EXACT METHODS AND METAHEURISTIC APPROACHES FOR DERIVING HIGH QUALITY FULLY RESOLVED CONSENSUS TREES*

Sandro Pirkwieser[1], Rubén Ruiz-Torrubiano[2] and Günther R. Raidl[1]

The consensus tree problem primarily arises in the domain of phylogenetics and seeks to find for a given collection of trees a single tree best representing it. Such a tree collection is usually obtained by biologists for a specific taxa set either via different phylogenetic inference methods or multiple applications of a non-deterministic procedure. Existing consensus methods often have the disadvantage 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. Hence we both utilize the fine-grained TreeRank similarity measure within metaheuristics and apply exact methods based on appropriate integer linear programming (ILP) models to find fully resolved (binary) consensus trees of high quality. We further give results on several real and new artificially generated data.

1. Introduction

The consensus tree problem [1] mostly occurs in phylogenetics [12], whereas the *phylogeny problem* is to infer a *phylogenetic tree* modeling the evolutionary relationship between a set L of related objects called *taxa*. For practical reasons (e.g. using different phylogenetic inference methods or multiple applications of a non-deterministic procedure [14]) it is likely that a biologist ends up with a collection T of several different and partly contradictory trees for one and the same taxa set L. 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.

This leads to the *consensus tree problem* (CTP), which seeks to find for a given collection of trees a single tree over L "best" representing it. 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 [4]. 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

* 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.

1 Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstraße 9-11/1861, A-1040 Vienna, Austria, {pirkwieser|raidl}@ads.tuwien.ac.at

2 Computer Science Department, Universidad Autónoma de Madrid, Ciudad Universitaria de Cantoblanco, 28049 Madrid, Spain, ruben.ruiz@uam.es

information as possible. Our approach is therefore based on maximizing more specific fine-grained measures, primarily the so-called *TreeRank* score [20].

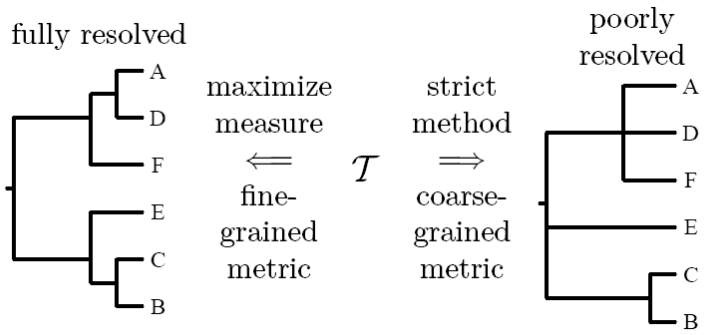


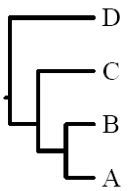
Figure 1: Consensus tree depending on method and metric

In Section 2 we report on previous as well as related work and define the TreeRank measure. The developed exact models and solution methods are presented in Section 3, followed by the metaheuristic approaches in Section 4. Experimental results on real and artificially generated CTP instances are given in Section 5, followed by concluding remarks in Section 6.

2. Previous and Related Work

Several consensus tree methods have already been proposed, see [4] for a good overview and comparison. Unfortunately, most methods have the drawback of being relatively strict, such as 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. 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* [20], originally introduced to handle database queries for similar trees in TreeBASE[3], 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 $O(|L|^2)$ [20]. The authors also defined the Down matrix D in an analogous way, but since $U = D^T$ it is redundant and the Up matrix is also called UpDown matrix. Having the matrices U_{T1} and U_{T2} for two trees T_1 and T_2, respectively, and assuming equal taxa sets, one can calculate the *UpDown distance* between them by



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

3 See `http://www.treebase.org`

Figure 2: Exemplary tree and its Up matrix

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

This distance is finally the basis for the *TreeRank* (TR) *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 [20]. 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 limit, 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 [5]. He presented several evolutionary algorithms (EAs) differing in the adopted evolution model and whether or not applying mutation and crossover. 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. 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:

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

A solution tree is encoded in a direct way via 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 (MA) and for the combination with the VNS, reported in Section 4.

