Reconstructing Phylogenies with Memetic Algorithms and Branch-and-Bound

José E. Gallardo, Carlos Cotta Antonio J. Fernández
Dept. Lenguajes y Ciencias de la Computación, ETSI Informática,
University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain.
E-mail: {pepeg,ccottap,afdez}@lcc.uma.es

Abstract

A phylogenetic tree represents the evolutionary history for a collection of organisms. We consider the problem of inferring such a tree given a certain set of data (genomic, proteomic, or even morphological). Given the computational hardness of this problem, exact approaches are inherently limited. However, exact techniques can still be useful to endow heuristic approaches with problem-awareness. We analyze this hybridization in the context of memetic algorithms and branch-and-bound techniques. Focusing in the ultrametric model for phylogenetic inference, we show that this combination can be synergetic. We analyze the parameters involved in this hybrid model, and determine a robust setting for these. A summary of related work is also provided.

1 Introduction

Phylogenetic trees provide a hierarchical representation of the degree of closeness among a set of organisms. The reconstruction of phylogenetic trees from data is undoubtedly a task of paramount importance in molecular biology. It has direct implications in areas such as multiple sequence alignment[20], protein structure prediction[53] or molecular epidemiological studies of viruses[48], just to cite a few. Unfortunately, this task turns out to be very hard from the computational point of view. First of all, the phylogeny problem is intrinsically complex: NP-hardness has been shown for phylogenetic inference under several models[11, 12, 13, 17, 62]. Secondly, while the utilization of a quality measure for evaluating hierarchies implies the definition of an 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.

Due to the reasons just mentioned, the use of classical exact techniques can be considered generally inappropriate in this context. Indeed, 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[58]) to complex metaheuristics (e.g., evolutionary algorithms[9]). At any rate, it is well-known that any heuristic method is going to perform in strict accordance with the amount of problem-knowledge it incorporates[10, 61]. In this sense, while classical exact techniques are not adequate as stand-alone solvers for this problem, they can still be very useful to create powerful problem-aware metaheuristics. We will precisely explore this possibility here, presenting a model for the integration of branch-and bound techniques (BnB)[35] and memetic algorithms (MAs)[45, 46, 47]. As it will be shown, this model can result in a synergistic combination yielding better results than those of the intervening techniques alone.

2 A Crash Introduction to 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[29].

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[14], e.g., the Jukes-Cantor model[27], the Kimura 2-parameter model[31], etc. 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, MP is grounded on Occam's razor, whereby given two equally predictive theories, the simpler should be chosen. Thus, the 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 founded on transforming the available sequence data into an $n \times n$ matrix M. This matrix is the unique information subsequently used. More precisely, edges in T are assigned a weight. We denote by $w(T, e)$ the weight of edge e in T. 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. 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 $\mathcal{V}(T)$ and $\mathcal{E}(T)$ respectively denote the set of vertices and edges of tree T, and let $\mathcal{L}(T)$ denote the set of all leaves of tree T. We further use $T(v)$, where $v \in \mathcal{V}(T)$, to denote the subtree of T rooted at node v, and $\pi(T, v)$ to denote the parent of v in T. Then, $\hat{M}_{ij} = \Delta(T, i, j) + \Delta(T, j, i)$, where

$$\Delta(T,a,b) = \begin{cases} w(T,(a',a)) + \Delta(T,a',b) & \text{if } b \notin \mathcal{L}(T(a)),\ a' = \pi(T,a) \\ 0 & \text{if } b \in \mathcal{L}(T(a)) \end{cases} \tag{1}$$

The quality of the tree can now be quantified in a variety of ways. On one hand, it is possible to consider some meta-distance measure between observed and inferred distances, e.g., the L_2 metric,

$$L_2(M, \hat{M}) = \sum_{i=1}^{n}\sum_{j=1}^{n}(M_{ij} - \hat{M}_{ij})^2, \tag{2}$$

thus seeking a least-squares approximation of the observed distance. To do so, a quadratic program must be solved in order to find the best edge weights for a given topology[40]. This may be computationally demanding when used within a search-and-score method.

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 $\hat{M}_{ij} \geqslant M_{ij}$, 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[1]. In this situation, minimizing the total weight of T (i.e., the sum of all edge-weights) is usually the criterion. Notice that by taking M_{ij} as the minimum number of evolutionary events needed to transform i in j, this 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[24]. 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

$$\hat{M}_{ij} \leqslant \max\{\hat{M}_{ik}, \hat{M}_{jk}\},\ 1 \leqslant i,j,k \leqslant n\ . \tag{3}$$

If $\hat{M}$ is ultrametric, the distance in T between any internal node h and any leaf l descendant of h is the same — e.g., see Figure 1 where the optimum ultrametric tree for an arthropod dataset[26] is depicted. This distance represents in each case the elapsed time since the corresponding evolutionary divergence event. Hence, a constant evolution rate is implicitly assumed, in the line of the molecular-clock hypothesis[39]. Although this hypothesis is not in vogue nowadays, this condition still provides a very good approximation to the optimal solution under more relaxed assumptions such as mere additivity[2]. Much more important, 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[62]. This can be solved as follows: let the *height* of a subtree be computed as:

$$\mathit{height}(T) = \begin{cases} 0 & \text{if } T = (l) \\ \max\left(\mathit{height}(L), \mathit{height}(R), \frac{D(L,R)}{2}\right) & \text{if } T = (h, L, R) \end{cases} \tag{4}$$

[1]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*[39].

[2]$\hat{M}$ is additive if for any i, j, k, l, the maximum of $\hat{M}_{ij} + \hat{M}_{kl}$, $\hat{M}_{ik} + \hat{M}_{jl}$, $\hat{M}_{il} + \hat{M}_{jk}$ is not unique.

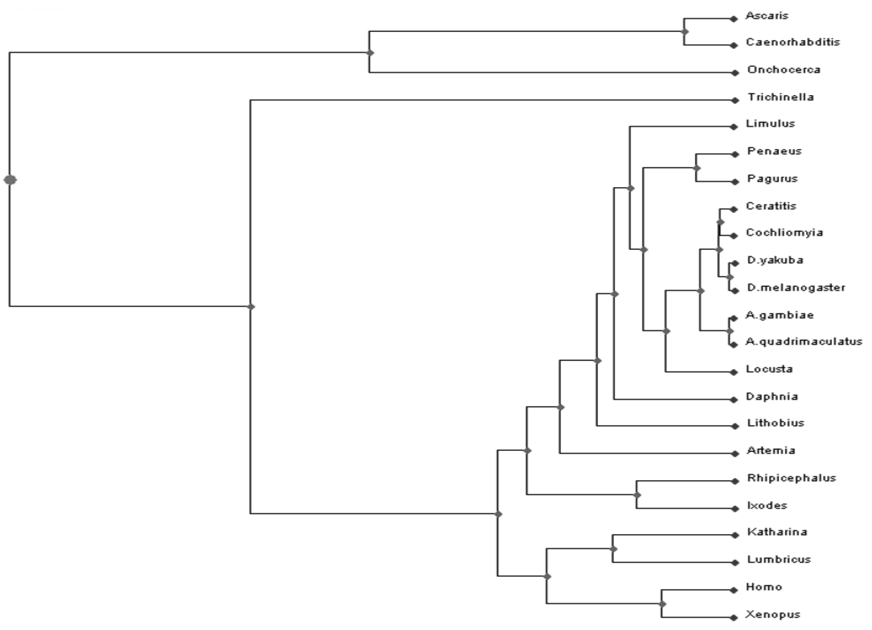


Figure 1: An example of ultrametric tree. Branch lengths represent the elapsed evolutionary time between the corresponding species. Notice that the horizontal distance from an interior node to any of its leaves is the same.

