Inferring Phylogenetic Trees Using Evolutionary Algorithms

Carlos Cotta[1] and Pablo Moscato[2]

[1] Dept. Lenguajes y Ciencias de la Computación, ETSI Informática,
University of Málaga, Campus de Teatinos, 29071 - Málaga - SPAIN
ccottap@lcc.uma.es

[2] Grupo de Engenharia de Computação em Sistemas Complexos,
Dept. de Engenharia de Computação e Automação Industrial,
Universidade Estadual de Campinas, C.P. 6101,
Campinas, SP, CEP 13083-970, Brazil
moscato@dca.fee.unicamp.br

Abstract. We consider the problem of estimating the evolutionary history of a collection of organisms in terms of a phylogenetic tree. This is a hard combinatorial optimization problem for which different EA approaches are proposed and evaluated. Using two problem instances of different sizes, it is shown that an EA that directly encodes trees and uses *ad-hoc* operators performs better than several decoder-based EAs, but does not scale well with the problem size. A greedy-decoder EA provides the overall best results, achieving near 100%-success at a lower computational cost than the remaining approaches.

1 Introduction

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. An accurate estimation of this evolutionary history is a very useful tool in many areas of Biology, such as multiple sequence alignment [4], or molecular epidemiological studies of viruses [11] among others.

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. Unfortunately, this constitutes a very hard combinatorial optimization problem for most optimality criteria. Exact techniques such as branch-and-bound can be used, but can be computationally unaffordable for even moderate (say, 30-40 taxa) problem instances. Hence, the use of heuristic techniques seems appropriate.

We are concerned in this work about the utilization of evolutionary algorithms (EAs) for tackling the Phylogeny Problem. In this sense, we have initially focused on distance-based measures (Section 2 will provide details about this and other quality measures, as well as about phylogenetic trees in general). From this

starting point, several EA approaches –based on the use of different representations and/or reproductive operators– have been devised and compared. More precisely, both EAs that directly conduct the search in the space of phylogenetic trees, and EAs that use auxiliary search spaces and decoders have been considered. These EAs are detailed in Section 3. Subsequently, the results of a thorough empirical comparison are reported in Section 4. We conclude by presenting a summary of our conclusions, and outlining future work in Section 5.

2 A Gentle Introduction to Phylogenetic Trees

Assume we are given some molecular-sequence data for a collection S of n taxa. A phylogenetic tree T is a tree with exactly n leaves, each one labeled by a taxon in S. Internal nodes in this tree correspond to hypothetical ancestral organisms, and edges in T represent ancestry-descent relationships. Taxa in S are also termed OTUs (*operational taxonomic units*), while internal nodes are termed HTU (*hypothetical taxonomic units*). Thus, a phylogenetic tree models the evolutionary history of the OTUs, back to their common ancestor (the root of the tree[1]). In the following, we will denote OTUs and HTUs with lowercase letters, and trees with uppercase letters. A LISP-like notation will be used to represent trees, e.g., (hTU) represents the tree rooted at h, with T and U as subtrees[2], and (o) represents a leaf labeled with o.

The goal of the Phylogeny Problem is finding the phylogenetic tree T that best resembles the evolutionary history of OTUs in S. For this purpose, it is clearly necessary to define an optimality criterion. Essentially, such a criterion (and subsequently an inference method) can fall within two major categories, *sequence*-based and *distance*-based. In sequence-based approaches, each node of T is assigned a sequence (known for OTUs, and inferred via pairwise alignments for HTUs). Then, the tree is evaluated using a criterion that –in most situations– is either *maximum likelihood* (ML) or *maximum parsimony* (MP). In the former, a stochastic model of evolution (e.g., the Jukes-Cantor model) is used in order to assess the likelihood that the current tree generated the observed data. 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 OTUs i and j. Then, we have an *observed* distance matrix M, and an *inferred* distance matrix $\hat{M}$ (obtained by making $\hat{M}_{ij}$ = distance from i to j in T). The quality of the tree can now be quantified in a variety of ways. Firstly, it is possible to consider some "distance" measure between M and $\hat{M}$;

[1] Biologists often focus on a relaxed model based on *unrooted* trees as well. In this work we will concentrate on rooted trees though. See [6, 8] for heuristic approaches to the inference of unrooted trees.

