Finding Consensus Trees by Evolutionary, Variable Neighborhood Search, and Hybrid Algorithms

Sandro Pirkwieser
pirkwieser@ads.tuwien.ac.at

Günther R. Raidl
raidl@ads.tuwien.ac.at

Institute of Computer Graphics and Algorithms,
Vienna University of Technology,
Favoritenstraße 9-11/186-1, 1040 Vienna, Austria

ABSTRACT

The consensus tree problem arises in the domain of phylogenetics and seeks to find for a given collection of trees a single tree best representing it. Usually, such a tree collection is obtained by biologists for a given taxa set either via different phylogenetic inference methods or multiple applications of a non-deterministic procedure. There exist various consensus methods which often have the drawback of being very strict, limiting the resulting consensus tree in terms of its resolution and/or precision. A reason for this typically is the coarse granularity of the tree metric used. To find fully resolved (binary) consensus trees of high quality, we consider the fine-grained TreeRank similarity measure and extend a previously presented evolutionary algorithm (EA) to a memetic algorithm (MA) by including different variants of local search using neighborhoods based on moves of single taxa as well as subtrees. Furthermore, we propose a variable neighborhood search (VNS) with an embedded variable neighborhood descent (VND) based on the same neighborhood structures. Finally sequential and intertwined combinations of the EA and MA with the VNS/VND are investigated. We give results on real and artificially generated data indicating in particular the benefits of the hybrid methods.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; G.1.6 [**Numerical Analysis**]: Optimization—*Constrained optimization*; J.3 [**Life and Medical Sciences**]: *Biology and genetics*

General Terms

Algorithms

Keywords

consensus tree problem, phylogenetics, memetic algorithm, variable neighborhood search, hybrid methods

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

1. INTRODUCTION

The consensus tree problem has been first described in [1] alongside with a solution method. It frequently appears in the domain of phylogenetics [13], though it has applications in other clustering domains as well.

A *phylogenetic tree* is composed of nodes and branches (arcs) and models the evolutionary relationship between a set $\mathcal{L}$ of related objects called *taxa*. These taxa are the labeled leaf nodes of the tree whereas unlabeled inner nodes represent probably extinct ancestors. In this work we will only consider rooted unweighted binary trees, i.e. there exists a single distinguished root node being the common ancestor of all taxa, the relations represented by the tree are not weighted by any means, and each inner node always has exactly two direct descendants.

The *phylogeny problem* is to infer the intermediate ancestors and branches, thus to derive the evolutionary relationships, from given species data like DNA or RNA sequences. For this purpose a multitude of conceptually different inference methods exists, each having individual advantages and drawbacks [15].

Unfortunately, it is likely that biologists end up with several different trees for one and the same taxa set $\mathcal{L}$ due to (i) having multiple data sets available, (ii) inferring with different methods, or (iii) repeated runs of the same non-deterministic method.

The question is then how to take advantage of these probably high-quality but different and partly contradictory trees beside manually comparing and merging them. A possible solution is to look for a single tree over $\mathcal{L}$ "best" representing the collection $\mathcal{T}$. This non-trivial task is known as the *consensus tree problem* (CTP).

On the one hand the meaning of "best" depends on the desired information to retain in the consensus tree and on the other hand the possible consensus tree is literally restricted by the degree of strictness of the applied method as well as the granularity of the tree metric used, see also [3]. Figure 1 shows a schematic representation of this circumstance. Generally, a strict method and a coarse-grained metric rather lead to poorly resolved trees with few inner nodes having high degrees, and a substantial portion of the information contained in the input trees is lost. In contrast, we aim at deriving fully resolved (thus, binary) high-quality consensus trees inheriting as much information as possible. Our approach is therefore based on maximizing a more specific fine-grained measure, the so-called *TreeRank* score.

In Section 2 we report on previous as well as related work and define the TreeRank measure. Meaningful tree neigh-

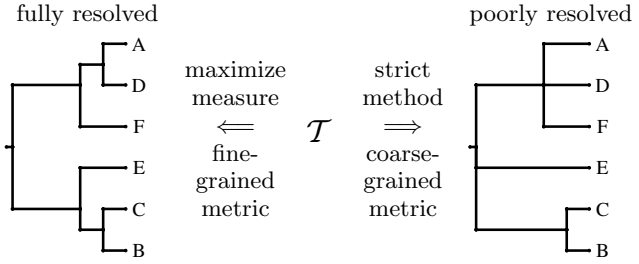


Figure 1: Consensus tree depending on method and metric.

borhood structures are presented in Section 3. They are applied in a variable neighborhood search (VNS) with an embedded variable neighborhood descent (VND) as described in Section 4. The neighborhoods are further utilized in Section 5 to extend an existing evolutionary algorithm (EA) by local search to a memetic algorithm (MA). Finally, in Section 6 we consider sequential and intertwined combinations of the EA (MA) and VNS (VND). Experimental results on real and artificially generated CTP instances are given in Section 7, followed by concluding remarks in Section 8.

