Scatter search with path relinking for phylogenetic inference

Carlos Cotta *

Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga, ETSI Informática (3.2.49), Campus de Teatinos, 29071-Málaga, Spain

Received 30 June 2003; accepted 10 May 2004
Available online 12 October 2004

Abstract

We propose the use of scatter search with path relinking for the inference of phylogenetic trees. Solutions are here represented as trees whose leaves span the set of species under study. These trees are evaluated using a minimum weight criterion under the ultrametric model. The main features of this approach are the utilization of a crossover-based schema for diversification generation, the use of path relinking for solution combination, and the utilization of an improvement method based on internal rotations of subtrees. The resulting algorithm is compared to other approaches such as evolutionary and memetic algorithms, using real data as benchmark. Scatter search provides better results for these instances. © 2004 Elsevier B.V. All rights reserved.

Keywords: Evolutionary computation; Phylogenetic inference; Ultrametric trees

1. Introduction

Molecular biology and genomics are living in a period of effervescence. The many initiatives in the life sciences already planned and currently in execution are producing an unprecedented flood of data [34], and as a result many of the challenges in biology are increasingly becoming challenges in mathematics [22], and fundamentally in computing. In effect, the task of dealing with large-scale combinatorial problems arising in bioinformatics is undoubtedly one of the greatest challenges to be addressed by computer science researchers [26,42]. New techniques, and new insights for algorithm design are needed. This challenge is bound to exert a strong impact on computer science since—among other factors—these new methods should be easily adapted to high-performance, distributed, computing systems.

Among the challenging areas in molecular biology one can cite the construction of evolutionary trees, a problem of great importance for tasks such as multiple sequence alignment [17], protein structure prediction [36] or molecular epidemiological studies of viruses [33] among others. The construction of an evolutionary tree amounts to the
production of a hierarchy showing the degree of closeness among a set of organisms. The associated difficulty of this task is inherent to the computational complexity and the sheer amount of data to process. Computational tools are clearly required to cope with these data. In this sense, the use of exact techniques can be considered generally inappropriate here for two reasons: firstly, the intrinsic complexity of the problem (e.g., NP-hardness has been shown for phylogenetic inference under several models [6–8,12,43]); secondly, while the utilization of a quality measure for evaluating hierarchies implies the definition of a optimization problem, its global optimum has not the same significance as in other classical problems (the existence of some uncertainty in the underlying empirical data may make high-quality suboptimal solutions be equally valid). Thus, the use of heuristic techniques in this domain seems much more adequate. These can range from simple constructive heuristics (e.g., greedy agglomerative techniques such as UPGMA [40]) to complex metaheuristics (e.g., evolutionary algorithms—EAs—[4]). In this sense, the use of scatter search [27] is considered in this work.

Scatter search (SS) is a powerful metaheuristic based on populational search whose foundations can be traced back to the 70s in the context of combining decision rules and problem constraints. Unlike other populational approaches such as genetic algorithms, SS relies more on deterministic strategies rather than on randomization. This important methodological difference (which in turn motivates some other particularities of the approach, as it will be discussed later) notwithstanding, SS shares some crucial elements with evolutionary and memetic algorithms (MAs) [32], such as the use of combination procedures and local-improvement strategies. In this work, SS with path relinking [14–16] is used for inferring phylogenetic trees from genomic data. To this end, some basic concepts on phylogenetic analysis will be firstly provided in next section.

2. Background on phylogenetic inference

The inference of phylogenetic trees is one of the most important and challenging tasks in Systematic Biology. Such trees are used to represent the evolutionary history of a collection of $n$ organisms (or taxa) from their molecular sequence data, or from other form of dissimilarity information. The Phylogeny Problem can then be formulated as finding the phylogenetic tree that best—under a certain optimality criterion—represents the evolutionary history of a collection of taxa. For this purpose, it is clearly necessary to define an optimization criterion. Essentially, optimization criteria for assessing the goodness of a phylogenetic tree $T$ can fall within two major categories, sequence-based and distance-based [23].