While we are not aware of other metaheuristics to identify consensus trees, there exist quite a few of such approaches for phylogenetic inference: EAs similar to the aforementioned are described in [7, 16], and in [8] a MA additionally applies a local search based on subtree rotations. Further to mention are a greedy randomized adaptive search procedure (GRASP) and a VNS with embedded VND [2], basically utilizing the neighborhood structures described later in Section 4. Hence 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. Exact Methods

The TreeRank measure is not (directly) applicable as objective function for an ILP model, since it is highly non-linear. An alternative measure to be *minimized* is the previously introduced UpDown distance, which is a linear function on the values of the UpDown matrix. Nonetheless, an ILP model with this measure as objective function requires too much

computational effort to be solved, even for very small instancess. A much more efficient measure can be defined by representing a tree using rooted triplets [4]. A rooted triplet is a list *(a, b|c)* of three taxa *a, b, c* ∈ *L* in which the least common ancestor of *a* and *b* is a descendant of the least common ancestor of *a*, *b* and *c*. A solution tree composed of rooted triplets can be easily represented by means of binary variables $t_{a,b|c} \in \{0, 1\}$, where $t_{a,b|c} = 1$ states that the triplet *(a, b|c)* is present. The consensus tree can be defined as the one *maximizing* the number of common rooted triplets in the whole input tree collection. Let *R(T)* be the set of rooted triplets defining tree *T*. Following this, the *Weighted Triplet* (WT) measure is defined by:

$$WT(T, \mathcal{T}) = \sum_{k=1}^{K} |R(T) \cap R(T_k)| = \sum_{t_{a,b|c} \in R(T)} w^{\mathcal{T}}_{a,b|c} t_{a,b|c} \tag{1}$$

where each coefficient $w_{a,b|c}$ corresponds to the number of input trees in which triplet *(a, b|c)* is present. This measure also is linear, and easy to calculate. Moreover, weights can be easily assigned to the input tree collection representing the confidence on each phylogenetic inference method used.

The proposed ILP model includes both u_{ab} variables representing the values in the UpDown matrix of the final solution tree, and $t_{a,b|c}$ variables representing the presence or absence of each possible triplet. The purpose of this is to allow the use of an objective function based on the values of the UpDown matrix (more realistic), and at the same time obtain the tree defined by the triplet variables, in order to avoid the conversion from the UpDown matrix to the triplet variables (this algorithm is not yet known, and can be a topic of future investigation). Consistency of the UpDown matrix also requires that the $t_{a,b|c}$ are used. Following constraints are used together with one of the objective functions to define the ILP model to be solved:

$$u_{aa} = 0 \qquad \forall a \in \mathcal{L} \tag{2}$$
$$1 \leq u_{ab} \leq n - 1 \qquad \forall a, b \in \mathcal{L} \tag{3}$$
$$u_{ab} < u_{ac} + M(1 - t_{a,b|c}) \qquad \forall \{a, b, c\} \subset \mathcal{L} \tag{4}$$
$$u_{ba} < u_{bc} + M(1 - t_{a,b|c}) \qquad \forall \{a, b, c\} \subset \mathcal{L} \tag{5}$$
$$u_{ca} \leq u_{cb} + M(1 - t_{a,b|c}) \qquad \forall \{a, b, c\} \subset \mathcal{L} \tag{6}$$
$$u_{cb} \leq u_{ca} + M(1 - t_{a,b|c}) \qquad \forall \{a, b, c\} \subset \mathcal{L} \tag{7}$$
$$u_{ac} - u_{ab} \leq u_{bc} - u_{ba} + M(1 - t_{a,b|c}) \qquad \forall \{a, b, c\} \subset \mathcal{L} \tag{8}$$
$$u_{bc} - u_{ba} \leq u_{ac} - u_{ab} + M(1 - t_{a,b|c}) \qquad \forall \{a, b, c\} \subset \mathcal{L} \tag{9}$$
$$\min\{u_{ab} \mid b \in \mathcal{L} \setminus \{a\}\} = 1 \qquad \forall a \in \mathcal{L} \tag{10}$$
$$t_{a,b|c} + t_{b,c|a} + t_{a,c|b} = 1 \qquad \forall a < b < c \in \mathcal{L} \tag{11}$$
$$t_{a,b|c} + t_{a,d|c} - t_{b,d|c} \leq 1 \qquad \forall \{a, b, c, d\} \subset \mathcal{L} \tag{12}$$
$$t_{a,b|c} + t_{a,c|d} - t_{a,b|d} \leq 1 \qquad \forall \{a, b, c, d\} \subset \mathcal{L} \tag{13}$$
$$t_{ab|c} = t_{ba|c} \qquad \forall a < b, c \in \mathcal{L} \tag{14}$$

