# On the Application of Evolutionary Algorithms to the Consensus Tree Problem

Carlos Cotta

Dept. Lenguajes y Ciencias de la Computación, University of Málaga,
ETSI Informática, Campus de Teatinos, 29071 - Málaga, Spain

ccottap@lcc.uma.es

**Abstract.** Computing consensus trees amounts to finding a single tree that summarizes a collection of trees. Three evolutionary algorithms are defined for this problem, featuring characteristics of genetic programming (GP), evolution strategies (ES) and evolutionary programming (EP) respectively. These algorithms are evaluated on a benchmark composed of phylogenetic trees computed from genomic data. The GP-like algorithm is shown to provide better results than the other evolutionary algorithms, and than two greedy heuristics defined *ad hoc* for this problem.

## 1 Introduction

Trees are ubiquitous data structures. They appear in diverse domains such as information retrieval [1], scheduling [2], computer graphics [3], and bioinformatics [4] among others. In all these cases, there exists the need of representing data in a hierarchical fashion, and hence the use of trees. Unfortunately, it is generally the case that finding or constructing the optimal tree for one of these applications is a very hard problem. Consider for example the inference of phylogenetic trees, a problem from the bioinformatics domain. This problem seeks a tree representing the evolutionary history of a collection of species. This is typically done on the basis of molecular information –e.g., DNA sequences– from these species, and can be approached in a number of ways: maximum likelihood, parsimony, distance matrices, etc. [5]. The problem is $NP-$hard under most models [6–8].

Hardness results motivate heuristic approaches for finding near-optimal trees. Sticking with the phylogeny problem, both greedy heuristics [9] and metaheuristics [10–13] have been used. A number of different high-quality trees can be found, each possibly telling something about the *true* solution. Furthermore, the fact that data come from biological experiments, which are not exact, makes near-optimal solutions (even near-optimal with respect to different criteria) be almost as relevant as the actual optimum. It is in this situation where the consensus tree problem comes into play. Essentially, a consensus method tries to summarize a collection of trees provided as input, returning a single tree [14]. This implies identifying common substructures in the input trees and representing these in the output tree.

Consensus trees are extremely important in many domains. For example, in phylogenetic inference, it has been observed that independently derived trees are

unlikely to have spurious clades (or clusters) in common [15]. Thus, the clades appearing in most or all the input trees can be considered reliable. Unfortunately again, finding consensus trees is also a hard problem in general (see e.g. [16].) The use of heuristic techniques is thus in order.

We consider the use of several evolutionary algorithms (EAs) for constructing consensus trees. These evolutionary algorithms differ in the operator set and in the evolution model, and will be compared on a benchmark from the phylogeny domain. The comparison will also include two greedy heuristics defined *ad hoc* for this problem.

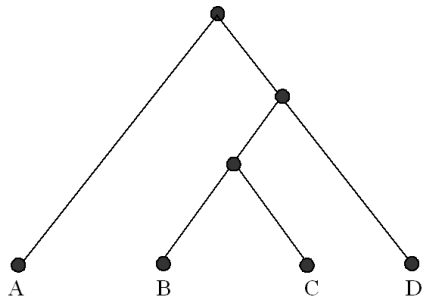
## 2 Background on Consensus Methods

Let $T$ be a strictly binary rooted tree; a LISP-like notation will be used to denote the structure of the tree. Thus, $(sLR)$ is the tree with root $s$, and with $L$ and $R$ as subtrees, and $()$ is an empty tree. The notation $(a)$ is a shortcut for $(a()())$. Let $\mathcal{L}(T)$ be the set of leaves of $T$. Each edge $e$ in $T$ defines a bipartition $\pi_T(e) = \langle S_1, S_2 \rangle$, where $S_1$ are the leaves in $\mathcal{L}(T)$ that can be reached from the root passing through $e$, and $S_2$ are the remaining leaves. We define $\Pi(T) = \{\pi_T(e) \mid e \in T\}$.