In sequence-based approaches, each node of $T$ is assigned a sequence. Such a sequence is known for the leaves (i.e., the taxa being classified), and can be inferred via pairwise alignments for internal nodes. Subsequently, the tree is evaluated using a criterion that in most situations is either maximum likelihood (ML) or maximum parsimony (MP). ML criteria are based on the assumption of a stochastic model of evolution, e.g., the Jukes-Cantor model [21], the Kimura 2-parameter model [24], etc. (see also [10]). Such a model is used in order to assess the likelihood that the current tree generated the observed data. The optimal tree would be the one that maximizes this likelihood. On the other hand, an MP criterion specifies that the tree requiring the fewest number of evolutionary changes to explain the data is preferred.

As to distance-based approaches, they are based on transforming the available sequence data into an $n \times n$ matrix $M$. This matrix is the only information used in the subsequent inference process. More precisely, edges in $T$ are assigned a weight. The basic idea here is that $M_{ij}$ represents the evolutionary distance or dissimilarity between taxa $i$ and $j$. We thus have an observed distance matrix $M$ (the input data), and an inferred distance matrix $\hat{M}$ (obtained by making $\hat{M}_{ij} =$ distance from $i$ to $j$ in $T$, i.e., the sum of the weights of all edges in the path from $i$ to $j$ in $T$). The quality of the tree can now be quantified in a variety of ways. On one hand, it is possible to consider some “distance” measure between $M$ and $\hat{M}$; usual examples are:
• the STRESS measure [38]: \( \text{STRESS}(M, \hat{M}) = \frac{1}{\sum_{i,j} |M_{ij} - \hat{M}_{ij}|} \), i.e., the normalized sum of absolute differences,

• the \( L_2 \) metric: \( L_2(M, \hat{M}) = \sum_{i,j} (M_{ij} - \hat{M}_{ij})^2 \), i.e., a least-squares approximation,

• the \( L_\infty \) metric: \( L_\infty(M, \hat{M}) = \max_{i,j} |M_{ij} - \hat{M}_{ij}| \), i.e., the maximum absolute difference between observed and inferred data.

On the other hand, quality can be directly measured from \( T \). This is typically the case when edge-weighting has been constrained so as to have \( M_{ij} \geq \hat{M}_{ij} \), i.e., to have inferred distances greater than observed distances. This constraint is based on the fact that the observed distance between two taxa will be always a lower bound of the real evolutionary distance between them (in essence, this is due to the existence of a set of phenomena—reversal, parallelism, and convergence—that make taxa appear more related than they really are; these phenomena are collectively termed homoplasy); see [29] for details. In this situation, minimizing the total weight of \( T \) (i.e., the sum of all edge-weights) is usually the criterion. This is precisely the situation considered in this work.

Notice that by taking \( M_{ij} \) as the minimum number of evolutionary events needed to transform \( i \) in \( j \), this last approach resembles MP. Actually, distance-based methods can be generally considered as an intermediate strategy between ML and MP, exhibiting good performance in practice as well [20]. For these reasons, we have focused in distance-based approaches in this work. To be precise, we have forced \( \hat{M} \) to be ultrametric [2]. In this case, it holds that

\[ \hat{M}_{ij} \leq \max\{\hat{M}_{ik}, \hat{M}_{jk}\}, \quad 1 \leq i, j, k \leq n. \tag{1} \]

If \( \hat{M} \) is ultrametric, then the distance in \( T \) between any internal node \( h \) and any leaf \( o \) descendant of \( h \) is the same. Very popular when the molecular-clock hypothesis [29] was in vogue, this condition provides a very good approximation to the optimal solution under more relaxed assumptions such as mere additivity (\( M \) is additive if for any \( i, j, k, l \), the maximum of \( M_{ij} + M_{kl}, M_{ik} + \hat{M}_{jl}, M_{il} + \hat{M}_{jk} \) is not unique). It is also easy to compute: for a given tree \( T \), and observed matrix \( M \), edge weights can be determined in \( O(n^2) \) time using the algorithm presented in [43]. This is known as the min ultrametric tree with a given topology problem. Let us represent trees using a LISP-like notation, i.e., trees are represented as \((r, L, R)\), where \( r \) is the root, \( L \) is the left subtree, and \( R \) is the right subtree. A leaf \( l \) is represented as \((l)\). Also, let \( \mathcal{L}(T) \) be the set of leaves of tree \( T \). Now, let the height of each node in the tree be computed as follows:

\[
\text{height}((l)) = 0, \tag{2}
\]

\[
\text{height}((r, L, R)) = \max \left( \text{height}(L), \text{height}(R), \frac{D(L, R)}{2} \right), \tag{3}
\]

where \( D(L, R) = \max\{|M_{ij}| i \in \mathcal{L}(L), j \in \mathcal{L}(R)\} \). Subsequently, the weight of an edge connecting two nodes \( i \) and \( j \) is defined as \( |\text{height}(i) - \text{height}(j)| \). In contrast to this easy computation, notice that finding optimal edge weights (i.e., those whose sum is minimal as mentioned before) for an additive tree requires solving a linear program with \( 2n - 2 \) variables (the number of internal edges), and \( n(n - 1)/2 \) constraints, one for each pair of taxa \((i, j)\), corresponding to the fact that the sum of weights for edges in the path connecting them in the tree—i.e., the inferred distance \( \hat{M}_{ij} \)—be greater or equal than the observed distance \( M_{ij} \) (this linear program would turn into a quadratic program if a least-squares approximation were sought [30]).

Notice that the number of possible phylogenetic trees for a given set of \( n \) taxa is huge: there are \((2n - 3)!! \) rooted trees [20], where \( k!! \) is the double factorial of \( k \) (i.e., the difference between successive factors is 2 rather than 1 as in the standard factorial). For example, there exist \( 8.2 \times 10^{21} \) possible trees for 20 taxa. This clearly illustrates the impossibility of applying exhaustive search to this problem. Furthermore, finding provably good solutions constitutes a very hard combinatorial optimization problem for most optimality criteria, as anticipated in the previous section. Exact techniques such as branch-and-bound can be used, but they are computationally unaffordable for even moderate-size (say, 30–40 taxa) problem in-
stances. Hence, the use of heuristic techniques seems appropriate.

Most typical heuristics for phylogenetic inference are variants of the single-link [39], complete-link [25], and average-link [40] algorithms. These are agglomerative clustering algorithms whose functioning matches the generic template shown in Fig. 1.

As it can be seen, these algorithms proceed by iteratively joining in a tree the two closest clusters, until just one group remains. They differ in the way intercluster distance is defined. To be precise, they consider the following distance measures:

- **Single-link:**
  \[ \text{dist}(T, T') = \min \{ M_{ij} | i \in \mathcal{P}(T), j \in \mathcal{P}(T') \} ; \]

- **Complete-link:**
  \[ \text{dist}(T, T') = \max \{ M_{ij} | i \in \mathcal{P}(T), j \in \mathcal{P}(T') \} ; \]

- **Average-link:**
  \[ \text{dist}(T, T') = \frac{1}{|\mathcal{P}(T)||\mathcal{P}(T')}| \sum_{i \in \mathcal{P}(T), j \in \mathcal{P}(T')} M_{ij} . \]

Notice that the complete-link can be regarded as a greedy approach for the quality criterion we have chosen (the height of an internal node is always one half of the largest distance between any of its leaves—recall Eq. (3)—and complete-link makes local decisions trying this minimize this quantity).

3. Scatter search for phylogenetic inference

As mentioned in Section 1, SS is a populational metaheuristic. This means that a pool of solutions for the problem at hand is maintained, and used for generating new tentative solutions. Like other populational metaheuristics such as MAs, this generation of new solutions is accomplished by combining sets of existing solutions, and by using local improvement strategies. SS has several distinctive features though. The main one can be found in the methodological principles of the algorithm: randomization is given a much more secondary role than in other populational metaheuristics, since deterministic rules are used. This can be appreciated in the sketch of SS presented in Fig. 2, e.g., by observing the absence of a mutation operation. More precisely, a closer inspection to this algorithmic sketch reveals the existence of the following components in SS:

- A **diversification generation method** for generating a collection of raw solutions, possibly using some initial solution as “seed”.
- An **improvement method** for enhancing the quality of raw solutions.
- A **reference set update method** for building the reference set from the initial set of solutions generated, and for maintaining it by incorporating some solutions produced in subsequent steps.
- A **subset generation method** for selecting solutions from the reference set, and arranging them in small groups (pairs, triplets, or larger groups) for undergoing combination.