The distance from a taxon to itself is zero (2), and the distance between two different taxa is at least one and at most the number of taxa minus one (3), since the latter is the largest depth of a binary tree having *n* leaves. Inequality (11) ensures that only one triplet *(a, b|c)*, *(b, c|a)*, or *(a, c|b)* is realized and inequalities (4)-(9) that the UpDown matrix is consistent. In these constraints *M* is a constant such that $M >> n$, where *n* is the number of taxa. This ensures that

the constraints are activated only when the triplet *(a, b|c)* is present ($t_{a,b|c} = 1$). Constraints (8)-(9) are called *path constraints*. They ensure that the values themselves of the UpDown matrix, and not only the relative distances, are consistent. The *row-min constraints* (10) ensure that no "artificial" inner nodes may be added to lower specific taxa, which might otherwise happen, depending on the objective function used. Inequalities (12) and (13) express the triplet transitivity and telescopic conditions, which are used in order to derive new necessary triplets from other ones; see [21] in case of quartets. Finally, equality (14) state that triplets *(a, b|c)* and *(b, a|c)* are equivalent. This model has $\Theta(n^3)$ variables and $\Theta(n^4)$ constraints. Note that the WT measure (1) as objective function together with inequalities (11)-(14) represents a completely independent model, which can be used when obtaining the UpDown matrix is not the main concern. This reduced model has the advantage of being very efficient, since the objective function is easy to calculate. Moreover, intensive computational tests are possible, and larger instances can be solved in practice. Hence this model will be subsequently used.

3.1. Lazy Constraints

When solving an ILP problem, it is often the case that some constraints are very unlikely violated. These are normally consistency constraints, which must be satisfied by any feasible solution. In this case, the elimination of these constraints from the model could be considered: Fewer constraints usually mean higher efficiency. The method, of course, would no longer be exact, and the feasibility of the solution obtained after solving the model to optimality should be checked after every optimization. To solve these inconvenients, and at the same time speed up the computations required, the constraints can be treated as *lazy constraints*: These are constraints which are initially not included in the optimization process, but they are added as soon as it is detected that they are violated by the current candidate solution. Depending on the problem, this can dramatically improve the performance of the algorithm.

3.2. Heuristic Generation of Variables

Instead of reducing the number of constraints the optimization algorithm has to deal with, the number of *variables* considered for solving the problem can also be restricted at the beginning. Triplet variables $t_{a,b|c}$ corresponding to triplets *(a, b|c)* for which it is assumed that they are not present in the final solution are eliminated from the problem. This step is referred to as *pruning* the variable set. The important question is: How is the variable set to be pruned, so as not to discard the optimal solution? Since there is no theorical background to answer this question, the variable set is heuristically pruned: variables with a low probability of appearing in the optimal solution are pruned. These are the triplets *not appearing* in any input tree. Since the tree which best summarizes the information contained in the input trees is sought, it is expected that triplets not appearing in any input tree will not be present in the final solution.

The idea of beginning with a reduced set of variables is closely related to *column generation* [15] algorithms, which is a very powerful technique to solve ILP problems with many variables. First, a meaningful subset of variables is chosen to begin with. The optimal solution (if a feasible solution exists) to this subproblem is found, and the variables are *priced*: a reduced cost is calculated for each variable, which is based on the solution of the dual problem (which is also a linear problem). If this reduced cost is positive for all variables, then the solution is provably optimal. If not, the variable with the most negative reduced cost is added to the model. A column generation algorithm embedded in a branch-and-bound approach is called a *branch-and-price* algorithm. Since for the CTP the pricing problem is both difficult to state and to solve, we propose a heuristic method in which variables are

added when in an incumbent a new variable appears. For this purpose, a rounding procedure is applied to each non-integer candidate solution, so that a feasible tree is found. We call this method *heuristic column generation*. In this approach, the initial set of variables is chosen by the previous pruning approach.