where $D(L,R) = \max\{M_{ij} \mid i \in \mathcal{L}(L), j \in \mathcal{L}(R)\}$. Now, the weight of an edge connecting two nodes i and j is easily computed as

$$w\,(T,(i,j)) = |height\,(T(i)) - height\,(T(j))|\ . \tag{5}$$

Unlike this simple computation, finding optimal edge weights for an additive tree requires solving a linear program with $2n-2$ variables (the number of internal edges), and $n(n-1)/2$ binary constraints, one for each pair of taxa (i,j), corresponding to $\hat{M}_{ij} \geqslant M_{ij}$.

Notice that the number of possible phylogenetic trees for a given set of n taxa is huge: there are $(2n-3)!!$ rooted trees[24], 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 8.2×10^{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 instances. Hence, the use of heuristic techniques seems appropriate.

Most typical heuristics for phylogenetic inference are variants of the *single-link*[57], *complete-link*[32], and *average-link*[58] algorithms. These are agglomerative clustering algorithms whose functioning matches the generic template shown in Figure 2. 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 distance measures shown in Table 1. Notice that 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 equation (4)– and complete-link makes local decisions trying to minimize this quantity).

Other popular heuristics for phylogenetic inference are based in metaheuristic approaches, such as for example evolutionary algorithms (EAs). Next section will briefly overview previous work done in the application of EAs in this domain.

Agglomerative Clustering Algorithm

```
    INPUT:  An $n \times n$ distance matrix $M$.
    OUTPUT: A tree $T$.

1:  for $i = 1 : n$ do let $T_i \leftarrow (i)$ end for
2:  let $N_{clusters} \leftarrow n$
3:  while $N_{clusters} > 1$ do
4:      Select $T_i$ and $T_j$ $(1 \leqslant i < j \leqslant N_{clusters})$ for which
          $dist(T_i, T_j)$ is minimal.
5:      let $T_i \leftarrow (h, T_i, T_j)$
6:      let $T_j \leftarrow T_{N_{clusters}}$
7:      let $N_{clusters} \leftarrow N_{clusters} - 1$
8:  end while
9:  return $T_1$
```

Figure 2: Pseudocode of an agglomerative clustering algorithm. In line 5, h represents an undistinguishable internal node of the tree.

Table 1: Distance measures for determining the two *closest* clusters during agglomerative clustering.

Single-Link	:	$dist(T, T') = \min\{M_{ij} \mid i \in \mathcal{L}(T), j \in \mathcal{L}(T')\}$
Complete-Link	:	$dist(T, T') = \max\{M_{ij} \mid i \in \mathcal{L}(T), j \in \mathcal{L}(T')\}$
Average-Link	:	$dist(T, T') = \frac{1}{\lvert\mathcal{L}(T)\rvert \cdot \lvert\mathcal{L}(T')\rvert} \sum_{i \in \mathcal{L}(T), j \in \mathcal{L}(T')} M_{ij}$

3 Evolutionary Algorithms for the Phylogeny Problem

To the best of our knowledge, the first application of a genetic algorithm (GA) to phylogenetic inference was developed by Matsuda. The GA was used to construct phylogenetic trees from amino acid sequences using a ML approach[41, 42]. It was shown that the performance of the method was comparable to those of other tree-construction methods (i.e., neighbor-joining[54], MP, ML and UPGMA combined with different search algorithms). Later, an improved genetic algorithm was applied to *rbc*L sequence data of green plants[38]. The results were really promising as the GA required only 6% of the computational effort required by a conventional heuristic search using tree bisection/reconnection branch swapping to obtain the same maximum-likelihood topology.

Further work by Skourikhine[56] reported a self-adaptive genetic algorithm (GA) for the ML reconstruction of phylogenetic trees using nucleotide sequence data. The algorithm produced faster reconstructions of the trees with less computing power and automatic self-adjustment of settings of the optimization algorithm parameters. Other evolutionary proposals for searching the ML trees were developed by Meade *et al.*[43] and Katoh *et al.*[28] (considering simple GAs), and Lemmon and Milinkovitch[37] (using a multi-population GA). Parallel GAs under the ML optimality criterion were investigated by Brauer *et al.*[5] as well, with encouraging results. Also in the ML context, Shen and Heckendorn[55] showed that discretizing edge lengths changed the fundamental character of the search and could produce higher quality trees.

The representation issue has attracted a lot of interest in the application of EAs to the Phylogeny problem. Indeed, the choice of representation (and subsequently, the choice of operators) has a clear influence on the performance of the algorithm as shown by Reijmers *et al.*[51]. A number of different EAs –based on the use of alternative representations (direct and indirect) and/or reproductive operators– were developed, compared and evaluated using a distance-based measure by Cotta and Moscato[9]. The conclusion was that directly evolving phylogenetic trees yields better results than indirect approaches using decoders. The exception to this rule was shown by a greedy permutational-decoder based EA that provided the overall best results, achieving near 100%-success at a lower computational cost than the remaining approaches. However this

Branch and Bound Algorithm for the MUT Problem

```
INPUT:  An $n \times n$ distance matrix $M$.
OUTPUT: The minimum ultrametric tree for $M$.

 1:  Relabel taxa such that $(1, 2 \ldots, n)$ is a maxmim permutation
 2:  Create the root node $R = (h, (1), (2))$ of the BBT representing
     the unique topology with leaves 1 and 2
 3:  let $U \leftarrow$ Complete Link$(M)$; $UB \leftarrow w(U)$
 4:  let $open \leftarrow \{R\}$
 5:  while $open \neq \varnothing$ do
 6:    $open \leftarrow \{T \mid T \in open, LB(T) < UB\}$
 7:    Extract some $T$ from $open$
 8:    Compute $MUTT(T)$, $w(MUTT(T))$ and $LB(T)$
 9:    if $\mathcal{L}(T) = \{1, 2, \ldots, n\}$ then
10:      if $w(MUTT(T)) < UB$ then
11:        let $U \leftarrow MUTT(T)$; $UB \leftarrow w(MUTT(T))$
12:      end if
13:    else if $LB(T) < UB$ then
14:      let $open \leftarrow open \cup$ children of $T$
15:    end if
16:  end while
17:  return $U$
```

Figure 3: Pseudocode of Branch and Bound Algorithm for the MUT Problem.

latter approach is less scalable than its direct counterpart. Poladian[49] also presented some results in this line of research and proposed different representations for the phenotype and the genotype in two existing algorithms for phylogenetic inference (i.e., neighbor-joining and ML) co-utilized within a GA. One conclusion reached is that working directly with the trees presents some disadvantages as for instance to establish the concept of distance between tree topologies. Also, separating the genotype from the phenotype requires the use of a good mapping from one space to the other (although this case allows the use of a different set of genetic operators). The direct representation is used in the Gaphyl package by Congdon[6], whereas a permutational decoder is used by Kim *et al.*[30].