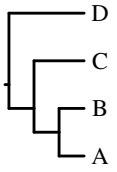
2. PREVIOUS AND RELATED WORK

Several consensus tree methods have already been proposed, see [3] for a good overview and comparison. Unfortunately, most methods have the drawback of being relatively strict, e.g. restricting the consensus tree to common substructures, and that the used tree metric is often coarse-grained, finally producing a quite poorly resolved or less intuitive solution tree. Prominent examples are the strict and majority consensus methods operating on clusters. A cluster is a subset of the set of taxa which contains all the descendants of its most recent common ancestor. The strict consensus method only retains clusters common to all input trees and the majority consensus method those appearing in more than half of them. The latter method can be regarded as a median method minimizing the number of non-common clusters, i.e. minimizing with respect to the *symmetric distance metric*. Further to mention is that the classical methods do not make use of any sophisticated search procedures and rely, if at all, on simple greedy approaches (e.g. the greedy consensus tree method available in PHYLIP [9]).

A recently proposed tree similarity measure, the *TreeRank measure* [23], originally introduced to handle database queries for similar trees[1], allows for more sophisticated procedures due to its fine granularity. This measure utilizes the quadratic Up matrix U which states for each pair of taxa (a, b) the number $U[a, b]$ of necessary up-traversals to reach from taxon a the least common ancestor of both taxa; see Figure 2 for an example. It can be derived in $O(|\mathcal{L}|^2)$ [23]. The authors also defined the Down matrix D in an analogous way, but since $U = D^{\mathrm{T}}$ it is redundant and the Up matrix is also called UpDown matrix. Having this matrix for two trees T_1 and T_2 and assuming equal taxa sets, one can calculate the *UpDown distance* between them by

$$UpDownDist(T_1, T_2) = \sum_{u,v \in \mathcal{L}} |U_{T_1}[u, v] - U_{T_2}[u, v]| \ .$$

[1]see `http://www.treebase.org`



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

Figure 2: Exemplary tree and its Up matrix.

This distance is finally the basis for the *TreeRank* score:

$$TreeRank(T_1, T_2) = \left(1 - \frac{UpDownDist(T_1, T_2)}{\sum_{u,v \in \mathcal{L}} U_{T_1}[u, v]}\right) \cdot 100\% \ .$$

For the general case of different taxa sets in T_1 and T_2 see [23]. The TreeRank score is thus a measure of the topological relationships in T_1 that are found to be the same or similar in T_2. It is bounded above by 100% but has no lower bound, which can be shown by comparing perfectly balanced and maximal unbalanced trees.

The first metaheuristic approaches applied to the consensus tree problem using the TreeRank measure have been described by Cotta [4]. Therein several evolutionary algorithms are presented differing in the way of whether applying mutation or crossover and the adopted evolution model. Tests on real-world instances indicate that solely applying the well-known prune-delete-graft recombination operator [16] in combination with a steady-state model performs best. This recombination operator selects a subtree of the first parent at random, removes its leaves in the second parent and grafts the subtree therein at a random position. Considering subtrees as building blocks, they are well preserved, inherited, and mixed by this operator. Although this operator was shown to be more destructive to one parent than to the other [20], it is the de facto standard when dealing with phylogenetic trees in evolutionary algorithms in general. The variants utilizing only mutation by scrambling subtrees or swapping taxa turned out to be inferior. It is further important to include the given input tree collection $\mathcal{T}$ in the initial population, otherwise the results are significantly worse. As fitness function the average TreeRank score of a candidate solution to the set of input trees is used:

$$TreeRank(T, \mathcal{T}) = \frac{\sum_{T' \in \mathcal{T}} TreeRank(T, T')}{|\mathcal{T}|} \ .$$

A solution tree is encoded in a pre-order traversal always stating the middle node followed by the nodes of the left and the right subtrees in a recursive way, yielding for the tree in Figure 2 (-1, -1, -1, A, B, C, D), whereas an inner node is represented by -1. This EA also forms the basis for our extension to a memetic algorithm and for the combination with the VND/VNS, reported in Sections 5 and 6. In [4] also greedy heuristics were proposed, but their performance was not satisfying; thus, we do not consider them here.

While we are not aware of other metaheuristics to identify consensus trees, there already exists a lot of such approaches for phylogenetic inference: Several EAs similar to the aforementioned are described in [6, 16]. In [7] an EA has been extended to a MA utilizing a local search based on subtree rotations. A more general survey of evolutionary computation in phylogenetics is given in [10]. Of further interest are applied metaheuristics apart from EAs like a greedy random-

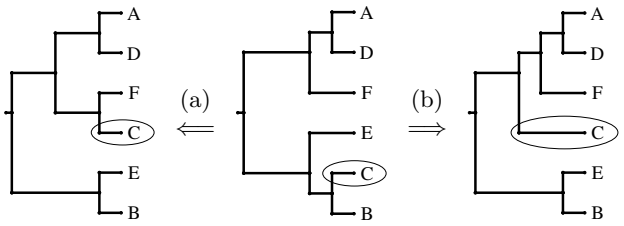


Figure 3: Exemplary Step moves of taxon C (a) to the leaf branch of F and (b) to an inner branch.