[2] Non-binary trees are also possible, but they can be easily reduced to binary ones.

usual examples are the $STRESS$ measure (normalized sum of absolute differences), the L_2 metric (least-squares approximation), or the L_∞ metric (minimize maximum absolute difference). Secondly, quality can be directly measured from T. This is typically the case when edge-weighting has been constrained so as to have $\hat{M}_{ij} \geq M_{ij}$; in this situation, minimizing the total weight of T is usually the criterion.

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 [5]. For these reasons, we have focused in distance-based approaches in this work. To be precise, we have considered the constraint mentioned above, regarding inferred dissimilarities to be greater or equal than observed ones. This is done by forcing $\hat{M}$ to be ultrametric (see [14] for details about ultrametricity); very popular when the molecular-clock hypothesis was in vogue, this condition provides a very good approximation to the optimal solution under more relaxed assumptions (e.g., mere additivity). Furthermore, it has allowed us using exact techniques in order to estimate the absolute quality of the solutions achieved by the different EAs.

3 Evolutionary Approaches to the Phylogeny Problem

In essence, two main approaches can be considered for tackling the Phylogeny Problem with EAs. The first one is the *direct* approach, in which the EA conducts the search in the space $\mathcal{S}_{Ph}$ of all possible phylogenetic trees. The second one is the *indirect* approach, in which an auxiliary $\mathcal{S}_{aux}$ space is used by the EA. In this latter case, a decoder [7] must be utilized in order to perform the $\mathcal{S}_{aux} \longrightarrow \mathcal{S}_{Ph}$ mapping. Either of these approaches requires defining adequate reproductive operators and/or representation schemes. These are described below.

3.1 Direct Search in the Phylogenetic-Tree Space

As mentioned above, a direct approach is characterized by performing the search in the space of all possible phylogenetic trees. Thus, each individual in the EA population directly represents a feasible tree. For this purpose, any of the encoding techniques commonly used in genetic programming (GP) –e.g., LISP-like expressions, preorder traversals, etc.– can be used. Subsequently, appropriate operators must be designed to manipulate this representation.

First of all, consider the recombination operator. This operator must take information pieces from both parents, and combine them to create some offspring. Since individuals directly encode trees in this case, these information pieces naturally emerge in the form of subtrees. Thus, recombination can be expressed in terms of pruning and grafting subtrees, much like it is typically done in GP. However, unlike the most classical GP scenario, subtrees cannot be randomly shuffled, since phylogenetic trees are constrained to have n leaves, each one representing a different OTU. Hence, a slightly different approach must be

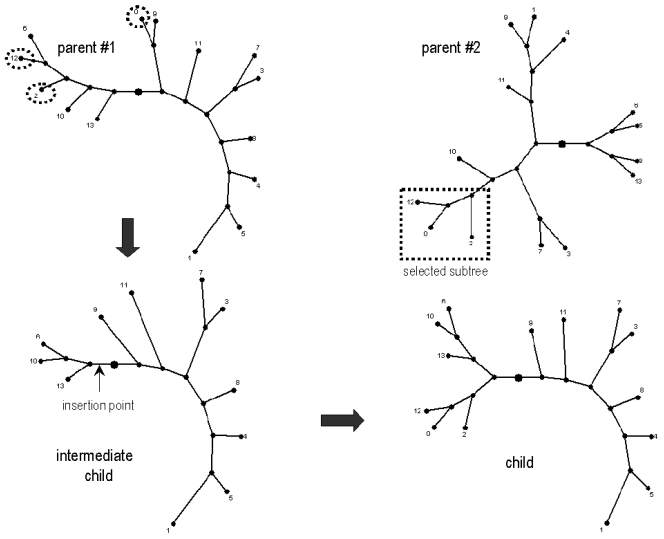

Fig. 1. Functioning of the PDG recombination. A subtree is selected in one of the parents, and inserted at a random point of the other parent, right after having deleted duplicates nodes. The root of the tree is marked with a thick node.

considered. To be precise, the recombination operator must take care of removing duplicates elements before attempting to regraft the selected subtree. Let T_1 and T_2 be the trees being recombined; the whole process would be as follows:

Prune-Delete-Graft Recombination (T_1, T_2)

1. Select a subtree T from T_2.
2. **for each** OTU $o \in T$ **do**
   (a) Find subtree U from T_1 such that $U = (h(o)U')$ or $U = (hU'(o))$.
   (b) Replace U by U' in T_1.
3. Select a random subtree V from T_1.
4. Replace V by $V' = (h'TV)$, where h' is a new HTU.