---

**Agglomerative Clustering Algorithm**

\[
\text{INPUT: a distance matrix M} \\
\text{OUTPUT: a tree T} \\
(1) \text{ for } i = 1 : n \text{ do } T_i \leftarrow (i) \\
(2) \text{ Let num\_clusters } \leftarrow n \\
(3) \text{ while num\_clusters } > 1 \text{ do} \\
\hspace{1em} (a) \text{ Select } T_i \text{ and } T_j (1 \leq i < j \leq \text{num\_clusters}) \text{ for which } \text{dist}(T_i, T_j) \text{ is minimal.} \\
\hspace{1em} (b) \text{ Let } T_i \leftarrow (h, T_i, T_j) \\
\hspace{1em} (c) \text{ Let } T_j \leftarrow T_{\text{num\_clusters}} \\
\hspace{1em} (d) \text{ Let num\_clusters } \leftarrow \text{num\_clusters} - 1 \\
(4) \text{ output } T_1
\]

Fig. 1. Pseudocode of an agglomerative clustering algorithm. In step 3b, h represents an undistinguishable internal node of the tree.
A solution combination method for creating new raw solutions by combining the information contained in a certain group of solutions.

A restart reference set method for refreshing the reference set once it has been found to be stagnated. This can be typically done by using the diversification generation method plus the improvement method mentioned above, but other strategies might be considered as well.

The specification of a particular SS algorithm is completed once the items above are detailed. The next subsections will be devoted to this purpose. Obviously, it is previously required to define how solutions are represented. In this application, solutions are trees expressing a particular evolutionary history for an element set $\mathcal{E}$. An arbitrary enumeration of elements in $\mathcal{E}$ will be assumed. Subsequently, each element in this set will be referred to by means of its index $i \in \{0, \ldots, n-1\}$ in this enumeration. These elements constitute the leaves of the tree. Since trees are binary, this means that there are $n-1$ internal nodes. These can be labeled with any negative value. Finally, trees are internally stored in linear strings of $2n-1$ positions by means of their pre-order traversal.

### 3.1. Diversification generation method

The diversification generation method serves two purposes in the SS algorithm considered: it is used for generating the initial population from which the reference set will be extracted at the beginning of the run, and it is utilized for refres-
hing the reference set whenever a restart is needed. In both cases it is used in combination with the improvement method that will be described in a further subsection.

When utilized for generating the initial population, a certain solution is used as seed. To be precise, the seed is the solution produced by the complete-link algorithm described in Section 2. This solution is inserted in the initial population, and used for generating the remaining $P_{\text{Size}} - 1$ configurations by means of the Prune–Delete–Graft (PDG) tree crossover operator (cf. [4,31]). PDG is a three-step procedure for recombining two trees $T_1$ and $T_2$:

1. Prune a subtree $T'$ from $T_1$.
2. Delete from $T_2$ all leaves occurring in $T'$.
3. Graft $T'$ at a randomly selected point of $T_2$.

Thus, PDG transfers a subtree from one tree to another, taking care of respecting the constraint that all elements in $\mathcal{E}$ are spanned. Let $T$ be the tree provided by the complete-link algorithm. The diversification generation method uses PDG for recombining $T$ with itself, i.e., trees are generated by moving a subtree of $T$ to a different position within itself. This method has two advantages. On one hand, the initial population is ensured to have good-quality solutions (the solution provided by the complete-link is known to be very good in general, and the remaining solutions are not completely random but perturbations of the former). On the other hand, it results in solutions that, while diverse, are not located in opposite regions of the search space. This is important since path relinking is used as combination method as it will be described in Section 3.5. Having very distant solutions in the reference set would notably increase the cost of the relinking procedure.

When used for restarting, the same mechanism is used, with the sole exception that this time pairs of trees from the reference set are randomly selected and submitted to PDG. This produces new solutions that introduce diversity in the reference set, but does not do it from scratch, since valuable existing information is also used. The best solution currently in the reference set is preserved. Once the improvement method has been applied to these new solutions, they are submitted to the reference set update method for obtaining the new reference set.