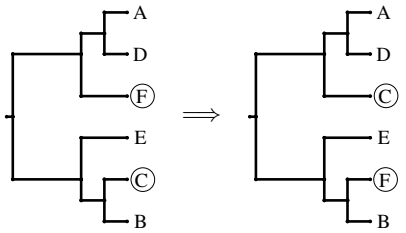


Figure 4: Exemplary Swap move of taxa C and F.

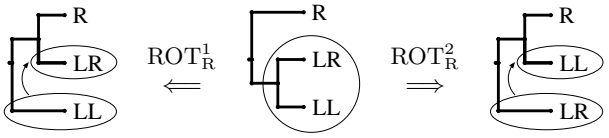


Figure 5: Schematic right rotations.

ized adaptive search procedure (GRASP) and a VNS with embedded VND utilizing NNI, Step, and SPR neighborhood structures [2]; see Section 3. A GRASP/VND hybrid using a multiple SPR neighborhood (i.e. a composition of successive SPR moves) was introduced in [19]. In our work we adopt some of these well working strategies originally proposed for phylogenetic inference to also solve the consensus tree problem in better ways.

3. NEIGHBORHOOD STRUCTURES

In this section the applied neighborhood structures *Step*, *Swap*, *Rotate*, and *SPRr* are described. For the unrooted case, a good overview of Step, Rotate, and the general SPR is given in [2]. At first we consider the neighborhoods dealing with single taxa.

A **Step** move consists of removing a taxon with its predecessor node and reinserting them at some other branch in the tree or as new root, see Figure 3 for an example. In a tree containing n taxa, there are always $n-2$ *inner branches* (i.e., arcs not directly leading to a taxon) and n *leaf branches* plus the position as new root. Hence after removing a single taxon with its leaf branch there are $((n-2-1)+(n-1)+1)-1 = 2n-4$ possible new insertion positions, yielding $n(2n-4)$ neighbors for a given tree reachable via one Step move. A Step move constitutes the smallest possible topology change of a tree. We search this neighborhood as well as the others following a deterministic first improvement strategy by utilizing the given pre-order traversal of the tree. In more detail, we go through the pre-order representation, consider each encountered taxon for removal and perform for it a nested tree traversal for enumerating all possible reinsertion points.

Our second neighborhood structure is defined by two related Step moves that exchange two taxa but keep the tree structure otherwise unchanged; thus, a **Swap** move is performed. Each pair of taxa can be swapped, resulting in $n(n-1)/2$ neighbors, whereof at most $n(n-1)/2-1$ are distinct because there exists at least one sibling pair. This neighborhood variant has not been used before; an example is given in Figure 4. When searching this neighborhood the possible Swap moves are deterministically enumerated similarly as the Step moves by two nested pre-order tree traversals.

The next two neighborhoods more generally operate on whole subtrees. The nearest neighbor interchange (NNI) move for the unrooted case from [2] can be interpreted as a rotation within the tree for the rooted case; thus, we call it a **Rotate** move. There are four possible rotations distinguishable, two right rotations (see Figure 5) and two left rotations in an analogous way. The two variants per direction result from the possibilities of connecting either the left or the right inner subtree directly to the (sub-)tree's root. For every inner branch there are two rotations possible, thus there are $2(n-2) = 2n-4$ Rotate neighbors. Hence it is a relatively small neighborhood. We search it by performing a pre-order traversal to enumerate all inner branches, trying for each one all valid rotations, and taking a best one in case several rotations lead to an improvement.

The last neighborhood we use is a restricted form of the subtree prune and re-graft (SPR) neighborhood, excluding the movement of single taxa, thus denoted by **SPRr**. A SPRr move selects a non-trivial subtree, prunes it from the tree, and re-grafts it at some other branch. There exist $O(n^2)$ neighbors; the actual number depends on the current tree topology and is investigated for SPR in more detail in [22]. Two exemplary moves are presented in Figure 6. The SPRr neighborhood is searched similarly as Step, i.e. we traverse the tree in pre-order to enumerate all non-trivial subtrees for pruning and perform for each a nested tree traversal to determine all branches for re-grafting.

3.1 Improvements

In order not to waste time on calculating the UpDown matrix always from scratch when evaluating a neighbor solution, we derived incremental update schemes for all the neighborhood structures. Only eventually affected parts of the UpDown matrix are efficiently updated. This strategy greatly improves the overall run-time. Unfortunately, it does not seem to be possible to follow this idea further by also calculating the UpDown distance and the TreeRank score itself in a significantly more efficient incremental way. Thus, these calculations are always completely performed. A second improvement we consider is to avoid unnecessary moves that would result in the original tree again: swapping two adjacent taxa or moving a taxon or subtree to the same relative position. In this way some calculations of the TreeRank score can be saved.