There is a variety of consensus methods defined in the literature, differing on the characteristics of the input, and on the output sought. As mentioned before, we concentrate here on trying to represent a collection of trees $\{T_1, \cdots, T_m\}$ as a single tree over $\cup_{i=1}^{m} \mathcal{L}(T_i)$. This can be approached in several ways, such as the tree compatibility problem [17], strict consensus [18], and the median tree [19], among others (see also [20].) While the two first models focus on finding a tree such that $\Pi(T) = \cup_{i=1}^{m} \Pi(T_i)$ (resp. $\Pi(T) = \cap_{i=1}^{m} \Pi(T_i)$), the median tree tries to minimize the sum of differences between $T$ and the input trees, i.e., $\min \sum_{i=1}^{m} d(T, T_i)$. Typically, the distance $d(T, T')$ between trees is defined as the number of non-common bipartitions in $\Pi(T)$ and $\Pi(T')$. This is also termed the partition metric.

The partition metric has several drawbacks. For example, two trees differing solely in the position of one leaf can be maximally different [21]. Alternative metrics can be considered. For example, one can cite the nearest neighbor interchange (NNI), subtree prune and regraft (SPR), and tree bisection and reconnection (TBR) (see [22].) These metrics have their own drawbacks though: computing the NNI metric or the TBR metric is $NP-$hard [22, 23]; the complexity of computing SPR is unknown, but it is conjectured to be $NP-$hard as well.

We employ, then, an alternative metric called TreeRank [24]. Given trees $T$ and $T'$, the TreeRank score provides a measure of the topological relationships in $T$ that are found to be the same or similar in $T'$. This is done with the help of an auxiliary data structure termed the *UpDown matrix*. This structure consists of a pair of matrices for each tree $T$, the *Up matrix* $U_T$, and the *Down matrix* $D_T$. These matrices are intended to capture information on the hierarchical structure of a tree. More precisely, they store the number of edges that must be traversed




*Up Matrix*

| | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 1 | 1 | 1 |
| B | 3 | 0 | 1 | 2 |
| C | 3 | 1 | 0 | 2 |
| D | 2 | 1 | 1 | 0 |

**Fig. 1.** A tree and its Up matrix.

upwards or downwards in order to go from a certain leaf to another one (see Fig. 1.) Formally,

$$\forall u \in L \; \forall v \in R \; \begin{cases} U_{(sLR)}[u,v] = 1 + depth(u,L) \\ U_{(sLR)}[v,u] = 1 + depth(v,R) \end{cases} \tag{1}$$

where $depth(x,T)$ is the depth of node $x$ in tree $T$ (i.e., the length of the path from the root of $T$ to $x$.) Notice that $U[u,u] = 0$, and $U[u,v] = D[v,u]$ (hence it suffices using just one of the matrices to have complete information on the hierarchical structure.) Now, the TreeRank score is computed as[1]:

$$TreeRank(T,T') = \left(1 - \frac{UpDownDist(T,T')}{\sum_{u,v \in \mathcal{L}(T)} U_T[u,v]}\right) \times 100\% \tag{2}$$

where

$$UpDownDist(T,T') = \sum_{u,v \in \mathcal{L}(T)} |U_T[u,v] - |U_{T'}[u,v]| \tag{3}$$

The UpDown matrix can be calculated in $O(|\mathcal{L}|^2)$ as shown in [24]. Notice that if $T$ and $T'$ are identical, then the TreeRank score is 100%. If no common substructure exists in $T$ and $T'$, then the TreeRank score will be near 0%, or even negative if the hierarchical structure is "reversed" in both trees. This TreeRank measure is currently being used in TreeBASE[2] –one of the most widely used phylogenetic databases– for the purposes of handling queries for similar trees.

## 3 Heuristics for the Consensus Tree Problem

We describe two greedy and three evolutionary heuristics for the consensus tree problem.

[1] We have adapted this definition of TreeRank to the fact that $\mathcal{L}(T) = \mathcal{L}(T')$. See [24] for a general formulation in the case that $\mathcal{L}(T) \neq \mathcal{L}(T')$.