A potential difficulty for the inference problem may arise when different data sets provide conflicting information about the inferred 'best' tree(s). Poladian and Jermiin[50] have proposed the application of evolutionary multi-objective optimization algorithms (EMOOA) in this case. They showed that EMOOA can resolve many of the issues concerned with analyzing multiple data sets that give conflicting signals about evolutionary relationships.

There have been also evolutionary proposals based on the parsimony criterion. For example, Ribeiro and Vianna[52] described a genetic algorithm that makes use of an innovative optimized crossover strategy which is an extension of path relinking[19]. This proposal was shown to be computationally promising. Cotta[8] described an approach based on scatter search[34] that uses path relinking too (although not based on the parsimony criterion but on distance methods). Hill *et al.*[23] presented a GA-based approach for weighted maximum parsimony phylogenetic inference, especially for data sets involving a large number of taxa.

This summary of evolutionary approaches to phylogenetic inference is not exhaustive. The reader may check the survey by Fogel[16] for further information on applications of evolutionary computation to the inference of phylogenies, and highlights of new and current challenges for facing this problem.

4 A BnB Algorithm for Phylogenetic Inference

As it was stated in Section 2, the *min ultrametric tree with a given topology* problem ($MUTT(T)$) can be determined in time $O(n^2)$, for a topology T containing n taxa. Hence, it suffices to solve this problem for each possible topology in order to find the best ultrametric tree (i.e., the one with minimum total weight –

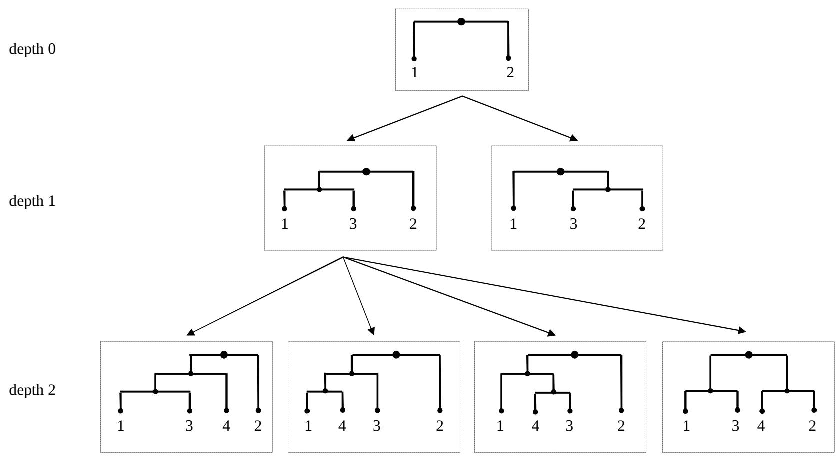


Figure 4: Partial Branch and Bound Tree.

the MUT problem). However, this approach is not suitable in practice as the number of topologies increases exponentially with n. BnB algorithms can be used with the aim of reducing the size of the explored space. For this purpose, the BnB algorithm[62] (see Figure 3) will explore a BnB tree (BBT) that contains in its leaves all possible topologies of ultrametric trees with leaf set $\{1, \ldots, n\}$. Let zero be the depth of the root of the BBT and $i + 1$ the depth of a child of a node with depth i. Therefore, internal nodes in the BBT at depth i correspond to topologies with leaf set $\{1, \ldots, i + 2\}$. Observe that any topology corresponding to a node in the BBT at depth i has $2 + 2i$ edges (see Figure 4). Accordingly, a *branching rule* that inserts the next taxon into each edge of the current topology can be used to generate the BBT. In this way, a node at depth i in the BBT has $2 + 2i$ children.

With the aim of pruning the BBT, the BnB maintains an upper bound corresponding to the weight of the best ultrametric tree found so far. For each internal node T_i in the BBT corresponding to a minimum ultrametric tree with leaf set $\{1, \ldots, i\}$, a lower bound $LB(T_i)$ on the weight of the minimum ultrametric tree T_n with leaf set $\{1, \ldots, n\}$ whose topology contains T_i can be calculated using the following inequality:

$$w(T_n) \geqslant w(T_i) + \sum_{i<j\leqslant n} min\{M_{kj} \mid k < j\}/2. \tag{6}$$

If this bound exceeds the value of the current upper bound, the node T_i and all its children can be safely pruned, as they will not lead to a better solution.

Initially, the complete-link algorithm described in Section 2 is used in line 3 to find a feasible solution, and the weight of this solution is set as upper bound. Clearly, finding large lower bounds early in the exploration of the BBT is helpful as the number of pruned nodes increases during the execution of the BnB algorithm. With the aim of maintaining a lower bound as large as possible, a heuristic order for the n taxa will be used. This way, as a preprocessing step, taxa are sorted to constitute a *maxmin* permutation, i.e., a permutation $(a_1, a_2, \ldots, a_n)$ of $\{1, \ldots, n\}$ such that $M_{a_1a_2} = \max(M)$ and $min_{k<i}\{M_{a_ia_k}\} \geqslant min_{k<i}\{M_{a_ja_k}\}$ for all $1 < i < j$. Note that a *maxmim* permutation for an $n \times n$ matrix can be found in $O(n^2)$ time.

As to the root node of the BBT, an interesting property regarding optimal ultrametric trees is that if $M_{uv} = \max(M)$, then there exists a minimum ultrametric tree $T = (r, L, R)$, such that $u \in \mathcal{L}(L)$ and $v \in \mathcal{L}(R)$. Since taxa are organized in a *maxmim* permutation, $M_{12} = \max(M)$, and hence the root of the BBT can be set as done in line 2.

The so generated BBT can be traversed in several ways. The most efficient (in terms of the number of iterations required to find the optimum and prove its optimality) is best-first, i.e., expanding firstly the most promising –according to the lower bound– nodes. However, the memory requirements can make this strategy unrealistic for large problem instances. The alternative is using a depth-first traversal. This strategy does not require large amounts of memory, but can expand much more nodes than best-first. A third option is using a breadth-first traversal (i.e., every node in a level is explored before moving to the next). In principle, this option would have the drawbacks of the previous two strategies, unless a heuristic choice is made: keep at each level just the best (according to some *quality* measure) k nodes. This implies sacrificing exactness, but provides a very effective heuristic search approach. The name *beam search* (BS) has been coined to denote this strategy[3, 60].