4. VNS WITH EMBEDDED VND

A detailed introduction to Variable Neighborhood Descent (VND) and Variable Neighborhood Search (VNS) is shown in [12]. The idea of VND is to extend simple local search by using several neighborhood structures—typically ordered according to increasing size or evaluation cost—and switching

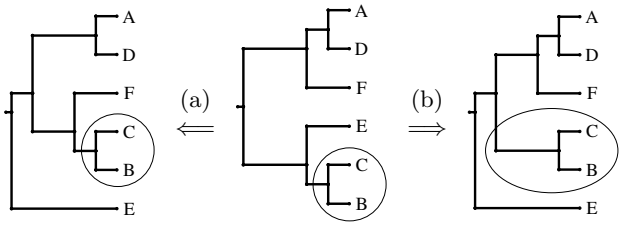


Figure 6: Exemplary SPRr moves of subtree (-1, B, C) (a) to the leaf branch of F and (b) to an inner branch.

between them in a systematic way. If no improving solution is found within one neighborhood, the next neighborhood is considered; else the search is recentered at the new incumbent and restarts with the first neighborhood structure. We use the neighborhoods described in the previous section and apply the already mentioned first-improvement strategy, thus always immediately accepting the first solution yielding a better TreeRank score than the current. Due to the size and related evaluation effort of the neighborhoods and their impact on a tree's structure the following order is used for the VND: Rotate, Swap, Step, and finally SPRr.

In contrast to VND, VNS focuses more on diversification by applying shaking, i.e. random moves in larger neighborhoods. For intensification, VNS includes a local search component, in this case VND. Algorithm 1 shows the pseudocode of the VNS with embedded VND. If the VNS is applied as a stand-alone algorithm, then we utilize an input tree from $\mathcal{T}$ with the highest TreeRank score as initial solution. For shaking, we perform a series of Step moves: A certain percentage of the taxa is randomly selected and moved to some other, also randomly chosen positions. We start with 5% of the taxa and gradually increase this portion by 5% up to 100% and hence the number of different VNS neighborhoods $k_{\max}$ is 20. Both VND and VNS terminate if an iteration or time limit is reached. VND also stops when the last neighborhood contains no better solution and thus, the current solution is locally optimal w.r.t. all VND neighborhood structures.

Algorithm 1: VNS with embedded VND (x)

```
Set neighborhood structures for VND: N_1=Rotate,
N_2=Swap, N_3=Step and N_4=SPRr;
k ← 0; // shaking strength
while VNS stopping condition is not met do
    while k ≠ k_max do
        // Shaking:
        x' ← Apply random Step moves on
        k · 5%|L| taxa of x selected at random;
        // Local search by VND:
        l ← 1;
        while l ≠ 4 and VND stopping condition is not
        met do
            // Exploration of neighborhood l:
            Search first improving neighbor x'' ∈ N_l(x');
            // Eventually move to x'' and
            // change neighborhood within VND:
            if better solution x'' found then
                x' ← x'';
                l ← 1;
            else l ← l + 1;
        // Eventually move to x' and
        // change shaking strength:
        if x' is better than x then
            x ← x'; // set new incumbent
            k ← 1; // reset shaking
        else k ← k + 1; // increase shaking
    // finished VNS iteration
    k ← 1; // reset shaking
```

5. MEMETIC ALGORITHM

The basic idea of a memetic algorithm (MA) is—at least as it is most commonly interpreted—to extend an evolutionary algorithm by a local search component in a suitable way to increase the solution quality or speed up the search; for a more general introduction to MAs see [17].

For the CTP, we use the EA variant that has been found to perform best in [4] as a basis. As already mentioned in Section 2 it always applies prune-delete-graft (PDG) recombination and no mutation. The EA selects parents via binary tournaments with replacement and works in a steady-state fashion, i.e. in each iteration one offspring is derived, and it always replaces the worst solution in the population with one exception: To avoid duplicates and enforce a minimum diversity, new solutions having the same TreeRank score as solutions in the population are immediately discarded. The initial population consists of a copy of the input tree collection $\mathcal{T}$ and otherwise randomly created trees on $\mathcal{L}$ without any bias. In addition to [4] we also tried to use recombination and mutation together, though according to preliminary tests the performance was usually worse, hence solely recombination was finally used, too. The EA terminates when a given time or iteration limit is reached.

We embedded different variants of local search utilizing the neighborhoods described in Section 3. SPRr was omitted as it yielded poorer results in preliminary tests, presumably because of the similarity to PDG recombination. Each of the variants uses a single neighborhood structure and applies a random neighbor step function; i.e. a random move is performed and the new solution is accepted if it is better than the original one. Improved trees are re-encoded in the chromosome in a Lamarckian manner. A local search phase terminates after a certain number of consecutive non-improving moves.

We further developed a *simple progressive search* (SPS) roughly following the idea in [11]. In this original work, the algorithm starts with the large SPR neighborhood and progressively reduces it by limiting the distance between the removal and insertion positions of a subtree until the neighborhood converges to NNI having a distance limit of one. In contrast, we use several different neighborhoods without this smooth transition though maintaining the idea to reduce the size of the applied neighborhood: In the first third of the MA—either w.r.t. an iteration or time limit— we apply the Step local search, followed by Swap in the second third, and finally Rotate in the last third, whereas the local search variants are the ones described before.