[2] `http://www.treebase.org`

### 3.1 Greedy Heuristics

The most typical heuristics for tree construction are essentially greedy (see e.g. [25].) In this sense, we firstly consider a greedy heuristic that constructs a consensus tree incrementally, adding one leaf at a time. This is done by testing all possible insertion points within the partially-constructed tree, and retaining the position that yields the best value of the TreeRank distance. This algorithm is denoted by H1, and its pseudocode is shown below:

**Heuristic** H1 ($\{T_1 \cdots, T_m\}$, $\langle o_1, \cdots, o_n \rangle$)
1. Let $A \leftarrow (h(o_1)(o_2))$
2. **for each** $j \in \mathbb{N}_{n-2}$ **do**
   (a) Let the branches of $A$ be numbered from 1 to $2j$.
   (b) **for each** insertion point $i \in \mathbb{N}_{2j}$ **do**
      i. Let $A_i \leftarrow A$
      ii. Insert leaf $o_{j+2}$ in branch $i$ of $A_i$.
   (c) Let $A \leftarrow \arg\min \{\sum_{i=1}^{m} TreeRank(A', T_i) \mid A' \in \{A_1, \cdots, A_{2j}\}\}$.
3. **return** $A$.

In this definition, $\mathbb{N}_k$ stands for the natural numbers in $[1..k]$. As it can be seen, heuristic H1 is fed with both the collection of trees whose consensus is sought, and with a permutation of the leaves that determines the order in which they will be inserted in the tree. The two first leaves in the permutation are used to construct the initial subtree.

The second greedy heuristic we consider is a variant of the previous one. It is also close in spirit to an approximation algorithm that was devised by Phillips and Tarnow [16] for the asymmetric median tree problem. The main idea of this heuristic –that we denote by H2– is to consider all possible pairs of trees in the target collection, constructing a greedy consensus tree for each of these pairs. This can be done on the basis of the H1 heuristic described before, as shown in the following pseudocode:

**Heuristic** H2 ($\{T_1 \cdots, T_m\}$, $\langle o_1, \cdots, o_n \rangle$)
1. Let $best \leftarrow -\infty$.
2. **for each** $i, j \in \mathbb{N}_m$, $i < j$ **do**
   (a) Let $A \leftarrow$H1($\{T_i, T_j\}, \langle o_1, \cdots, o_n \rangle$).
   (b) Let $d \leftarrow \sum_{k=1}^{m} TreeRank(A, T_k)$.
   (c) **if** $d > best$ **then**
      i. Let $best \leftarrow d$.
      ii. Let $T \leftarrow A$.
3. **return** $T$.

This algorithm admits a minor variant in which the leaf sequence is not the same for all pairs of trees, but separately computed for each of these pairs. This is precisely the version of H2 that we have considered in the experimentation, as it will be described in Sect. 4.

### 3.2 Evolutionary Heuristics

When using an EA for evolving a non-trivial data structure such as a tree, there exist two main alternatives [26, 27]: either perform the search directly on the space of all $n-$leaf trees, or conducting the search in an auxiliary space, using a decoder in order to construct the actual trees. In this work, the evolutionary heuristics considered are all defined in the line of the former approach, i.e., each individual in the EA population directly represents a tentative consensus tree. Given this choice of representation, appropriate operators for recombination and mutation must be defined.

First of all, consider the recombination operator. This operator must take information chunks from two parents, and combine them to create a descendant. In this case, these information chunks can be naturally expressed as subtrees. Hence, the recombination process can be approached much like it is typically done in genetic programming (GP), in terms of pruning and grafting subtrees. An important consideration must be nevertheless made: all trees must have all $n$ different leaves, with neither repetitions nor omissions. This means that when attempting to transfer a subtree $T'$ from a parent to another, all leaves in $\mathcal{L}(T')$ must be deleted in advance from the latter parent. Let $A_1$ and $A_2$ be the trees being recombined; the whole process would be then as follows (cf. [10, 12]):

**Operator** Prune-Delete-Graft Recombination $(T_1, T_2)$
1. Select a subtree $T$ from $T_2$.
2. **for each** leaf $o \in \mathcal{L}(T)$ **do**
   (a) Find subtree $U$ in $T_1$ 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 $V' = (h'TV)$ in $T_1$, where $h'$ is a new internal node.