Memetic Algorithm

```
INPUT:  An $n \times n$ distance matrix $M$.
OUTPUT: A tree $T$, inducing a near optimal matrix $\hat{M}$.

for $i = 1 : popsize$ do
    let $pop[i] \leftarrow$ RANDOMTREE($n$)
    EVALUATE($pop[i]$, $M$)
end for
let $N_{eval} \leftarrow 0$.
while $N_{eval} < maxevals$ do
    for $i = 1 : offsize$ do
        if recombination is performed then
            let $parent_1 \leftarrow$ SELECT($pop$); $parent_2 \leftarrow$ SELECT($pop$)
            let $offspring[i] \leftarrow$ RECOMBINATION($parent_1$, $parent_2$)
        else let $offspring[i] \leftarrow$ SELECT($pop$)
        end if
        if mutation is performed then
            let $offspring[i] \leftarrow$ MUTATE($offspring[i]$)
        end if
        Evaluate($offspring[i]$, $M$)
        let $N_{eval} \leftarrow N_{eval} + 1$
        if local improvement is performed then
            let $offspring[i] \leftarrow$ IMPROVE($offspring[i]$)
        end if
    end for
    $pop \leftarrow$ REPLACE($pop$, $offspring$)
end while
let $T \leftarrow$ BEST($pop$).
return $T$
```

Figure 5: Pseudocode of the memetic algorithm.

Unfortunately, BnB algorithms alone need too much time to be practical, except for very small sets of taxa. As we will show, a hybrid algorithm based on this latter strategy (BS) and a MA can provide better performance.

5 A Memetic Algorithm for Phylogenetic Inference

Memetic algorithms constitute a family of metaheuristics that blend together concepts from population-based techniques (e.g., evolutionary algorithms – EAs) and trajectory-based techniques (e.g., simulated annealing). Their central philosophy thus relies in two major pillars: individual improvement plus populational cooperation. More precisely, a MA can be characterized as a population of *agents* that alternate periods of self-improvement with periods of cooperation, and competition. The term 'agent' is purposefully used to indicate that the population constituents are not mere 'individuals', that is passive entities; on the contrary, they are active, and try to exploit all available knowledge about the target problem. This concern for adapting to the problem is backed up by ground-breaking theoretical results such as the *No Free Lunch Theorem*[61], and it is ultimately responsible for the impressive success record of MAs.

Adaptation to the target problem is achieved via the use of constructive heuristics, local-search methods, exact techniques, etc. For this reason, it is often the case that MAs are used under different names, such as 'Lamarckian EAs', or more generally 'hybrid EAs'. For the particular application we consider, namely the inference of phylogenetic trees from distance matrices, our MA uses problem-specific operators for manipulating solutions, and an *ad hoc* local-improvement scheme. The overall process is shown in Figure 5.

First of all, the population for this MA must be initialized. Each agent in the MA population represents

Prune-Delete-Graft Recombination

```
INPUT:  Two trees, $T_1$ and $T_2$ to be recombined.
OUTPUT: A tree, corresponding to the offspring.

1:  Select a subtree $T$ from $T_2$.
2:  for each $l \in \mathcal{L}(T)$ do
3:    Find subtree U from $T_1$ such that $U = (h, (l), U')$
      or $U = (h, U', (l))$.
4:    Replace $U$ by $U'$ in $T_1$.
5:  end for
6:  Select a random subtree $V$ from $T_1$.
7:  Replace $V$ by $V' = (h', T, V)$ in $T_1$, where $h'$ is a new vertex.
8:  return $T_1$
```

Figure 6: Pseudocode of the PDG recombination.

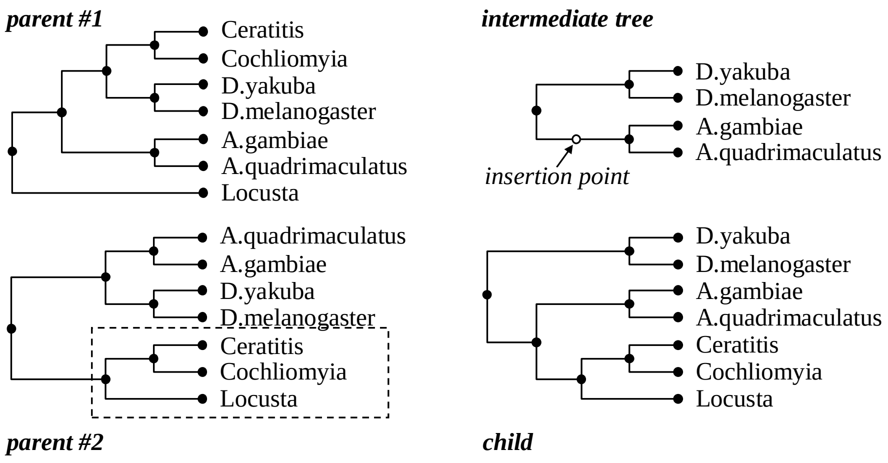


Figure 7: An example of PDG recombination.

a feasible tree, and hence the search is directly performed in the space of all possible phylogenetic trees[9]. These trees are initially created at random. Subsequently, the MA enters the main reproductive loop which comprises selection, recombination, mutation, local improvement, and replacement. Among these, the first and fifth components are rather problem independent, whereas the second to the fourth components are specifically crafted to the problem. Let us focus in these problem-aware operators.

The recombination operator is intended to produce new solutions by mixing the information comprised in several parental solutions. In this context, this mixing process amounts to the transference of tree-topology information from parents to offspring. To do so, a branch-exchange procedure can be used, i.e., a subtree can be pruned from one of the parents, and grafted onto the second one, much like it is done in the field of Genetic Programming[33]. Note however that phylogenetic trees are constrained to have exactly n leaves, each one representing a different species. Therefore, the recombination operator has to be careful of removing duplicates elements. Taking into account these considerations, the PRUNE-DELETE-GRAFT (PDG) tree crossover operator (see the pseudocode in Figure 6) has been used[9, 44]. PDG is a three-step procedure for recombing two trees T_1 and T_2 by pruning a subtree T from $T2$, deleting from T_1 all leaves occurring in T, and grafting T at a randomly selected point in $T1$. This is illustrated in Figure 7.

As to mutation, its purpose is to introduce new information (i.e., new topological relationships) in a certain solution. This can be accomplished in several ways. In this case, the MA considered uses the so-called SCRAMBLE operator that first selects a subtree Q from the tree being mutated, and then rearranges its topology in a random way – see Figure 8. Note that this mutation operator fulfills the previously mentioned constraint regarding the presence of exactly n leaves in the tree.

Scramble Mutation

```
Input:  A tree $T$ to be mutated.
Output: The same tree $T$ after mutation.

1:  Select a random subtree $Q$ from $T$.
2:  let $k \leftarrow |\mathcal{L}(Q)|$
3:  let $R \leftarrow$ RandomTree($k$)
4:  Relabel leaves in $\mathcal{L}(R)$ as leaves in $\mathcal{L}(Q)$.
5:  Replace $Q$ by $R$ in $T$.
6:  return $T$
```

Figure 8: Pseudocode of the Scramble mutation.

Finally, local search has been applied to individuals in the population with the aim of improving solutions produced by the recombination and mutation methods. 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[1]. In this work, the improvement method chosen is based on performing rotations within the tree.

Four symmetric operations are used within the local search improvement method. The first one, ROT^1_R, is defined as

$$ROT^1_R\,[\ (h,(h',T_{LL},T_{LR}),T_R)\ ] = (h,T_{LL},(h',T_{LR},T_R))\ \ , \qquad (7)$$

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. A ROT^2_R operation would have performed the same movement on T_{LL} rather than on T_{LR}.