Figure 1 illustrates the process. This PDG operator has been used by Moilanen [10] in the context of a sequence-based parsimony measure.

As to mutation operators, several options are possible. The following ones have been considered:

- SWAP: two OTUs are selected at random, and their positions are swapped.
- NNI: consider the sequence of leaves $L[T]$ defined as follows:

$$L[(hUV)] = L[U] : L[V]; \qquad L[(o)] = \langle o \rangle; \tag{1}$$

where : is the sequence-concatenation operator. Then, this operator swaps two neighboring leaves in this sequence.

– SCRAMBLE: first, a subtree T' is randomly selected from T. Then, its topology is rearranged at random.

All these mutation operators fulfill the previously-mentioned constraint regarding the presence of exactly n leaves/OTUs in the tree. Notice also that NNI can be seen as a particular case of SWAP. The basic idea here is that a small reordering or nearby leaves (corresponding to closely related species) is more likely to result in an improvement than a larger reordering (this is the case for SWAP, that can be considered somewhat disruptive in this sense). In addition to this, the version of NNI considered in this work checks whether the interchange provides a better –according to the fitness function– tree-structure, and reverts the interchange if this were not the case.

3.2 Decoder-based EAs for the Phylogeny Problem

As an alternative to directly conducting the search in the solution space, a combination of an auxiliary search space $\mathcal{S}_{aux}$ and a decoder D can be used. This approach is very rich in possibilities, and has several advantages. On one hand, it usually allows utilizing simpler evolutionary operators, due to the fact that $\mathcal{S}_{aux}$ is often an unconstrained search space (unlike $\mathcal{S}_{Ph}$). On the other hand, problem knowledge can be introduced by means of D. Among the potential drawbacks of this approach, one can cite the difficulty in some situations for exploring $\mathcal{S}_{aux}$ in a parsimonious way; the use of a decoder can hinder finding an acceptable notion of locality within $\mathcal{S}_{aux}$.

Basic Setting The first decoder approach considered resembles in some sense the ordinal (stack-based) representation of the TSP [9]. In this case, OTUs are assumed to be ordered in some sense. Let $S = \langle o_1, \cdots, o_n \rangle$ be the ordered sequence of OTUs[3]. The auxiliary search space $\mathcal{S}_{aux}$ is defined as $\mathcal{S}_{aux} = \prod_{i=1}^{n-2} \mathbb{N}_{2i}$, where $\mathbb{N}_k = \{1, \cdots, k\}$. The decoding process of an individual $s \in \mathcal{S}_{aux}$ is as follows:

ORDINAL DECODER(s)
1. Let $T = (h(o_1)(o_2))$
2. **for each** $i \in [1 : n-2]$ **do**
   (a) Let the branches of T be numbered from 1 to $2i$. Let branch s_i join node h and subtree U.
   (b) Replace U by $V = (h'(o_{i+2})U)$ in T, where h' is a new HTU.

As it can be seen, in a phylogenetic tree with k leaves, there exist $2k-1$ nodes, and hence $2k-2$ branches. Each element in s indicates in which of these branches the next OTU in the sequence must be inserted. Thus, the ith element of s ranges from 1 to $2i$. One of the nicest properties of this ordinal representation is its orthogonality [12]. Plainly, this means that any s is feasible as long as each s_i is in its corresponding range. Hence, any positional recombination operator

[3] In this work, a *maxmin* sequence [14] has been considered: the two first elements are those whose distance in M is maximal; each subsequent element o_i ($i > 2$) is the one for which $d(o_i) = \min\{M_{o_i,o_j} \mid j < i\}$ is maximal.

such as single-point crossover (SPX), or uniform crossover (UX) will produce a feasible child when applied to feasible parents.

One of the weak points of the ordinal decoder presented above can be found in the use of a fixed sequence: it may be harder to construct good solutions for a certain problem instance using sequence A than using sequence B. This admits two possible solutions. First, a smart method for constructing an appropriate OTU sequence given a certain problem instance could be devised. Alternatively, the EA can make this sequence evolve, along with the ordinal insertion points. In this latter case, the search space is $\mathcal{S}'_{aux} = \mathcal{S}_{aux} \times \mathbb{P}_n$, where $\mathcal{S}_{aux}$ is the space of ordinal sequences described above, and $\mathbb{P}_n$ is the space of n-element permutations. The decoding process would be identical as indicated above, with the sole difference that the OTU sequence would be taken from s as well. The same considerations regarding the use of standard positional operators are applicable in this case too. Additionally, it must be noted that recombination can also be done on the permutation segment of individuals, using any standard permutational operator for this purpose (e.g., OX, UCX [3], etc.)[4].