Notice that the fact that the set of leaves is the same for all trees during all the run, makes in principle the mutation operator be dispensable: topological diversity can be produced by recombination as well (for instance, notice that recombining a tree with itself can yield a different tree.) We thus consider a first EA in which reproduction is done exclusively by means of recombination, and denote it by GP.

An alternative approach to that described above is possible, namely using no recombination but just mutation. In this sense, there are numerous possibilities for performing mutation on trees, see e.g. [11]. In this work, we have used two mutation operators:

- SCRAMBLE: Let $T$ be the tree to be mutated; firstly, a subtree $T'$ in $T$ is selected at random and pruned from $T$. Then, a new random subtree $T''$ is generated, with $\mathcal{L}(T') = \mathcal{L}(T'')$, and grafted at the original location of $T'$. In other words, the topology of a certain subtree of $T$ is rearranged at random.

- SWAP: it consists of selecting two leaves of $T$ at random, subsequently swapping their places.

We have considered two mutation-based evolutionary heuristics. The first one is denoted by EP, and consists of applying either SCRAMBLE or SWAP to an individual with 50% probability. The second one is denoted by ES, and just utilizes SCRAMBLE. However, an internal parameter is used to control the size of the subtrees to be rearranged. This parameter can be regarded as a step size, and evolves with each individual. To be precise, whenever a tree is to be mutated, the step size is firstly mutated using a gaussian distribution, and the mutated parameter value is used to select a subtree of the appropriate size to be fed to SCRAMBLE.

The acronyms –GP, EP, ES– used to denote each of the algorithms are intended to reflect their similarity with the corresponding EA family, namely genetic programming, evolutionary programming, and evolution strategies. However, it must be noted that the algorithms have been kept simple, and do not fully exploit the potential of the corresponding paradigm. This has been done so in order to obtain a first assessment on the usefulness of the different operators and evolution models for this problem.

## 4 Computational Results

The experiments have been performed using two test suites. Both of them comprise three different instances obtained from the biological domain. To be precise, three datasets for phylogenetic inference have been downloaded from TreeBASE. The size of these datasets ranges from 134 up to 178 leaves, as shown in Table 1. In the first test suite –termed AGGLOM– each dataset has been fed to three classical agglomerative clustering techniques: *single-link* [28], *complete-link* [9], and *average link* [29]. Thus, a collection of three trees is obtained in each case. In the second test suite –termed SCATTER– we consider for each dataset the different trees obtained in ten runs of a scatter search metaheuristic [30].

**Table 1.** The test suites considered in the experimentation. Mean distance refers to the average of distances of each tree in the collection to the whole tree set.

| | M877 | | M971 | | M808 | |
|---|---|---|---|---|---|---|
| | AGGLOM | SCATTER | AGGLOM | SCATTER | AGGLOM | SCATTER |
| mean distance | 44.78 | 94.56 | 56.71 | 89.79 | 31.59 | 88.80 |
| #leaves | 134 | | 158 | | 178 | |
| source | [31] | | [32] | | [33] | |

The parameterization of the algorithms is the following: GP is a steady-state EA, with a population of 100 individuals, using binary tournament for selection; EP has also a population size of 100 individuals, but uses flat selection, i.e., each individual is mutated once, thus yielding a population of 100 descendants. Subsequently, the best 100 out of the 100 existing individuals and the 100 newly

created descendants constitute the population for the next generation (i.e., a *plus* replacement strategy); as to the ES algorithm, it follows a (16,100) evolution model, and uses $n/10$ (where $n$ is the number of leaves) as the hyperparameter for the gaussian mutation of step sizes. In all cases, the algorithms are run for a total number of 250,000 fitness evaluations. Also, trees in the target collection are injected in the initial population.