Analogously, ROT^1_L and ROT^2_L are mirror-inverted versions of the previous operations. For each interior node of the tree, it is first checked whether a ROT_R movement is possible, and if so, whether ROT^1_R or ROT^2_R produce an improvement. If this were the case, the change would be retained, and the improvement method would stop. Otherwise, ROT_L movements would be analogously attempted. If no improvement is possible either, the procedure is recursively applied to the left and right subtrees of h.

The application of this local improvement strategy may be costly if applied to every new solution created by the MA. This circumstance has been also recognized in other domains[7, 25], where the use of partial Lamarckism has been proposed. We have opted for a variant of this strategy, performing local search only when the quality of a solution improves the current incumbent.

6 A Hybrid Algorithm

Branch-and-Bound and memetic algorithms represent two very different approaches for tackling the MUT problem. These approaches are not incompatible though. In this section, we describe a hybrid model that combines these two techniques. To be precise, it executes the BnB and MA in an interleaved way. The goal is combining synergistically these two different solving approaches, exploiting the capability of BnB for identifying provably good regions of the search space, and the power of the MA for exploring these. This hybrid algorithm is described in Figure 9.

The algorithm starts by executing BS (i.e., *Beam Search*, see Section 4) for l_0 levels of the search tree. Afterwards, the MA and BS are interleaved until a termination condition is reached. Every time the MA is run, its population is initialized using random nodes in the BS queue. Let us note that nodes in the BS queue represent partial trees in which some taxa are included but another ones are not, so they must first be converted in full trees. For this purpose, the greedy completion algorithm described in Figure 10 is applied to every partial tree.

The intended goal of the initialization explained above is to lead the MA to these regions of the search space (recall that the nodes in the queue represent subsets of the search space considered promising by the BnB; hence, the MA is used for finding probably good solutions in this region). Upon stabilization of the MA, control is returned to the BnB algorithm. The lower bound for the optimal solution obtained by the

Hybrid Algorithm for the MUT Problem

```
1:  for $l_0$ levels do run BS
2:  do
3:    select randomly $popsize$ nodes from problem queue
4:    initialize MA population with selected nodes
5:    run MA
6:    if MA solution < BS solution then
7:      let BS solution ← MA solution
8:    for $l$ levels do run BS
9:  until timeout or tree exhausted
10: return BS solution
```

Figure 9: Pseudocode of the hybrid algorithm.

Greedy Completion Algorithm

```
    INPUT:  A tree $T$ and a list of nodes $ns$.
    OUTPUT: The tree obtained by inserting all nodes in $ns$ into $T$.

1:  while $ns \neq \varnothing$ do
2:    Extract $n$ from $ns$
3:    let $best \leftarrow \infty$
4:    for each insertion point $p$ in $T$ do
5:      let $T_{trial} \leftarrow$ insert $n$ at position $p$ of $T$
6:      if $w(T_{trial}) < best$ then
7:        let $best \leftarrow w(T_{trial})$; $T_{best} \leftarrow T_{trial}$
8:      end if
9:    end for
10:   let $T \leftarrow T_{best}$
11: end while
12: return $T$
```

Figure 10: Pseudocode of the Greedy Completion Algorithm.

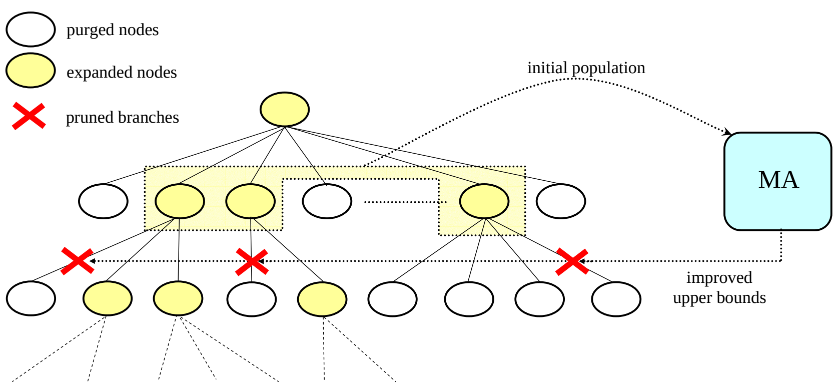


Figure 11: Sketch of the hybrid algorithm.

MA is then compared to the current incumbent in the BnB, updating the latter if necessary. This may lead to new pruned branches in the BS tree; see Figure 11 for an outline of the general process. This is repeated until the search tree is exhausted or a time limit is reached.

Several parameters can be tuned to control the algorithm. The aim of parameters l_0 and l is to control the balance between MA and BS. Parameter l_0 indicates how many levels the BS descends before starting running the MA. A lower value for l_0 would give more computation time to the MA component of the algorithm. On the other hand, parameter l indicates how often, after this initial descent, should the MA be run. In this case, a low value for this parameter would execute more often the MA component of the algorithm, thus increasing its influence in the final outcome. Furthermore, parameter k controls the maximum number of partial solutions maintained on each level of the BS. Increasing this parameter improves the quality of the results obtained by the BS component, but a very high value may delay the descent along the search tree.

7 Experimental results

The evaluation of the hybrid model has been approached in two stages. Firstly, a sensitivity analysis was performed to determine an appropriate parameterization. Subsequently, the model was deployed on a full set of problem instances, and compared with its constituent algorithms as stand-alone techniques. Before presenting these results, lets us briefly describe the experimental setting.

7.1 Experimental Setting

The experiments have been performed in a Pentium IV PC (2400MHz and 512MB of main memory). All algorithms were coded in C and compiled using gcc 3.2. For the MA and hybrid algorithm, the experiments were done using a steady-state evolutionary algorithm ($popsize = 100$, $p_m = 1/(2n-1)$, $p_X = 0.9$, binary tournament selection). With the aim of maintaining diversity, duplicated individuals were not allowed in the population.

The different algorithms have been applied to five data sets comprising real biological data. These have been downloaded from TreeBASE[3], an online repository of publicly available data, and comprise DNA sequences for a number of taxa ranging from 85 up to 178 (see Table 2 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. In all cases, distance matrices have been computed using the DNADIST program of the Phylip package[4]. To be precise, the Kimura 2-parameter model (with default parameterization) has been used.

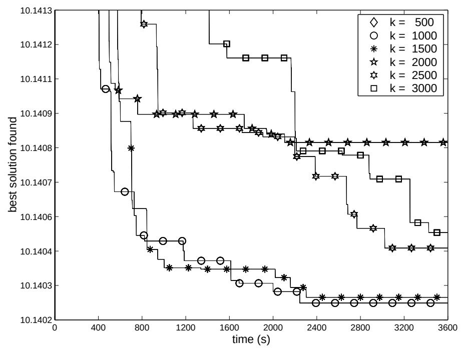
[3]`http://www.treebase.org`
[4]`http://evolution.genetics.washington.edu/phylip.html`

Table 2: Description of the data sets used in the experimentation.