Using Guidance The above decoders are essentially blind, i.e., they do not use any phenotypic information in order to guide the construction process; they take all the information they need from the decoder input s. It is reasonable to consider the use of some kind of guidance information though. Two main possibilities are considered here: using this guidance during the decoding stage, or when recombining.

Starting with this latter one, note that recombination is given two individuals s^1 and s^2; rather than simply mixing information from these two individuals in order to create the child, it might be useful to get some assessment on the quality of the partially constructed solution in order to guide the process. A simple way of doing this is following a greedy approach:

GREEDY-INSERTION XOVER(s^1, s^2)
1. Let $T = (h(o_1)(o_2))$
2. **for each** $i \in [1 : n-2]$ **do**
   (a) Let the branches of T be numbered from 1 to $2i$. Let branch s^1_i (resp. s^2_i) join node h_1 and subtree U_1 (resp. h_2 and U_2).
   (b) Let $T_1, T_2 = T$. Replace U_1 by $V = (h'(o_{i+2})U_1)$ in T_1. Act analogously with U_2 in T_2.
   (c) Let T=best(T_1, T_2).

In the above description, the OTU sequence is not necessarily fixed. Actually, it can be taken from either of the parents, or constructed by recombining (using a standard operator) the parental sequences. Notice also that a symmetric approach can be defined, i.e., assuming a certain insertion sequence, and taking greedy decisions on the structure of the OTU sequence. In this case, the basic units upon which decisions are taken cannot be single OTUs, since the positional

[4] The notation O_1/O_2 will be used to denote that one of $\{O_1,O_2\}$ is applied on an individual, while O_1+O_2 will denote that both operators are sequentially applied.

representation of permutations is not orthogonal. On the contrary, *blocks*[5] must be considered. Then,

GREEDY-ORDER XOVER(o^1, o^2)

1. Identify block structure in o^1, o^2. Let $B_1^{1|2}, \cdots, B_m^{1|2}$ be the blocks.
2. Let $T_1 = (h(o_1^1)(o_2^1))$. Add remaining OTUs in B_1^1 (do the same with T_2 and B_1^2).
3. Let T=best(T_1, T_2).
4. **for each** $j \in [2 : m]$ **do**
   (a) Let $T_1 = T$; **for each** OTU $o_k^1 \in B_j^1$ **do**
      - Insert OTU o_k^1 in branch s_k.

   (b) Act analogously with T_2 and B_j^2.
   (c) Let T=best(T_1, T_2).

This version of the recombination operator has the advantage that requires fewer assessments of partial solutions, since OTUs are inserted in blocks rather than one at a time. Again, the insertion sequence can be fixed, taken from one of the parents, or obtained by recombination.

Finally, consider a similar approach to those above, but focused on the decoder stage rather than on recombination. In this case, the OTU sequence is given, but the insertion sequence is not; the decoder is responsible for finding adequate insertion points for each OTU. The process could be as follows:

PERMUTATIONAL DECODER($\langle o_1, \cdots, o_n \rangle$)

1. Let $T = (h(o_1)(o_2))$
2. **for each** $j \in [1 : n-2]$ **do**
   (a) **for each** insertion point $i \in \mathbb{N}_{2i}$ **do**
      - Let $T_i = T$. Insert OTU o_{j+2} in branch i.

   (b) Let T=best($T_1, \cdots, T_{2i}$).

Next section will be devoted to provide empirical evidence regarding the potential usefulness of this and all previous approaches.

4 Empirical Results

The experiments have been done with an elitist generational EA ($popsize = 100$, $p_c = .9$, $p_m = 0.01$) using linear ranking selection ($\eta = 2.0$). No fine tuning of these standard parameters was attempted. A maximum number of 10^6 evaluations has been enforced. In order to provide a fair comparison, the internal assessments of partial solutions performed by some operators and decoders have been accounted as well.

Two problem instances with 20 and 34 OTUs respectively have been considered [1, 13]. These instances have been obtained by using conditional Kolmogorov complexity to calculate inter-OTU distances[6] [2] from mtDNA sequences. A

[5] A block is a compact subsequence of elements such that both parents have the same elements in this segment, although in a possibly-different order (see [3]).