### 3.2. Improvement method

The improvement method is responsible for enhancing raw solutions produced by the diversification generation method, or by the solution combination method. In general, this is achieved by applying small changes to a solution, keeping them if they produce a quality increase, or discarding them otherwise.

Different strategies would have been possible given the tree representation used in this problem. The reader may check [1] for a description of several neighborhoods for trees (unrooted trees actually, although the ideas presented therein can be readily adapted to rooted trees). In this case, the improvement method chosen is based on performing rotations within the tree. Considering the quality measure used (minimizing the weight of an ultrametric tree), this strategy offers a very good tradeoff between its computational cost, and the attainable quality improvement.

Four symmetric operations are used within the improvement method. The first one, $\text{ROT}_{1}^{R}$, is defined as

\[
\text{ROT}_{1}^{R}[(h, (h', T_{\text{LL}}, T_{\text{LR}}), T_{R})] = (h, T_{\text{LL}}, (h', T_{\text{LR}}, T_{R})),
\]

(4)

where $h$ and $h'$ are internal nodes. This operation moves $T_{LR}$, the right subtree of the left subtree of $h$, to the right so it becomes the left subtree of the right subtree of $h$. In order to test whether a change is satisfactory, it is not necessary to evaluate the whole tree: it suffices to check whether the following inequality holds:

\[
\max\{M_{ij} | i, j \in \mathcal{L}(T_{\text{LL}}) \cup \mathcal{L}(T_{LR})\} > \max\{M_{ij} | i, j \in \mathcal{L}(T_{LR}) \cup \mathcal{L}(T_{R})\}.
\]

(5)

A $\text{ROT}_{2}^{R}$ operation would have performed the same movement on $T_{\text{LL}}$ rather than on $T_{LR}$. Analogously, $\text{ROT}_{1}^{L}$ and $\text{ROT}_{2}^{L}$ are mirror-inverted versions of the previous operations. For each interior
node of the tree, it is first checked whether a ROTR movement is possible, and if so, whether ROTR or ROT2R produce an improvement. If this were the case, the change would be retained, and the improvement method would stop. Otherwise, ROTL movements would be analogously attempted. If no improvement is possible either, the procedure is recursively applied to the left and right subtrees of h.

3.3. Reference set update method

The reference set update method must produce the reference set for the next step by using the current reference set and the newly produced offspring (or by using the initial population generated by diversification at the beginning of the run or after a restart). Again, several strategies are possible here. A first criterion for having a new solution gaining membership of the reference set is quality: whenever a new solution is better than the worst existing solution in the reference set, the latter is replaced by the former. This would be analogous to the plus replacement strategy commonly used for instance in evolution strategies (i.e., the best |RefSet| solutions in RefSet ∪ Offspring are kept).

Another typical criterion for insertion in the reference set is diversity. The reference set would be then structured in two tiers: the first one corresponds to quality and is managed in the same way as described above; the second one corresponds to diversity, and membership to it is gained when a solution increases the distance of tier-2 to tier-1 (i.e., the minimum distance of the new solution to any solution in tier-1 is greater than the minimum distance of an existing solution in tier-2 to any solution in tier-1). Of course, this implies having to define a distance measure among solutions. Some authors have proposed considering even three tiers, having tier-3 for containing “good generators” [27]. For simplicity, these multi-tier strategies have not been considered in this work, being quality the criterion used. This criterion is complemented with a test-for-duplicates, so that a solution is not inserted in the reference set if it is identical to an existing solution.

A variant of this update method has been also considered: rather than generating all descendants and then deciding which of them will be included in the reference set, descendants can be generated one-at-a-time, and inserted in the reference set if they qualify for it. This is called a dynamic updating as opposed to the static updating described before. The dynamic updating strategy allows good solutions being immediately available for reproduction, and thus can sometimes accelerate the convergence towards high-quality regions of the search space. In some sense, the dynamic updating resembles the steady-state replacement strategy used in genetic algorithms, while the static updating would be similar to a generational model (but only to some extent because the number of descendants is not fixed for all steps—see below).