Regarding the greedy heuristics, H1 is run using leaf permutations *compatible* with the topologies of trees in the collection (one leaf permutation for each tree)[3]. To be precise, let $T = (hLR)$; the sequence $\langle T \rangle = \langle L \rangle :: \langle R \rangle$, where :: represents sequence concatenation, and $\langle L \rangle$ and $\langle R \rangle$ are computed recursively ($\langle (a) \rangle = \langle a \rangle$), is compatible with $T$. H1 has then been run using sequences $\langle T_i \rangle, \cdots, \langle T_m \rangle$. The same is done in the internal invocations of H1 within H2.

**Table 2.** Results (averaged for 20 runs) of the EAs and the greedy heuristics on the two test suites considered. *sdv.* and *med.* stand for standard deviation and median respectively.

AGGLOM test suite

| | M877 | | | M971 | | | M808 | | |
|---|---|---|---|---|---|---|---|---|---|
| | best | mean ± sdv. | med. | best | mean ± sdv. | med. | best | mean ± sdv. | med. |
| GP | 51.80 | 51.68 ± 0.06 | 51.68 | 63.32 | 63.26 ± 0.04 | 63.27 | 48.45 | 48.41 ± 0.02 | 48.42 |
| EP | 50.91 | 50.87 ± 0.03 | 50.87 | 62.55 | 62.49 ± 0.02 | 62.48 | 48.16 | 48.16 ± 0.00 | 48.16 |
| ES | 50.55 | 50.34 ± 0.06 | 50.33 | 62.48 | 62.48 ± 0.01 | 62.48 | 48.15 | 48.15 ± 0.00 | 48.15 |
| H1 | 43.30 | 36.81 ± 5.80 | 37.92 | 61.21 | 57.80 ± 4.22 | 60.33 | 38.01 | 35.29 ± 1.94 | 34.23 |
| H2 | 49.53 | 40.08 ± 6.73 | 39.40 | 61.06 | 55.37 ± 5.39 | 57.74 | 47.92 | 32.22 ± 12.98 | 32.30 |

SCATTER test suite

| | M877 | | | M971 | | | M808 | | |
|---|---|---|---|---|---|---|---|---|---|
| | best | mean ± sdv. | med. | best | mean ± sdv. | med. | best | mean ± sdv. | med. |
| GP | 96.03 | 96.03 ± 0.00 | 96.03 | 91.90 | 91.90 ± 0.00 | 91.90 | 91.04 | 90.93 ± 0.09 | 90.97 |
| EP | 95.81 | 95.81 ± 0.00 | 95.81 | 91.90 | 91.90 ± 0.00 | 91.90 | 89.96 | 89.94 ± 0.03 | 89.94 |
| ES | 95.81 | 95.81 ± 0.00 | 95.81 | 91.90 | 91.90 ± 0.00 | 91.90 | 89.89 | 89.87 ± 0.02 | 89.87 |
| H1 | 83.81 | 81.33 ± 2.22 | 82.31 | 86.62 | 73.06 ± 8.94 | 74.07 | 84.58 | 78.27 ± 3.22 | 77.93 |
| H2 | 90.65 | 83.69 ± 3.43 | 84.25 | 86.49 | 74.48 ± 7.53 | 75.29 | 81.96 | 76.47 ± 3.70 | 77.52 |

The results are shown in Table 2. Notice firstly that all evolutionary heuristics perform clearly better than the greedy heuristics. The latter can hardly produce consensus trees of score similar to those already in the corresponding collection. The evolutionary algorithms can however produce consensus trees of high quality, achieving overall scores superior to the original trees. This implies that the topological information of the collection is being effectively summarized

[3] A leaf sequence is said to be compatible with a tree topology if there exists a layout of the tree in which its leaves are ordered from left to right as in the sequence, and no two branches cross.

in the consensus tree. The relative performance of the EAs indicates that EP is better than ES, and that GP provides the best outcome. A non-parametric statistical test (a Wilcoxon ranksum test [34]), has been used to corroborate the significance of these results. Fitness differences are always statistically significant (at the standard 5% significance level), except for the EP vs ES comparison on the m877 instance of the SCATTER test suite, and for all EAs on the m971 instance of the same test suite.