[6] $M_{ij} = 1 - \frac{K(i) - K(i|j)}{K(ij)}$

Table 1. Results for the 20-OTU instance (averaged for 50 runs).

Algorithm		best	mean ± std.dev.	%success	#evals
PDG	SWAP	8.290368	8.290368 ± 0.000000	100%	246,092
	NNI	8.290368	8.290368 ± 0.000000	100%	85,435
	SCRAMBLE	8.290368	8.290374 ± 0.000042	98%	146,162
Ordinal	SPX	8.337564	8.497876 ± 0.096093	0%	–
	DPX	8.325214	8.466251 ± 0.081407	0%	–
	UX	8.345291	8.471477 ± 0.085760	0%	–
Adaptive ordinal	SPX / OX	8.310436	8.417089 ± 0.066013	0%	–
	SPX / UCX	8.303018	8.400509 ± 0.047586	0%	–
	SPX + OX	8.294011	8.413814 ± 0.067423	0%	–
	SPX + UCX	8.304323	8.409855 ± 0.071146	0%	–
	DPX / OX	8.305980	8.395530 ± 0.064806	0%	–
	DPX / UCX	8.302258	8.415585 ± 0.072114	0%	–
	DPX + OX	8.295929	8.417998 ± 0.064371	0%	–
	DPX + UCX	8.331137	8.416023 ± 0.061231	0%	–
	UX / OX	8.295403	8.409900 ± 0.077820	0%	–
	UX / UCX	8.294327	8.394244 ± 0.059192	0%	–
	UX + OX	8.305803	8.409900 ± 0.077820	0%	–
	UX + UCX	8.290684	8.397843 ± 0.057408	0%	–
Greedy Xover	*insertion*	8.341772	8.481938 ± 0.072921	0%	–
	order	8.299083	8.453090 ± 0.088807	0%	–
	UCX+*insertion*	8.350527	8.460593 ± 0.078182	0%	–
	UX+*order*	8.320109	8.407623 ± 0.052265	0%	–
Permutational Decoder		8.290368	8.290368 ± 0.000000	100%	23,370

branch-and-bound algorithm following [14] has been implemented, so as to know the exact optimal solutions. While the 20-OTU instance could be solved rather efficiently (a couple of seconds on a DIGITAL Alpha 400), the 34-OTU instance revealed itself as much harder to solve; it took about six hours and a half on the same machine, and more than half a billion subproblems were evaluated.

The results for the 20-OTU instance are shown in Table 1. The %-success column indicates the number of times the optimal solution is found, and #evals is the mean number of evaluations required in these successful runs. Notice firstly the good results provided by the PDG operator; near 100%-success rates are achieved in combination with any of the mutation operators discussed. While the absolute goodness of these results can be due to the low difficulty of this instance, its relative superiority over most decoder-based approaches is still informative. As it can be seen, the ordinal representation does not manage to find the optimal solution either with SPX, DPX (double-point crossover) or UX. When the OTU sequence is evolved along with the insertion points, performance is clearly improved, although no optimal solution is found. The results for the guided crossover are not satisfactory either. The reason may lie in the high cost of evaluating partial solutions. Actually, the greedy-order crossover yields slightly better results, due to the fact that fewer internal evaluations are needed. Finally, the results for the permutational decoder (using UCX) are the overall best. Unlike the plain decoding of the previous EAs, the greedy decoding is less

Table 2. Results for the 34-OTU instance (averaged for 50 runs).

Algorithm		best	mean ± std.dev.	%success	#evals
PDG	SWAP	14.855763	14.860250 ± 0.002611	2%	933,639
	NNI	14.855763	14.857864 ± 0.001827	18%	427,468
	SCRAMBLE	14.855763	14.858426 ± 0.002104	10%	722,502
Ordinal	SPX	14.959411	15.100436 ± 0.079702	0%	–
	DPX	14.932403	15.141366 ± 0.093397	0%	–
	UX	14.939449	15.112828 ± 0.089895	0%	–
Adaptive ordinal	SPX / OX	14.904175	15.032073 ± 0.060582	0%	–
	SPX / UCX	14.920124	15.034428 ± 0.066467	0%	–
	SPX + OX	14.930042	15.042107 ± 0.074107	0%	–
	SPX + UCX	14.905937	15.031634 ± 0.071966	0%	–
	DPX / OX	14.920697	15.048883 ± 0.080941	0%	–
	DPX / UCX	14.898218	15.042495 ± 0.097559	0%	–
	DPX + OX	14.897718	15.016650 ± 0.073133	0%	–
	DPX + UCX	14.924216	15.043213 ± 0.074294	0%	–
	UX / OX	14.924170	15.050490 ± 0.072886	0%	–
	UX / UCX	14.896434	15.044002 ± 0.071149	0%	–
	UX + OX	14.915946	15.043606 ± 0.077418	0%	–
	UX + UCX	14.923092	15.035905 ± 0.077034	0%	–
Greedy XOver	*insertion*	14.954607	15.127468 ± 0.092056	0%	–
	order	14.951596	15.080942 ± 0.072010	0%	–
	UCX+*insertion*	14.957719	15.107558 ± 0.092509	0%	–
	UX+*order*	14.918159	15.031375 ± 0.067859	0%	–
Permutational Decoder		14.855763	14.855772 ± 0.000049	96%	397,512