3.4. Subset generation method

This method generates the groups of solutions that will undergo combination. As shown in Fig. 2, these groups are not necessarily pairs of solutions, but can be subsets of higher cardinality. This will ultimately depend on the particular combination method used, for not all such methods are capable of handling multi-parent combination.

A binary combination method has been considered in this work, and hence this subset generation method form couples of solutions. This is done exhaustively, producing all possible pairs. Let μ be the size of the reference set; then, the maximum number of pairs produced is λmax = μ(μ − 1)/2. Since the combination method utilized is deterministic, and no duplicates are accepted in the reference set, it does not make sense to combine again pairs of solutions that were already coupled before. For this reason, the actual number of pairs λ generated in a certain step will be always lower or equal than λmax.

3.5. Solution combination method

This method is fed with the subsets generated by the previous method, and produces new trial solutions by combining the information contained in each of these subsets. Some resemblance can thus be found with the functioning of crossover operators in genetic algorithms. Nevertheless, it must be noted that the latter are almost always
based on randomization (there are some exceptions though; e.g., see [5]); on the contrary, the use of deterministic solution combination methods in SS is not rare a vis. Such deterministic methods are often associated with the availability of problem-dependent heuristic rules, exploitable for combining solutions.

In this work, solution combination is achieved via path relinking (PR). PR is a strategy that was initially proposed for integrating diversification and intensification in tabu search. Grosso modo, PR is based on generating a trajectory in the search space. This trajectory starts from a certain solution (called initiating solution), and is headed towards another solution or set of solutions (called guiding solution(s)). This trajectory is constructed by performing moves on the current solution, such that attributes in the guiding solution are increasingly added, and attributes not in the guiding solution are increasingly dropped. This differs from classical local search techniques in which attributes are usually added/dropped at random, and kept only if a quality increase is attained. PR is not “blind to quality” though, for the move to be applied is commonly selected as the best (in some problem-specific sense) among the available ones. Once the trajectory is completed, the output of the procedure is the best solution found in it. The improvement method can be applied to this solution and/or to some solutions along the path.

The PR procedure used here is adapted to the tree representation utilized. Each step consists of swapping and/or transferring leaves between the left and right branches of a certain subtree. For this process, the corresponding subtree in the destination tree is taken as reference. Upon completion of the step, the leaf set in the right and left branches of the former subtree is the same as in the guiding solution. A more precise and detailed description of the process can be found in Figs. 3 and 4.

As it can be seen, the main procedure is recursive, being initially invoked using a copy of $T_{\text{source}}$ as the initial value of $T_{\text{best}}$, and sending pointers to the root of both the source and destination trees. Subsequently, each step rearranges the leaves of a part of the tree, so that they are relocated in the same relative position (i.e., left or right) with respect to the roots of the subtrees considered. A part of this rearrangement is done via swaps. If there are remaining leaves after the swapping, these are transferred to the corresponding tree, testing all possible insertion points. This involves evaluating the different tentative trees generated, a process whose cost is accounted in order to have a valid estimate of the associated computational overhead. To be precise, since the cost of evaluating a tree is $O(n^2)$ where $n$ is the number of leaves, each time a subtree with $i$ leaves is evaluated this is accounted as $i^2/n^2$ full evaluations. This is accumulated for the whole path relinking process, and added to the SS total.

4. Experimental results

The SS algorithm described in the previous section has been applied to five data sets comprising real biological data. These have been downloaded from TreeBASE, an online repository of publicly available data, and comprise DNA sequences for a number of taxa ranging from 85 up to 178 (see Table 1 for details). The selection of this test suite is intended to cover uniformly a reasonable range of instance sizes. These are well beyond the tenable limit for exact techniques such as branch and bound [43]. In all cases, distance matrices have been computed using the DNADIST program of the Phylip package. To be precise, the Kimura 2-parameter model (with default parameterization) has been used.