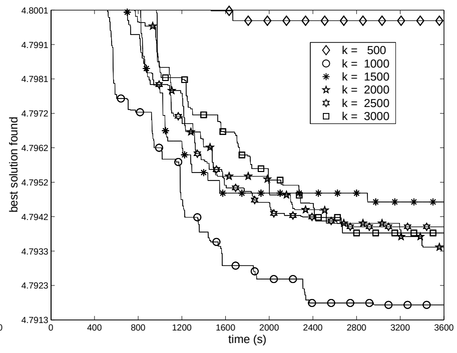
	M420	M1097	M877	M971	M808
number of taxa	85	107	134	158	178
sequence length	1016	2084	2684	1193	3453
data source	[59]	[18]	[21]	[4]	[22]

7.2 Sensitivity Analysis on the Hybrid Algorithm

As noted in Section 6, the functioning of the hybrid algorithm can be tuned using several parameters. Aiming to determine good values for these parameters, a sensitivity analysis was done for problems in the data set. First of all, different values for k (the search breadth in BS) were tested. Figure 12 shows the evolution of the hybrid algorithm for values of $k \in \{500, 1000, 1500, 2000, 2500, 3000\}$. As it can be seen, the best quality of the solution is obtained for $k = 1000$. For $k = 500$, the performance of the hybrid algorithm degrades because too few nodes are kept by the BS. For greater values, effectiveness also decreases since most of the computation is dedicated to the BS part of the hybrid algorithm.



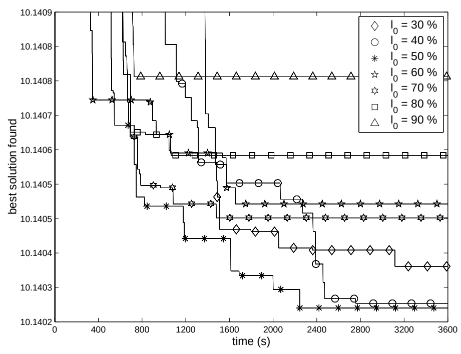
(a) M877



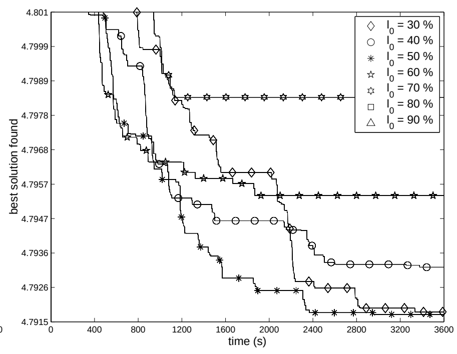
(b) M971

Figure 12: Evolution of best fitness for different values of k for different instances.

Next, the l_0 parameter was analyzed. Figure 13 shows the evolution of the hybrid algorithm for different values of l_0 (percentages are with respect to the number of taxa). The best performance of the algorithm is obtained when the MA starts running after the BS has descended 50% of the number of taxa. For $l_0 \in \{30\%, 40\%\}$ performance is slightly worse and it decreases considerably for greater values of l_0. Lastly,



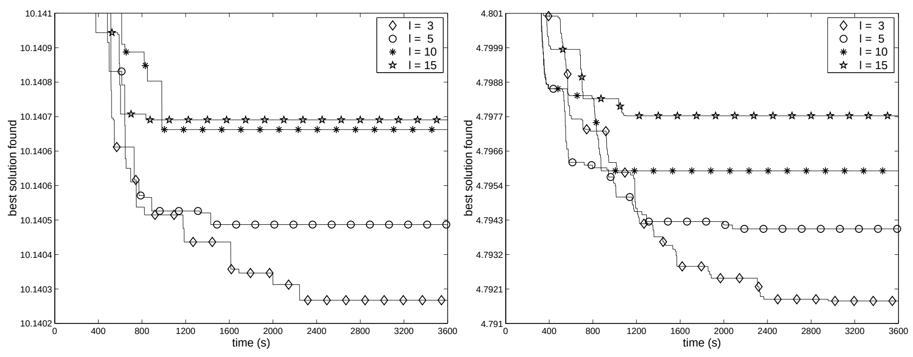
(a) M877



(b) M971

Figure 13: Evolution of best fitness for different values of l_0 for different instances.

for the l parameter, the best value is $l = 3$ (see Figure 14). Efficiency clearly improves with lower values for l.



(a) M877 (b) M971

Figure 14: Evolution of best fitness for different values of l for different instances.

7.3 Analysis of Results

After the preliminary study in Section 7.2, the parameters for the hybrid algorithm were set to: $k = 1000$, $l_0 = 0.5n$ and $l = 3$. For the BS algorithm alone, the greedy algorithm described in Figure 10 was applied to the best partial solution in each level of the BBT, to improve the incumbent solution. A single execution for each instance was performed for the BS algorithm (it is a deterministic method) whilst 25 independent runs per instance were carried out for the MA and hybrid algorithm. All algorithms were run for 3600 seconds.

First of all, results for the three basic agglomerative algorithms, as well as for the neighbor-joining[54] and Fitch-Margoliash[15] algorithms are shown in Table 3. 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.

Table 3: Results of the classical heuristics on the data sets used.

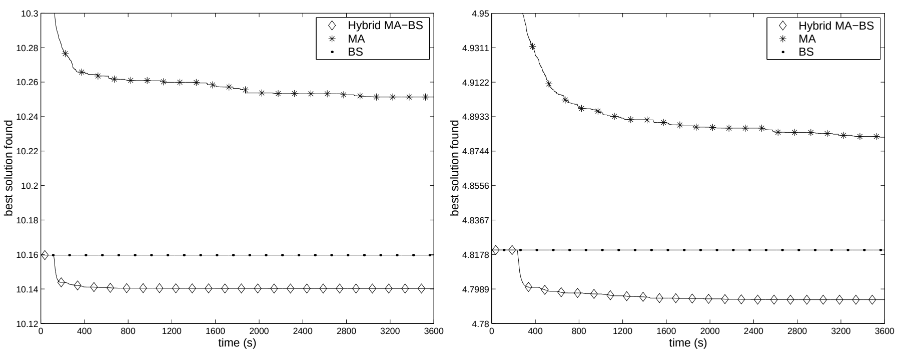
	M420	M1097	M877	M971	M808
single-link	2.93600	1.26385	53.82255	6.47470	66.51205
complete-link	2.35685	1.01205	10.15965	4.82025	11.58550
average-link	2.50110	1.04855	10.89800	5.15390	12.45225
neighbor-joining	3.44775	1.24985	18.21510	5.46825	38.20275
Fitch-Margoliash	2.63675	1.09675	29.09415	5.91770	31.86065

The results of the BS, MA and hybrid algorithms are shown in Table 4. Figure 15 compares the temporal evolution of the best solution found for the BS, MA and hybrid algorithm for two different instances in the data set. As it can be seen, the BS algorithm does not improve the initial bound provided by the complete link algorithm. Results for the MA alone are worse than the ones provided by BS. The hybrid algorithm provides better results than the constituent algorithms all over the run. This indicates the synergy of this combination, thus supporting the idea that this is a profitable approach for tackling phylogenetic inference.