**Table 3.** Results (averaged for 20 runs) of the GP algorithm without seeding the initial population on AGGLOM (#1) and SCATTER (#2).

| | M877 | | | M971 | | | M808 | | |
|---|---|---|---|---|---|---|---|---|---|
| | best | mean ± sdv. | med. | best | mean ± sdv. | med. | best | mean ± sdv. | med. |
| #1 | 46.80 | 45.19 ± 1.16 | 45.32 | 57.51 | 56.63 ± 0.58 | 56.67 | 40.78 | 39.21 ± 0.86 | 39.18 |
| #2 | 81.99 | 79.86 ± 1.17 | 79.93 | 79.88 | 78.05 ± 0.78 | 78.01 | 79.12 | 77.05 ± 0.85 | 77.03 |

A final experiment has been done to confirm the usefulness of injecting the original tree collection in the initial population. The results are shown in Table 3 just for the GP algorithm. As it can be seen, the performance drop is dramatic. Without seeding, the algorithm would require much longer execution times in order to achieve the performance level of its seeded counterpart.

## 5 Conclusions

We have presented five heuristics for summarizing a collection of trees into a consensus tree, using the TreeRank measure as the scoring metric. From these five, the three evolutionary heuristics have been shown to be effective in summarizing within a single tree topological information contained in the target tree collection. Furthermore, a recombination-based EA has been shown to provide the better results. An important part of the performance of these algorithms is due to the seeding of the initial population.

Regarding the EAs, there is much room for improvement in the underlying evolution model, as anticipated in Sect. 3.2. For example, the EP algorithm could incorporate self-adaptation as well [35], evolving the number of times each mutation operator is used, or a hyperparameter controlling a probability distribution –e.g., a Poisson distribution– over this number of mutations. This would bring closer the ES and EP approaches, and constitutes a line for future developments.

With respect to the greedy heuristics, their performance is not satisfactory. Nevertheless, if not as stand-alone techniques, they can still be useful embedded within an EA. As mentioned in Sect. 3.1, both H1 and H2 must be fed with a particular leaf sequence. It is then conceivable to have an EA evolving leaf permutations to be fed to these heuristics. This way, they would act as decoders,

and the EA could benefit from their greedy functioning. This is another line for future developments.

## Acknowledgements

Thanks are due to the anonymous reviewers for their useful suggestions. The author is partially supported by Spanish MCyT, and FEDER under contract TIC2002-04498-C05-02.

## References

1. Foster, C.: Information retrieval: information storage and retrieval using AVL trees. In: Proceedings of the 1965 20th ACM National Conference, New York NY, ACM Press (1965) 192–205
2. Garofalakis, M., Özden, B., Silberschatz, A.: Resource scheduling in enhanced pay-per-view continuous media databases. In Jarke, M., et al., eds.: Proceedings of the 1997 International Conference on Very Large Databases, Athens, Greece, Morgan Kaufmann (1997) 516–525
3. Naylor, B.: Partitioning tree image representation and generation from 3D geometric models. In Booth, K., Fournier, A., eds.: Proceedings of the 1992 Conference on Graphics Interface, San Francisco CA, Morgan Kaufmann (1992) 201–212
4. Holmes, S.: Phylogenies: An overview. In Halloran, M., Geisser, S., eds.: Statistics and Genetics. Springer-Verlag, New York NY (1999) 81–119
5. Kim, J., Warnow, T.: Tutorial on phylogenetic tree estimation. In Lengauer, T., et al., eds.: Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology, Heidelberg, AAAI Press (1999) 196–205
6. Day, W., Johnson, D., Sankoff, D.: The computational complexity of inferring rooted phylogenies by parsimony. Mathematical Biosciences **81** (1986) 33–42
7. Foulds, L., Graham, R.: The Steiner problem in phylogeny is $NP-$complete. Advances in Applied Mathematics **3** (1982) 439–49
8. Wu, B., Chao, K.M., Tang, C.: Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices. Journal of Combinatorial Optimization **3** (1999) 199–211
9. King, B.: Step-wise clustering procedures. Journal of the American Statistical Association **69** (1967) 86–101
10. Moilanen, A.: Searching for the most parsimonious trees with simulated evolution. Cladistics **15** (1999) 39–50
11. Andreatta, A., Ribeiro, C.: Heuristics for the phylogeny problem. Journal of Heuristics **8** (2002) 429–447
12. Cotta, C., Moscato, P.: Inferring phylogenetic trees using evolutionary algorithms. In Merelo, J., et al., eds.: Parallel Problem Solving From Nature VII. Volume 2439 of Lecture Notes in Computer Science. Springer-Verlag, Berlin (2002) 720–729
13. Barker, D.: LVB: parsimony and simulated annealing in the search for phylogenetic trees. Bioinformatics **20** (2004) 274–275
14. Bryant, D.: A classification of consensus methods for phylogenetics. In Janowitz, M., et al., eds.: Bioconsensus. DIMACS-AMS (2003) 163–184
15. Swofford, D.: When are phylogeny estimates from molecular and morphological data incongruent? In Miyamoto, M., Cracraft, J., eds.: Phylogenetic analysis of DNA sequences. Oxford University Press (1991) 295–333