Although a single application of one of the local search variants is relatively fast, trying to improve every offspring would dramatically increase the overall run-time without a

Algorithm 2: Memetic Algorithm

```
Initialize population;
while stopping condition is not met do
    Select parents x_a and x_b via binary tournaments;
    x_c ← Apply PDG recombination on x_a and x_b;
    if new best solution x_c found then
        Apply local search on x_c;
    Insert x_c in population when there is no other
    solution with the same TreeRank score, replacing
    the worst solution;
```

substantial quality gain in the final solution. Preliminary tests further indicated that it is also not wise to apply local search on a certain portion of randomly chosen offsprings. Similarly as in [7], we perform it only on new incumbent solutions, which is restrained but turned out to be most effective. A pseudo-code of the MA is shown in Algorithm 2.

6. EA/VND/VNS HYBRIDS

Over the last years, hybrid metaheuristics have become increasingly popular as they are often able to exploit the advantages of different simpler optimization techniques yielding an improved overall performance [8, 18]. In fact, in our case it is an obvious idea to combine the EA with the described VND or VNS. In our first approach, we do this as in the MA of the previous section: The VND is always only applied to new best solutions as a strong local improvement. This EA/VND hybrid will be denoted by Hyb_B.

Another combination possibility is to run the EA and the VNS in a pure sequential way: The EA's finally best solution is used as the starting point for the VNS. We term this algorithmic setting as Hyb_S.

A refinement of this approach is the intertwined execution of the EA and the VNS, as it is shown in Algorithm 3: A prespecified total execution time T is divided into 2τ slots and both algorithms are alternately applied. The EA starts and retains its population during its pauses. When the VNS takes over, it always begins with the EA's so far best solution and with its first neighborhood. Of course, each final solution of the VNS at the end of its slots is also inserted into the EA's population, replacing the worst solution and rejecting solutions having the same TreeRank scores as already existing ones. We denote such an intertwined setting with τ EA/VNS phases by $\text{Hyb}_{\text{I}\tau}$.

One might think it is even better to use the MA instead of the EA in the hybrid variants, but this is not true in general. Preliminary tests have clearly shown a worse performance of a sequential MA/VNS combination. This is probably due to the MA already focusing too strongly on local optima and thus seeding the VNS with a solution less suitable for further improvement. In the following, we therefore only consider in more detail an intertwined MA/VNS hybrid which we denote by $\text{Hyb}^*_{\text{I}\tau}$.

7. EXPERIMENTAL RESULTS

This section gives details of the algorithm settings, introduces the test instances, and presents the computational results as well as a comparison between them. All approaches have been implemented in C++ and were compiled with GCC 4.1.2. The experiments were performed on a 2.2 GHz Dual-Core AMD Opteron 2214 PC with 4 GB RAM. The best performing (pure) EA from [4] has been reimplemented as proposed in this original work and described in Section 5.

Algorithm 3: Intertwined EA/VNS Hybrid (T, τ)

```
Initialize population of EA;
t_slot = T/(2 · τ); // time limit per slot
iter ← 0;
while iter < τ do
    Run the EA for duration t_slot;
    x ← best solution in EA's population;
    Run the VNS on x for duration t_slot;
    x' ← best solution found by VNS;
    if x' is better than x then
        Insert x' in population of EA;
    iter ← iter + 1;
```

7.1 Algorithms Settings

The population size of the EA and MA was set to 100. In the MA local search is applied to all new incumbent solutions until 100 consecutive non-improving moves have been tried. Due to space limitations, we present here only results for the MA with simple progressive search, which proved to generally outperform the variants where only a single neighborhood structure is considered. As mentioned earlier, the initial solution for the standalone VNS is the tree with highest TreeRank score of the input collection. Hyb_B is performing the VND on all new incumbents limiting runtime per call to at most 5% of the overall time limit. In case of Hyb_S the EA and the VNS are given 50% of the computation time each, thus it basically corresponds to $\text{Hyb}_{\text{I}1}$. The intertwined EA/VNS hybrid $\text{Hyb}_{\text{I}\tau}$ was applied with $\tau = 4$ alterations, thus in the following denoted by $\text{Hyb}_{\text{I}4}$. For $\text{Hyb}^*_{\text{I}\tau}$ we basically combine the settings of the best performing MA with SPS and of $\text{Hyb}_{\text{I}4}$ and call it $\text{Hyb}^*_{\text{I}4}$.

7.2 Test Instances

We test the algorithms on three types of instances: trees resulting from three simple agglomerative clusterings (single-link and complete-link [14] as well as average-link, also known as unweighted pair group method with arithmetic mean [21]), trees resulting from several runs of the scatter search approach in [5], and new artificial trees. The latter are created by generating one initial random tree and deriving the actual input trees out of it by copying it and applying a series of perturbations in the neighborhoods described in Section 3: Random Step, Swap, Rotate, and SPRr moves are equally likely performed. In order to be able to control the similarity of the resulting input trees in a good way, we defined minimum and maximum pairwise TreeRank scores and performed the perturbations until a derived tree achieves a pairwise score w.r.t. the initial tree within these limits. If the actual score is less than the lower bound, the previous move is undone and the process continues. Note that the initial tree is finally discarded and not included in the input tree collection. A schematic presentation of this process is shown in Figure 7. An advantage of these artificially generated instances is that the known initial tree, although not necessarily the best possible consensus tree, lends itself as a reference solution.