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–[36] has been used for this purpose. Unlike t-test, this test does not assume normality of the samples. In this case, the test indicates that the difference of the hybrid algorithm with respect to the MA is always significant (at the standard 5% significance level).

8 Conclusions

In this paper we have investigated the effects of hybridizing an exact method (i.e., Branch and Bound) and a heuristic technique (i.e., a memetic algorithm) for the reconstruction of phylogenetic trees, a computationally



(a) M877 (b) M971

Figure 15: Evolution of best fitness for the hybrid algorithm, MA and BS for different instances. Curves for the Hybrid and MA are averaged over 25 runs.

Table 4: Results of the hybrid algorithm, MA and BS on the data sets used.

	Hybrid Algorithm			
Problem	best	mean ± std. dev.	worst	median
M420	2.34905	2.34918 ± 0.00028	2.34995	2.34910
M1097	1.01160	1.01177 ± 0.00010	1.01190	1.01170
M877	10.14010	10.14025 ± 0.00025	10.14080	10.14010
M971	4.78780	4.79303 ± 0.00253	4.79920	4.79235
M808	11.51415	11.51685 ± 0.00237	11.52525	11.51610

	Memetic Algorithm			
Problem	best	mean ± std. dev.	worst	median
M420	2.35555	2.37937 ± 0.01516	2.41785	2.37475
M1097	1.01315	1.02676 ± 0.01024	1.04570	1.02660
M877	10.19010	10.25138 ± 0.03557	10.36410	10.24875
M971	4.83480	4.88207 ± 0.03494	4.94110	4.88340
M808	11.70265	11.81517 ± 0.07383	12.03925	11.80730

	Beam Search Algorithm			
Problem	best	mean ± std. dev.	worst	median
M420	2.355350	n.a.	n.a.	n.a.
M1097	1.012050	n.a.	n.a.	n.a.
M877	10.159650	n.a.	n.a.	n.a.
M971	4.820250	n.a.	n.a.	n.a.
M808	11.585500	n.a.	n.a.	n.a.

hard task. Our hybrid model tries to combine the best advantages of both techniques as well as minimize the drawbacks of both methods as stand-alone techniques. The proposal is based on an interleaved collaboration of both approaches where, basically, the MA provides improved bounds that the BnB algorithm can use to purge the problem queue, whereas the BnB guides the MA to look into promising regions of the search space.

A performance analysis conducted on data sets comprising real biological data shows that the hybrid model outperformed each of its constituent techniques, as well as classical agglomerative algorithms and other efficient tree-construction methods. At any rate, it must be noted that the hybrid model represents a scheme whose performance depends on several parameters. To gain a better understanding of the model, we have done a dynamics and sensitivity analysis of the model and provided some guidelines about the rationale behind the setting of the parameters.

There is still room for improvements. For instance, more results could be obtained and analyzed by taking into account another different ways to traverse the search tree in the BnB part, or by considering the possibility of transforming the hybrid algorithm in a complete anytime algorithm (via replacing BS by alternatives models of BnB). Future work will also focus on the parallelization of the MA and BnB, leading to better performance. In this case, a parallelized model would demand a new analysis about parameter selection in order to achieve better understanding of the model.

Acknowledgements

This work was partially supported by Spanish MCyT under contracts TIN2004-7943-C04-01 and TIN2005-08818-C04-01.

References

[1] A.A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002.

[2] H.J. Bandelt. Recognition of tree metrics. *SIAM Journal on Discrete Mathematics*, 3(1):1–6, 1990.

[3] A. Barr and E.A. Feigenbaum. *Handbook of Artificial Intelligence*. Morgan Kaufmann, New York NY, 1981.

[4] M. Binder, D.S. Hibbett, and H.P. Molitoris. Phylogenetic relationships of the marine gasteromycete Nia vibrissa. *Mycologia*, 93:679–688, 2001.

[5] M.J. Brauer, M.T. Holder, L.A. Dries, D.J. Zwickle, P.O. Lewis, and D.M. Hillis. Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. *Molecular Biology and Evolution*, 19(10):1717–1726, 2002.

[6] C.B. Congdon. Gaphyl: A genetic algorithms approach to cladistics. In L. De Raedt and A. Siebes, editors, *5th European Conference on Principles of Data Mining and Knowledge Discovery – PKDD 2001*, volume 2168 of *Lecture Notes in Computer Science*, pages 67–78, Freiburg, Germany, September 2001. Springer-Verlag.

[7] C. Cotta. Memetic algorithms with partial lamarckism for the shortest common supersequence problem. In J. Mira and J.R. Álvarez, editors, *Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach*, volume 3562 of *Lecture Notes in Computer Science*, pages 84–91, Berlin Heidelberg, 2005. Springer-Verlag.

[8] C. Cotta. Scatter search with path relinking for phylogenetic inference. *European Journal of Operational Research*, 169(2):520–532, 2006.

[9] C. Cotta and P. Moscato. Inferring phylogenetic trees using evolutionary algorithms. In J.J. Merelo et al., editors, *Parallel Problem Solving From Nature VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 720–729. Springer-Verlag, Berlin, 2002.

[10] J. Culberson. On the futility of blind search: An algorithmic view of "No Free Lunch". *Evolutionary Computation*, 6(2):109–128, 1998.

[11] W.H.E. Day. Computationally difficult problems in phylogeny systematics. *Journal of Theoretic Biology*, 103:429–438, 1983.

[12] W.H.E. Day. Computational complexity of inferring phylogenies from dissimilarity matrices. *Bulletin of Mathematical Biology*, 49(4):461–467, 1987.

[13] W.H.E. Day, D.S. Johnson, and D. Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences*, 81:33–42, 1986.

[14] J. Felsenstein. Statistical inference of phylogenies (with discussion). *Journal of the Royal Statistical Society A*, 146:246–272, 1983.

[15] W.M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155:279–284, 1967.

[16] G.B. Fogel. Evolutionary computation for the inference of natural evolutionary histories. *IEEE Connections*, 3(1):11–14, 2005.

[17] L.R. Foulds and R.L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:439–49, 1982.

[18] L.M. Giussani, J.H. Cota-Sanchez, F.O. Zuloaga, and E.A. Kellogg. A molecular phylogeny of the grass subfamily Panicoideae (Poaceae) shows multiple origins of C4 photosynthesis. *American Journal of Botany*, 88:1993–2012, 2001.

[19] F. Glover. Scatter search and path relinking. In D. Corne, M. Dorigo, and F. Glover, editors, *New Methods in Optimization*, pages 291–316. McGraw-Hill, London, 1999.

[20] J. Hein. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. *Molecular Biology and Evolution*, 6:649–668, 1989.

[21] D.S. Hibbett and M.J. Donoghue. Analysis of character correlations among wood decay mechanisms, mating systems, and substrate ranges in homobasidiomycetes. *Systematic Biology*, 50:1–27, 2001.

[22] D.S. Hibbett, L.-B. Gilbert, and M.J. Donoghue. Evolutionary instability of ectomycorrhizal symbioses in basidiomycetes. *Nature*, 407:506–508, 2000.