4. Metaheuristic Approaches

Contrary to problem specific heuristics, metaheuristics operate on a higher level and guide one or more of these heuristics so as to escape local minima. Thereby they try to maintain an appropriate balance between intensification and diversification [3]. In this section we will extend the EA mentioned in Section 2 (for more details see the original work [5]) by a local search component obtaining a memetic algorithm [17]. Thus additional intensification is added with the aim to get solutions of higher quality. This local search component as well as the VNS with VND, described later on, are based on the following tree neighborhood structures: *Step*, *Swap*, *Rotate* and *SPRr*, see [2] for an overview of these (with the general SPR) for unrooted trees. The corresponding neighborhoods are defined for a given solution to be those solutions reachable via one move of the specific neighborhood structure.

Two of them operate on single taxa: a Step move removes a taxon with its predecessor node and reinserts them at some other branch in the tree or as new root, whereas a Swap move consists of two related Step moves that exchange two taxa but keep the tree structure otherwise unchanged. The other two deal with whole subtrees: a Rotate move performs a rotation within a tree, i.e. rotating a subtree, finally a SPRr move is a restricted variant of a subtree prune and re-graft (SPR) move selecting a nontrivial subtree, pruning it from the tree, and re-grafting it at some other branch.

The best performing local search component for the MA resembles those detailed in [10]. In the first third of the MA—either w.r.t. an iteration or time limit—we apply a Step local search, followed by Swap in the second third, and finally Rotate in the last third, whereas the local search performs a random move and accepts improving solutions until a certain number of consecutive non-improving moves is reached; we chose 100 to be this limit. An additional restriction is to apply the local search only after a new incumbent solution has been found, which saves a lot of computational effort.

Next we briefly describe the VNS with embedded VND, for more details see [11]. The basic idea of VND is to use several neighborhood structures with an appropriate ordering and systematically switch between them. Thus if no better solution could be found in the actual neighborhood go to the next one, otherwise start with the first again. We chose the ordering to be Rotate, Swap, Step and SPRr, due to the size and related evaluation effort. All four neighborhood structures are searched in a deterministic way applying a first-improvement strategy, i.e. always immediately accepting the first solution yielding a better TreeRank score than the current. Whereas the VND can obviously intensify the search, the VNS is responsible for adequately adding diversification. This is done via so-called shaking moves, which is in our case accomplished by making a certain number of Step moves selected at random. Shaking is applied whenever the VND's last neighborhood led to no improvement.

Following the recent practice of hybridizing metaheuristics [18] to exploit the advantages of different simpler optimization techniques yielding an improved overall performance, we combine the EA/MA and the VNS (with the embedded VND) in two ways. First, we run the EA and the VNS sequentially starting with the EA which initializes the VNS afterwards with

the best solution found. Using the MA instead of the EA in this setting leads to an overall inferior performance due to the MA's stronger focus on locally optimal solutions. The second hybrid method is to apply the EA/MA and the VNS in an intertwined fashion by dividing the execution time into slots and execute both algorithms alternately. Thereby the EA/MA retains its population throughout all cycles, whereas the VNS always restarts using the so far best solution.

5. Experimental Results

All approaches have been implemented in C++ and were compiled with GCC 4.1.2. The experiments were performed on a 2.2 GHz DC AMD Opteron 2214 PC with 4GB RAM.

5.1 Exact Methods

The ILP-based methods proposed in Section 3 are compared when used to solve the model (11)-(14) with Triplet Score as objective function (eq.(1)), since this approach allows intensive computational tests to be performed. In the experiments, both real (`mammals20` and `mammals34`) and random instances (`instancex_r`) are included. The random instances are expected to have quite dissimilar input trees, on the other hand real instances are expected to include highly similar input trees. The ILP solver CPLEX 10.0 was used for the first approach (lazy constraints), while the open-source alternative COIN-BCP with CPLEX as LP solver was used for branch-and-price, since the feature of adding new variables is not available in CPLEX.

Instance	Taxa	CPLEX + lazy cons						
		Best	Best (TR)	Best (TR) In	Time	Time (w.o. init)	Total lazy	Used lazy
instance1_r	9	141	57.51	66.32	0.06	0.04	4536	202
instance2_r	12	367	44.30	57.57	5.47	5.4	17820	1374
instance3_r	12	392	51.38	60.88	0.17	0.1	17820	288
instance4_r	15	819	48.75	56.73	12.25	12.06	49140	2994
instance5_r	15	1014	68.73	75.90	0.23	0.04	49140	1073
instance6_r	15	1088	72.94	74.29	0.25	0.05	49140	433
instance7_r	20	2113	52.94	69.84	53.81	53.11	174420	4182
instance8_r	20	2397	60.67	69.75	1.62	0.91	174420	1993
instance9_r	20	2275	54.89	69.34	1.36	0.69	174420	2080
instance10_r	25	4719	62.56	68.54	8.89	7.13	455400	2659
mammals20_r	20	3186*	91.60	91.60	0.79	0.1	174420	0
mammals34_r	34	16552	86.20	86.72	7.97	1.24	1669536	0