sensitive to the disruptive effect that genotypic recombination can have on the phenotype. Hence, it manages to find the optimal solution in 100% of the runs, using a lower number of evaluations (including internal calculations) than PDG with SWAP, NNI, or SCRAMBLE.

The results for the 34-OTU instance (Table 2) are completely consistent with this analysis. Again, PDG-based EAs perform better than EAs using ordinal decoders or greedy crossover. However, their success rate is notably lower here, due to the higher difficulty of the instance. The performance of the permutational-decoder EA is only marginally affected though. Optimal solutions are found on a regular basis (96%-success) at a lower computational cost than EAs using PDG.

5 Conclusions

A number of EAs for solving the Phylogeny Problem have been developed and compared. An empirical evaluation of these EAs using distance-based measures has shown that directly evolving phylogenetic trees yields better results than indirect approaches using decoders. A notable exception to this rule is provided by the greedy permutational decoder. This approach consistently provides optimal solutions at a lower cost than PDG-based EAs. Moreover, these latter EAs suffer from a scalability problem, exhibiting a clear performance drop when the number of OTUs increases. This does not seem to be the case for the permutational greedy decoder, at least for the instance sizes considered in this work.

Future work will try to confirm these results on larger problem instances. Optimal solutions will not be available in this case, but qualitative assessments will still be possible. Work is in progress here. The use of different evaluation measures is also an interesting line for future developments. In this sense, we plan to test alternative distance-based criteria, as well as maximum-likelihood approaches.

Acknowledgements The first author is partially supported by Spanish CICYT under grant TIC1999-0754-C03. The second author is supported by Brazilian CNPq under Proj. 52.1100/01-1.

References

1. Y. Cao, N. Okada, and M. Hasegawa. Phylogenetic position of guinea pigs revisited. *Molecular Biology and Evolution*, 14(4):461–464, 1997.
2. X. Chen, S. Kwong, and M. Li. A compression algorithm for DNA sequences and its application in genome comparisons. *Genome Informatics*, 10:51–61, 1999.
3. C. Cotta and J.M. Troya. Genetic forma recombination in permutation flowshop problems. *Evolutionary Computation*, 6(1):25–44, 1998.
4. 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.
5. S. Holmes. Phylogenies: An overview. In Halloran and Geisser, editors, *Statistics and Genetics*, pages 81–119. Springer-Verlag, New York NY, 1999.
6. D. Huson, S. Nettles, and T. Warnow. Disk-covering, a fast converging method for phylogenetic tree reconstruction. *Journal of Computational Biology*, 6(3):369–386, 1999.
7. S Koziel and Z. Michalewicz. A decoder-based evolutionary algorithm for constrained parameter optimization problems. In T. Bäeck, A.E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature V – LNCS 1498*, pages 231–240. Springer-Verlag, Berlin Heidelberg, 1998.
8. H. Matsuda. Protein phylogenetic inference using maximum likelihood with a genetic algorithm. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 512–523. World Scientific, 1996.
9. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 1992.
10. A. Moilanen. Searching for the most parsimonious trees with simulated evolution. *Cladistics*, 15:39–50, 1999.
11. 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.
12. N.J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–205, 1991.
13. A. Reyes, C. Gissi, G. Pesole, F.M. Catzeflis, and C. Saccone. Where do rodents fit? Evidence from the complete genome of *Sciurus vulgaris*. *Molecular Biology and Evolution*, 17(6):979–983, 2000.
14. Bang Ye Wu, Kun-Mao Chao, and Chuan Yi Tang. Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices. *Journal of Combinatorial Optimization*, 3(2):199–211, 1999.