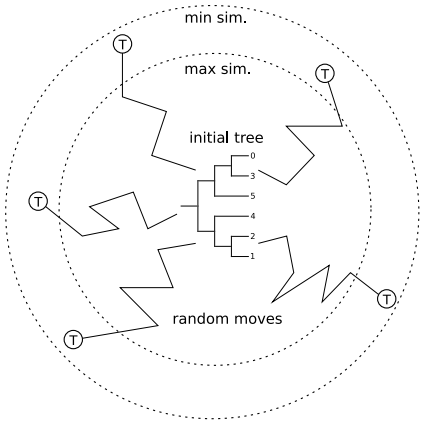


Figure 7: Artificial instances created by randomly perturbing an initial tree.

The instances are differing in the amount of taxa (134 to 178), the number of input trees (3 to 10) and their average similarity to each other (below 50% up to 90%).

7.3 Comparison of Algorithms

Our aim was to make a comparison of the different algorithms as fair as possible. Hence per instance we ran the pure EA for 500000 iterations and used the consumed CPU time as limit for all others. Following this we did 30 runs per algorithm setting and instance and state following TreeRank scores in Tables 1–4: the best result (best), the mean value (mean) with the corresponding standard deviation (sdv.) as well as the median value (med.). Overall best obtained mean values are printed bold. For the agglomerative clustering and scatter search instances we also give the number of trees and taxa in parentheses, e.g. "(3x134)" for 3 trees with 134 taxa each. In case of the artificial tree instances the name itself holds this information and also the TreeRank score upper bound used during creation, e.g. "5x150_70" for 5 trees with 150 taxa each and a maximal TreeRank score of 70% w.r.t the initial tree. As lower limit the given upper bound minus 10% was used, i.e. 60% in the previous example. For these instances the TreeRank score of the initial input tree is given in line "init tree", too. We further list for each instance the TreeRank score of the best tree of the input collection $\mathcal{T}$ ("best input") and the aforementioned time limit ("CPU-time") in seconds. In the following discussion, all performance differences (i.e. differences in TreeRank score mean values) pointed out have been statistically verified by Wilcoxon rank sum tests and error levels are less than 5%.

For seven out of the 12 instances the MA yields on average significantly better solutions than the pure EA. In the remaining five cases the observed differences are not significant (although for four the MAs mean values are higher and only for instance M808scatter the EA achieved a slightly better mean). VNS is always significantly better than the pure EA except for M808scatter, 5x175_80, and 5x175_90, where the latter achieved higher scores. Compared to the MA, the VNS exhibits clearly better results on the real-world instances, but was less effective on the artificially generated ones, for which it achieved a higher mean score in only a single case (5x150_80). We believe the reason for this lies in the initial solution of the VNS, which is the input tree having the highest TreeRank score. While this best input tree turned out to lie relatively close to high quality consensus trees in case of our real-world instances, there are generally larger differences in the artificial instances. As the VNS is dominated by its strong local search of the embedded VND, which consumes the major part of the CPU-time on larger instances, its diversification abilities are less pronounced than those of the population-based MA.

Now we concentrate on the hybrid approaches. The results of Hyb_B, in which VND is used to locally optimize new incumbent solutions within the EA, was disappointing, as its final solutions are generally inferior to those of the other hybrids and VNS performed in most of the real-world instances better. The more systematic EA/VNS combinations Hyb_S, $\text{Hyb}_{\text{I}4}$, and $\text{Hyb}^*_{\text{I}4}$ are more successful. They consistently achieve the overall best results; at least one of them performed on all instances significantly better than all other algorithms. The only exception is Onco10, where the VNS yields equally good results. Although there are only few statistically significant differences between the sequential and the intertwined hybrids, the former tends to yield better results for the real-world instances whereas the latter seems to be better suited for the artificial instances. Finally, the intertwined MA/VNS hybrid $\text{Hyb}^*_{\text{I}4}$ shows for many instances a better performance—in fact sometimes even the best—when compared to the same variant utilizing the pure EA only ($\text{Hyb}_{\text{I}4}$). Altogether the intertwined variants can be clearly considered the most successful.

When comparing the scores of the best input trees with those of the overall best solutions found, it is apparent that our real instances have less room for improvement, mostly below 2%, whereas the artificially generated ones allow for about 5% or even more. Another interesting fact is the relation between the TreeRank scores of the artificial instances' initial trees and the corresponding best solutions found. As can be observed in Tables 3 and 4, the initial tree is more likely the (nearly) optimal consensus tree (regarding the TreeRank score) when the derived input trees are close to the initial tree, hence when the radius of the inner circle in Figure 7 is small. The results of the best solutions also show that only the hybrid algorithms are consistently able to find consensus trees being better than or equal to the initial trees, whereas the latter happens for instances 5x150_90 and 5x175_90.