The SS algorithm utilized has a reference set of $\mu = 5$ solutions. The initial population from which the reference set is extracted at the beginning of the run is composed of $\lambda_{\text{max}} = \mu(\mu - 1)/2$ solutions. This is also the maximum size of the offspring population used for updating the reference set in each step. Whenever all pairs of solutions in the reference set have been mated without yielding a new improved solution, the reference set is restarted as mentioned in Section 3.1. The algorithm is run on each instance for a total number of $10,000n$

---

**Path Relinking in Tree Space – Main**

**INPUT:** three trees $T_{source}$, $T_{dest}$, and $T_{best}$, and pointers to subtrees $S$ and $S'$ ($S \in T$, $S' \in T'$)

**OUTPUT:** a tree $T_{best}$

(1) **if** $T_{source} \neq T_{dest}$ **then**

(a) Let $T$ be the output of applying the basic step to $S$ and $S'$. Replace $S$ by $T$ in $T_{source}$.

(b) **if** $\text{cost}(T_{source}) < \text{cost}(T_{best})$ **then**

   (i) Let $T_{best} \leftarrow T_{source}$

   (ii) Apply improvement method to $T_{best}$.

(c) Apply path relinking recursively to $T_{source}$, $T_{dest}$, $T_{best}$, $\text{left}(T)$, $\text{left}(S')$.

(d) Apply path relinking recursively to $T_{source}$, $T_{dest}$, $T_{best}$, $\text{right}(T)$, $\text{right}(S')$.

(2) **Output** $T_{best}$

Fig. 3. Pseudocode of the path relinking procedure. Input parameters are passed by reference. This way, $\text{left}(T) = \text{left}(S')$ after the first recursive call. Similarly, $\text{right}(T) = \text{right}(S')$ (and hence $T = S'$) after the second recursive call. Finally, $T_{best}$ stores the overall best tree found in the trajectory.

**Path Relinking in Tree Space – Basic Step**

**INPUT:** two trees $T$ and $T'$

**OUTPUT:** a tree $T_{best}$

(1) Let $L_1 \leftarrow L(\text{left}(T))$; let $L_2 \leftarrow L(\text{left}(T'))$; let $R_1 \leftarrow L(\text{right}(T))$; let $R_2 \leftarrow L(\text{right}(T'))$

(2) Let $M^- \leftarrow R_2 \cap L_1$; let $M^- \leftarrow L_2 \cap R_1$

(3) for $i = 1 : \min(|M^-|, |M^-|)$ **do**

   (a) Let $e$ (respectively $e'$) be the $i^{th}$ element of $M^-$ (respectively $M^-$).

   (b) Swap $e$ and $e'$ in $T$.

(4) Let $T_{best} \leftarrow T$

   Assume $|M^-| > |M^-|$ (otherwise change right by left, and $|M^-|$ by $|M^-|$ below.)

(5) for $i = |M^-| + 1 : |M^-|$ **do**

   (a) Remove $i^{th}$ element of $M^-$ from $\text{left}(T)$.

   (b) Let $T_{best} \leftarrow \text{WorstTree}$

   (c) for each insertion point $p$ in $\text{right}(T)$ **do**

      (i) Let $T_{trial} \leftarrow T$; insert $i^{th}$ element of $M^-$ in position $p$ of $\text{right}(T_{trial})$.

      (ii) if $\text{cost}(T_{trial}) < \text{cost}(T_{best})$ **then** $T_{best} \leftarrow T_{trial}$

   (d) Let $T \leftarrow T_{best}$

(6) **Output** $T_{best}$

Fig. 4. The basic step of path relinking. Elements within sets $M^-$ and $M^-$ are assumed enumerated according to their appearance order (from left to right) in $T$. 
evaluations, where \( n \) is the number of taxa. This has been repeated 10 times. The results are shown in Table 3. For comparison purposes, results for an EA taken from [4], using the PDG operator for recombination, and the SCRAMBLE operator for mutation (a subtree is selected at random, and its topology is rearranged) have been included. On the basis of this EA, a MA using the rotation operators described in Section 3.2 for local improvement is considered as well.

First of all, results for the three basic agglomerative algorithms, as well as for the neighbor-joining [37] and Fitch–Margoliash [11] algorithms are shown in Table 2. As expected, the complete-link algorithm provides much better performance for the quality criterion selected than the other approaches. According to the experience with smaller instances (for which the optimal solution can be calculated), this value usually lies near the optimum for this evaluation model.