[23] T. Hill, A. Lundgren, R. Fredriksson, and H.B. Schiöth. Genetic algorithm for large-scale maximum parsimony phylogenetic analysis of proteins. *Biochim Biophys Acta*, 1725(1):19–29, August 2005.

[24] S. Holmes. Phylogenies: An overview. In M.E. Halloran and S. Geisser, editors, *Statistics and Genetics*, pages 81–119. Springer-Verlag, New York NY, 1999.

[25] C. Houck, J.A. Joines, M.G. Kay, and J.R. Wilson. Empirical investigation of the benefits of partial lamarckianism. *Evolutionary Computation*, 5(1):31–60, 1997.

[26] U.W. Hwang, M. Friedrich, D. Tautz, C.J. Park, and W. Kim. Mitochondrial protein phylogeny joins myriapods with chelicerates. *Nature*, 413:154–157, 2001.

[27] T.H. Jukes and C.R. Cantor. Evolution of protein molecules. In H.N. Munro, editor, *Mammalian Protein Metabolism*, volume 3, pages 21–132. Academic Press, New York NY, 1969.

[28] K. Katoh, K.-I. Kuma, and T. Miyata. Genetic algorithm-based maximum-likelihood analysis for molecular phylogeny. *Journal of Molecular Evolution*, 53:477–484, 2001.

[29] J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. In T. Lengauer et al., editors, *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, Heidelberg, 1999. The American Association for Artificial Intelligence Press.

[30] Y.-H. Kim, S.-K. Lee, and B.-R. Moon. Optimizing the order of taxon addition in phylogenetic tree construction using genetic algorithm. In E. Cantú-Paz et al., editors, *Genetic and Evolutionary Computation – GECCO 2003*, volume 2724 of *LNCS*, pages 2168–2178, Chicago, 12-16 July 2003. Springer-Verlag.

[31] M. Kimura. Estimation of evolutionary distances between homologous nucleotide sequences. *Proceedings of the Natural Academy of Sciences of the United States of America*, 78:454–458, 1981.

[32] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 69:86–101, 1967.

[33] J.R. Koza. *Genetic Programming.* MIT Press, Cambridge MA, 1992.

[34] M. Laguna and R. Martí. *Scatter Search. Methodology and Implementations in C.* Kluwer Academic Publishers, Boston MA, 2003.

[35] E.L. Lawler and D.E. Wood. Branch and bounds methods: A survey. *Operations Research*, 4(4):669–719, 1966.

[36] E.L. Lehmann. *Nonparametric Statistical Methods Based on Ranks.* McGraw-Hill, New York NY, 1975.

[37] A.R. Lemmon and M.C. Milinkovitch. The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation. *Proceedings of the Natural Academy of Sciences of the United States of America*, 99(16):10516–10521, 2002.

[38] P.O. Lewis. A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. *Molecular Biology and Evolution*, 15(3):277–283, 1998.

[39] W.H. Li. *Molecular Evolution.* Sinauer, Boston, 1997.

[40] J.A. Lozano and P. Larrañaga. Applying genetic algorithms to search for the best hierarchical clustering of a dataset. *Pattern Recognition Letters*, 20(9):911–918, 1999.

[41] H. Matsuda. Construction of phylogenetic trees from amino acid sequences using a genetic algorithm. In *Genome Informatics Workshop*, pages 19–28. Universal Academy Press, December 1995.

[42] H. Matsuda. Protein phylogenetic inference using maximum likelihood with a genetic algorithm. In L. Hunter and T. E. Klein, editors, *1st Pacific Symposium on Biocomputing'96*, pages 512–523, London, January 1996. World Scientific.

[43] A. Meade, D. Corne, M. Pagel, and R. Sibly. Using evolutionary algorithms to estimate transition rates of discrete characteristics in phylogenetic trees. In *2001 Congress on Evolutionary computation (CEC 2001)*, volume 2, pages 1170–1176, Seoul, Korea, May 2001. IEEE.

[44] A. Moilanen. Searching for the most parsimonious trees with simulated evolution. *Cladistics*, 15:39–50, 1999.

[45] P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999.

[46] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers, Boston MA, 2003.

[47] P. Moscato, A. Mendes, and C. Cotta. Memetic algorithms. In G.C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, pages 53–85. Springer-Verlag, Berlin Heidelberg, 2004.

[48] C.-K. Ong, S. Nee, A. Rambaut, H.-U. Bernard, and P.H. Harvey. Elucidating the population histories and transmission dynamics of papillomaviruses using phylogenetic trees. *Journal of Molecular Evolution*, 44:199–206, 1997.

[49] L. Poladian. A GA for maximum likelihood phylogenetic inference using neighbour-joining as a genotype to phenotype mapping. In H.-G. Beyer and U.-M. O'Reilly, editors, *Genetic and Evolutionary Computation Conference GECCO 2005*, pages 415–422, Washington DC, USA, June 2005. ACM.

[50] L. Poladian and L.S. Jermiin. Multi-objective evolutionary algorithms and phylogenetic inference with multiple data sets. *Soft Computing*, 10(4):359–368, 2005.

[51] T. H. Reijmers, R. Wehrens, and L. M. C. Buydens. Quality criteria of genetic algorithms for construction of phylogenetic trees. *Journal of Computational Chemistry*, 20(8):867–876, 1999.

[52] C. C. Ribeiro and D.S. Vianna. A genetic algorithm for the phylogeny problem using an optimized crossover strategy based on path-relinking. In S. Lifschitz et al., editors, *II Brazilian Workshop on Bioinformatics (WOB 2004)*, pages 97–102, Macaé, RJ, Brazil, December 2003.

[53] B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology*, 232:584–599, 1993.

[54] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[55] J. Shen and R.B. Heckendorn. Discrete branch length representations for genetic algorithms in phylogenetic search. In G.R. Raidl et al., editors, *Applications of Evolutionary Computing 2004*, volume 3005 of *Lecture Notes in Computer Science*, pages 94–103, Coimbra, Portugal, April 2004. Springer.

[56] A.N. Skourikhine. Phylogenetic tree reconstruction using self-adaptive genetic algorithm. In N.G. Bourbakis, editor, *1st IEEE International Symposium on Bioinformatics and Biomedical Engineering BIBE 2000*, pages 129–134, Arlington, Virginia, USA, November 2000. IEEE Computer Society.

[57] P.H. Sneath and R.R. Sokal. *Numerical Taxonomy.* Freeman, London, UK, 1973.

[58] R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *University Kansas Science Bulletin*, 38:1409–1438, 1958.

[59] M.F. Whiting, J.C. Carpenter, Q.D. Wheeler, and W.C. Wheeler. The strepsiptera problem: phylogeny of the holometabolous insect orders inferred from 18S and 28S ribosomal DNA sequences and morphology. *Systematic Biology*, 46(1):1–68, 1997.

[60] P. Wiston. *Artificial Intelligence.* Addison-Wesley, Reading MA, 1984.

[61] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[62] B.Y. Wu, K.-M. Chao, and C.Y. Tang. Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices. *Journal of Combinatorial Optimization*, 3(2):199–211, 1999.