8. CONCLUSIONS

In this work, we introduced four meaningful neighborhood structures for the CTP, whereof Step and Rotate are adopted from approaches for deriving phylogenetic trees, SPRr is a restricted variant of SPR and Swap is a specific form of two Step moves and has not been described before. A previously presented evolutionary algorithm using the fine-grained TreeRank similarity measure has been extended to several memetic algorithm variants by embedding a randomized local search utilizing these neighborhood structures. Thereof the variant using the simple progressive search, which changes the neighborhoods over time, performs best. Furthermore, a variable neighborhood descent procedure that also uses these neighborhood structures has been developed and embedded in a variable neighborhood search. The latter performs shaking by applying an increasing number of random Step moves. The evaluation of neighborhoods could be substantially sped up by implementing a (partly) incremental scheme through updates of the Up-

Table 1: Results on agglomerative clustering tree instances.

	M877 (3x134)				M971 (3x158)				M808agglom (3x178)			
	best	mean	sdv.	med.	best	mean	sdv.	med.	best	mean	sdv.	med.
EA	51.85	51.68	0.10	51.70	63.38	63.28	0.04	63.28	48.47	48.44	0.01	48.44
MA	51.90	51.73	0.11	51.75	63.36	63.28	0.05	63.28	48.51	48.46	0.02	48.46
VNS	51.97	51.88	0.05	51.88	63.34	63.32	0.01	63.32	48.53	48.50	0.01	48.49
Hyb_B	51.94	51.82	0.09	51.84	63.41	63.34	0.04	63.35	48.51	48.49	0.01	48.48
Hyb_S	51.97	**51.89**	0.04	51.89	63.41	**63.36**	0.03	63.36	48.52	48.50	0.02	48.50
Hyb_{I4}	52.00	**51.89**	0.06	51.88	63.40	63.35	0.03	63.36	48.53	48.51	0.01	48.51
Hyb^*_{I4}	51.99	51.88	0.07	51.89	63.41	63.33	0.05	63.34	48.53	**48.52**	0.01	48.52
best input	49.99				62.41				48.14			
CPU-time [s]	228				283				480			

Table 2: Results on scatter search tree instances.

	Onco9 (9x148)				Onco10 (10x148)				M808scatter (10x178)			
	best	mean	sdv.	med.	best	mean	sdv.	med.	best	mean	sdv.	med.
EA	91.21	90.98	0.10	90.97	91.01	90.98	0.02	90.97	91.05	90.98	0.05	90.98
MA	91.21	90.99	0.11	90.98	91.07	90.98	0.03	90.98	91.08	90.96	0.07	90.97
VNS	91.16	91.16	0	91.16	91.09	**91.09**	0	91.09	90.96	90.96	0	90.96
Hyb_B	91.27	91.12	0.07	91.12	91.12	91.05	0.05	91.07	91.05	90.92	0.10	90.94
Hyb_S	91.29	**91.21**	0.04	91.22	91.13	**91.09**	0.02	91.09	91.09	**91.00**	0.04	91.00
Hyb_{I4}	91.26	91.20	0.04	91.22	91.13	**91.09**	0.02	91.09	91.10	90.99	0.06	91.00
Hyb^*_{I4}	91.28	**91.21**	0.03	91.22	91.12	**91.09**	0.01	91.09	91.11	90.99	0.06	90.99
best input	89.96				90.87				89.85			
CPU-time [s]	391				420				573			

Down matrix. Next, we hybridized the EA and MA with the VNS or the VND by running them either in sequential or intertwined order exchanging improved solutions.

Performance has been evaluated with extensive tests on agglomerative clustering, scatter search, and carefully generated artificial instances providing more room for improvement. The MA utilizing SPS, the VNS, and the hybrid approaches are usually able to outperform the EA. On our real-world instances, the VNS turned out to be better than the MA, but on the artificial instances the situation was vice-versa. We explained this behavior with the VNS' stronger focus on intensification and weaker diversification abilities.

The hybrid approaches generally perform best, clearly exploiting the benefits of the individual algorithms they combine. Especially the intertwined hybrids yield consistently excellent and in most cases the overall best solutions.

Future work should include experiments on more problem instances with different properties. Promising seems to be the investigation of intertwined hybrids with more sophisticated alteration schemes, in particular since we observed that in some cases the EA or MA is not able to improve the solution in later time-slots anymore. Currently, we are investigating extended versions of the Swap, Step, and SPRr neighborhoods where we exploit the given Up-Down distance information to realize guided neighborhood explorations biased towards changing "expensive" (i.e. dissimilar) structures first. A further idea is to consider other neighborhood orders for the VND or even a (self-)adaptive strategy.

9. ACKNOWLEDGEMENTS

This work is funded by the Austrian Exchange Service, Acciones Integradas Austria-Spain, under grant 13/2006 and by the Austrian Science Fund (FWF) under contract number P20342-N13. We also want to thank Carlos Cotta, Antonio J. Fernández and José E. Gallardo for their valuable feedback regarding this project.

10. REFERENCES

[1] E. N. Adams. Consensus techniques and the comparison of taxonomic trees. *Systematic Zoology*, 21(4):390–397, 1972.

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

[3] D. Bryant. A classification of consensus methods for phylogenies. In M. Janowitz et al., editors, *Bioconsensus*, DIMACS, pages 163–184. AMS, 2003.

[4] C. Cotta. On the application of evolutionary algorithms to the consensus tree problem. In J. Gottlieb and G. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3248 of *LNCS*, pages 58–67. Springer, 2005.

