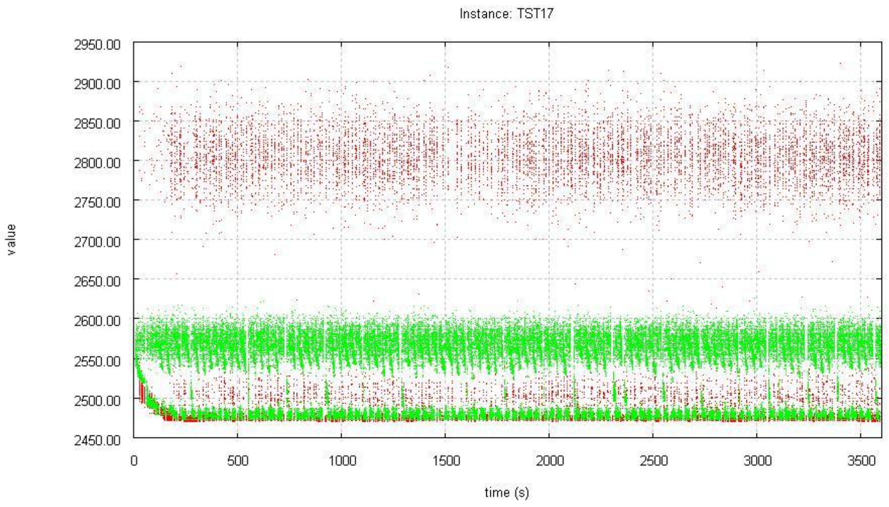

Fig. 8. Solution values obtained for instance TST17 by algorithms `GA+PR+LS` (red) and `RKGA+LS` (green) along 3,600 seconds.