16. Phillips, C., Warnow, T.: The asymmetric median tree a new model for building consensus trees. Discrete Applied Mathematics **71** (1996) 311–335
17. Gusfield, D.: Efficient algorithms for inferring evolutionary trees. Networks **21** (1991) 19–28
18. Day, W.: Optimal algorithms for comparing trees with labeled leaves. Journal of Classiffication **2** (1985) 7–28
19. Barthélemy, J.P., McMorris, F.: The median procedure for $n$-trees. Journal of Classiffication **3** (1986) 329–334
20. Östlin, A.: Constructing evolutionary trees. Algorithms and complexity. PhD thesis, Department of Computer Science, Lund University (2001)
21. Penny, D., Hendy, M.: The use of tree comparison metrics. Systematic Zoology **34** (1985)
22. Allen, B., Steel, M.: Subtree transfer operations and their induced metrics on evolutionary trees. Annals of Combinatorics **5** (2001) 1–15
23. DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J., Zhang, L.: On computing the nearest neighbor interchange distance. In Du, D., Pardalos, P., Wang, J., eds.: Proceedings of the DIMACS Workshop on Discrete Problems with Medical Applications. Volume 55 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science., American Mathematical Society (2000) 125–143
24. Wang, J., Shan, H., Shasha, D., Piel, W.: Treerank: A similarity measure for nearest neighbor searching in phylogenetic databases. In: Proceedings of the 15th International Conference on Scientific and Statistical Database Management, Cambridge MA, IEEE Press (2003) 171–180
25. Jain, A., Murty, N., Flynn, P.: Data clustering: A review. ACM Computing Surveys **31** (1999) 264–323
26. Raidl, G., Julstrom, B.: Edge sets: an effective evolutionary coding of spanning trees. IEEE Transactions on Evolutionary Computation **7** (2003) 225–239
27. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms. Studies in Fuzziness and Soft Computing. Physica, Heidelberg (2002)
28. Sneath, P., Sokal, R.: Numerical Taxonomy. Freeman, London, UK (1973)
29. Ward, J.: Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association **58** (1963) 236–244
30. Cotta, C.: Scatter search with path relinking for phylogenetic inference. European Journal of Operational Research (2005) (to appear).
31. Hibbett, D., Donoghue, M.: Analysis of character correlations among wood decay mechanisms, mating systems, and substrate ranges in homobasidiomycetes. Systematic Biology **50** (2001) 1–27
32. Binder, M., Hibbett, D., Molitoris, H.: Phylogenetic relationships of the marine gasteromycete Nia vibrissa. Mycologia **93** (2001) 679–688
33. Hibbett, D., Gilbert, L.B., Donoghue, M.: Evolutionary instability of ectomycorrhizal symbioses in basidiomycetes. Nature **407** (2000) 506–508
34. Lehmann, E., D'Abrera, H.: Nonparametrics: Statistical Methods Based on Ranks. Prentice-Hall, Englewood Cliffs, NJ (1998)
35. Fogel, D.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, Piscataway, NJ (2000)