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

[6] C. Cotta and P. Moscato. Inferring phylogenetic trees using evolutionary algorithms. In J. Merelo et al., editors, *Parallel Problem Solving From Nature VII*, volume 2439 of *LNCS*, pages 720–729. Springer, 2002.

[7] C. Cotta and P. Moscato. A memetic-aided approach to hierarchical clustering from distance matrices: Application to gene expression clustering and phylogeny. *BioSystems*, 72(1–2):75–97, 2003.

[8] M. El-Abd and M. Kamel. A taxonomy of cooperative search algorithms. In M. J. Blesa et al., editors,

Table 3: Results on artificial tree instances with 150 taxa.

	5x150_70				5x150_80				5x150_90			
	best	mean	sdv.	med.	best	mean	sdv.	med.	best	mean	sdv.	med.
EA	68.86	67.81	0.40	67.70	78.56	77.95	0.28	77.98	88.65	87.99	0.31	88.01
MA	69.10	68.55	0.26	68.61	78.55	78.15	0.24	78.20	88.87	88.46	0.30	88.54
VNS	68.47	68.47	0	68.47	78.53	78.53	0	78.53	88.35	88.35	0	88.35
Hyb_B	69.40	68.90	0.21	68.83	78.83	78.46	0.26	78.53	88.91	88.50	0.34	88.59
Hyb_S	69.48	68.97	0.32	68.97	78.85	78.64	0.20	78.70	88.96	88.78	0.18	88.82
Hyb_{I4}	69.52	69.06	0.26	69.05	78.84	**78.67**	0.12	78.69	88.96	**88.84**	0.12	88.88
Hyb^*_{I4}	69.53	**69.16**	0.23	69.23	78.82	**78.67**	0.17	78.73	88.96	88.82	0.14	88.87
best input	64.54				72.65				84.34			
init tree	67.24				78.27				88.96			
CPU-time [s]	297				310				314			

Table 4: Results on artificial tree instances with 175 taxa.

	5x175_70				5x175_80				5x175_90			
	best	mean	sdv.	med.	best	mean	sdv.	med.	best	mean	sdv.	med.
EA	69.57	68.66	0.38	68.65	76.94	75.98	0.48	75.96	86.44	85.71	0.47	85.62
MA	70.91	70.57	0.17	70.60	77.31	76.92	0.19	76.94	86.44	85.77	0.36	85.79
VNS	69.53	69.53	0	69.53	73.78	73.73	0.03	73.71	85.31	85.31	0.01	85.31
Hyb_B	70.95	70.47	0.24	70.51	77.51	76.95	0.33	76.95	86.33	85.95	0.28	86.00
Hyb_S	70.99	70.37	0.24	70.38	77.56	77.18	0.24	77.21	86.44	**86.37**	0.10	86.42
Hyb_{I4}	71.20	70.70	0.19	70.67	77.50	77.21	0.18	77.22	86.44	86.33	0.12	86.39
Hyb^*_{I4}	71.27	**70.94**	0.16	70.99	77.53	**77.27**	0.12	77.26	86.44	86.25	0.18	86.27
best input	64.75				72.10				81.34			
init tree	69.23				77.35				86.44			
CPU-time [s]	382				384				380			

Proceedings of Hybrid Metaheuristics, Second International Workshop, volume 3636 of *LNCS*, pages 32–41. Springer, 2005.

[9] J. Felsenstein. PHYLIP (Phylogeny Inference Package) version 3.6, 2005. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

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

[11] A. Goëffon, J.-M. Richer, and J.-K. Hao. Progressive tree neighborhood applied to the maximum parsimony problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(1):136–145, 2008.

[12] P. Hansen and N. Mladenović. Variable neighborhood search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publishers, 2003.

[13] S. Holmes. Phylogenies: An overview. In M. Halloran and S. Geisser, editors, *Statistics and Genetics*, pages 81–119. Springer, 1999.

[14] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[15] 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*, pages 196–205. AAAI Press, 1999.

[16] A. Moilanen. Searching for most parsimonious trees with simulated evolutionary optimization. *Cladistics*, 15(1):39–50, 1999.

[17] 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, 2003.

[18] G. R. Raidl. A unified view on hybrid metaheuristics. In F. Almeida et al., editors, *Proceedings of Hybrid Metaheuristics, Third International Workshop*, volume 4030 of *LNCS*, pages 1–12. Springer, 2006.

[19] C. C. Ribeiro and D. S. Vianna. A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. *International Transactions in Operational Research*, 12:325–338, 2005.

[20] L. Sheneman and J. A. Foster. Estimating the destructiveness of crossover on binary tree representations. In *Proceedings of the 8th Conference on Genetic and Evolutionary Computation*, pages 1427–1428. ACM Press, 2006.

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

[22] Y. S. Song. On the combinatorics of rooted binary phylogenetic trees. *Annals of Combinatorics*, 7:365–379, 2003.

[23] J. T. L. Wang, H. Shan, D. Shasha, and W. H. Piel. TreeRank: a similarity measure for nearest neighbor searching in phylogenetic databases. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*, pages 171–180. IEEE Press, 2003.