Subsequently, the results of the population-based metaheuristics are shown in Table 3. As it can be seen, the MA provides better results than the EA, thus supporting the usefulness of the rotation operators for local improvement. As to SS, it is capable of improving the results of the MA, both with static and dynamic updating of the reference set. This improvement tends to increase with the number of taxa. To test the significance of these results, a statistical analysis has been conducted. A Wilcoxon rank sum test (also known as Mann–Whitney \( U \) test) [28] has been used for this purpose. This test does not assume normality of the samples (as for example \( t \)-test does). In this case, the test indicates that the difference of SS (both static and dynamic) with respect to the EA and MA is always significant (at the standard 5% significance level). The difference of the MA with respect to the EA is always significant as well. There is no significant difference between the static and dynamic SS.

A final test has been done using no heuristic seeding of the initial population in the SS algorithm. The results are shown in Table 4. As expected, the results are inferior to those of SS with seeding, thus confirming the usefulness of this approach; nevertheless, the algorithm manages to find solutions that improve those of the complete-link algorithm for all instances considered. The improvement with respect to the EA and MA is here statistically significant in all cases too. This confirms the strength of SS for tackling this problem.

A comment must be made regarding the computational cost of the algorithms. Both basic agglomerative algorithms and neighbor-joining run in the order of seconds; Fitch–Margoliash is more expensive, running in 1, 2, 8, 18 and 37 minutes for each of the data sets respectively (using the KITSCH program of the Phylip package); as to SS, it takes 1, 1.5, 3, 4.5 and 12 minutes, respectively (all times have been measured in a P4—3.06GHz under Windows XP). Of course, SS is an anytime algorithm, and hence the runtime ultimately depends on the number of evaluations allowed. Notice also that the problem being treated does not require real-time response, and therefore these times are perfectly acceptable.
(notice for example that the time required for pre-processing sequences and computing the distance matrix can be similar or longer than these times).

5. Conclusions

The main objective of this work has been analyzing the competitiveness of SS for phylogenetic
inference. With regard to this, the results must be considered satisfactory, since both EAs/MAs and classical heuristics could be beaten. The difference with respect to the complete-link algorithm is not large, but recall that the later is known to be of high quality. Notice also that some biologically-relevant changes in the tree topology may lead to just small changes in the total tree weight. At any rate, it must be stressed that finding a collection of high-quality, biologically-realistic phylogenetic trees is very important to ground further analysis, e.g., for building consensus trees [20]. To this end, SS constitutes a practical approach for improving raw solutions provided by this or other heuristics.

It is interesting to notice the fact that the SS algorithm presented does not use randomization except for generating the initial population, and restarting the reference set. The deterministic strategies presented here worked well on these data instances. Of course, by no means this implies that randomization plays a minor role, for these diversification phases are essential for the performance of the algorithm. Actually, the algorithm presented could be somehow considered as a hybrid of MA and SS (the diversification phase realized for restarting the reference set can be viewed as a memetic generational step). In this sense, this approach combines ideas from both worlds with very encouraging results.

There are many lines for future developments. On one hand, multi-tier strategies for managing the reference set can be tried. This raises the issue of selecting appropriate measures for evaluating diversity in this context. Regarding this, some distance measures on trees have been defined in the literature—e.g., the Robinson and Foulds metric [35], quartet distance [9], etc.—and could be used for this purpose. Other lines for future work can be found in the use of different improvement operators. As mentioned in Section 3.2, alternative tree neighborhoods have been defined in the literature. It may be interesting to test the performance of SS with a different improvement method, and even consider the possibility of using simultaneously a number of different methods. Preliminary results indicate that these trajectory-based improvement methods may be not competitive with population-based metaheuristics as stand-alone techniques in the particular context presented in this work (although they have provided good results in a parsimony-based context [1]). Nevertheless, their integration within SS might be fruitful.

Acknowledgments

This work is partially supported by Spanish MCyT and FEDER under contract TIC2002-04498-C05-02. Thanks are due to the anonymous reviewers for their detailed comments, and to Dr. Moscato for continuous (and compelling) discussions on bioinformatics.

References