Table 1: Results obtained with lazy constraints

As Table 1 shows, the performance of the solver using lazy constraints is generally good. In this table, the second column shows the numbers of taxa, while the third, fourth and fifth columns show the best WT scores, their corresponding TR values and the TR values of the best input trees, respectively. Overall times to solve the problem with and without initialization step (i.e. initializing the model) are listed in the next two columns. Finally, the last two columns show the total numbers of lazy constraints, and the numbers of lazy constraints which are added to the problem formulation. The largest real instance (34 taxa) is solved in less than eight seconds, while the largest random instance (25 taxa) is solved in less than nine seconds. Nonetheless, the solver requires more than fifty seconds to solve `instance7_r`, because it takes longer to find the first integer feasible solution than in the other 20 taxa instances and more lazy constraints (4182) had to be added to the initial formulation. In the other 20 taxa instances only about 2000 lazy constraints had to be added. The same behaviour can be observed in the 15 taxa instances. It is remarkable that in real instances no lazy constraints nedded to be added, since the input trees are quite similar. As can be seen, both WT score and TR measure are quite uncorrelated: the best WT tree is in

most cases worse than the best input tree regarding TR, except in case of the `mammals20` instance where the optimal tree regarding both measures is one of the input trees.

Instance	Taxa	BCP + Heur. col. gen.					
		Best	Pruning	Added vars	Time	Time (w.o. init)	Time BCP
instance1_r	9	140	71.43	4	1.24	0.32	1.04
instance2_r	12	367	71.97	5	29.00	13.74	119.12
instance3_r	12	392	69.70	11	24.07	8.87	20.15
instance4_r	15	819	69.52	10	131.14	10.48	238.04
instance5_r	15	1003	57.95	2	117.12	1.33	121.64
instance6_r	15	1086	52.31	18	116.48	2.35	121.40
instance7_r	20	2079	66.49	31	20853.27	19270.70	2725.15
instance8_r	20	2390	60.70	0	1515.83	7.7	1533.81
instance9_r	20	2267	63.95	1	1611.59	20.66	1521.89
instance10_r	25	4712	61.77	0	10331.13	15.92	10409.24
mammals20_r	20	3186	40.18	0	1423.38	0.96	1528.80
mammals34_r	34	16552	40.91	0	130323.55	11.27	139226.51

Table 2: Results obtained with heuristic column generation

Table 2 shows the results using the heuristic column generation approach with BCP. The values obtained are therefore not necessarily optimal. An additional column labeled “Time BCP” shows the total time required without column generation by the standard BCP solver. The column “Pruning” shows the percentages of remaining variables after the pruning procedure. The quality of the solutions obtained seems to be near optimal. In the real instances, and in two random instances (`instance3_r` and `instance4_r`) the optimal solution is already found (in the real instances, no additional variables are needed). In general, using this approach represents an improvement in execution time when compared to normal BCP. More pruning takes place in real instances, mainly due to the similarity among the input trees, than in random instances, where input trees are expected to be more dissimilar. The number of added variables is not particularly high, taking into account that a great number of variables was eliminated. This can have two possible explanations: either few variables need to be added in order to find a solution of reasonably good quality, or the rounding heuristic is not able to better identify which variables need to be added in order to find the optimal solution. Since BCP generally needs more time than CPLEX does, even the CPU times of Tables 1 and 2 without considering the initialization time cannot be directly compared.

5.2. Metaheuristics

We test the metaheuristics on three types of larger instances: (1) trees resulting from three simple agglomerative clusterings: single-link and complete-link [13] as well as average-link [19], with `M877` and `M971` consisting of three trees and 134 and 158 taxa, respectively, (2) trees resulting from several runs of the scatter search approach in [6], with `Onco9` consisting of nine and `Onco10` of ten trees and 148 taxa each, and (3) 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 4: Random Step, Swap, Rotate, and SPRr moves are equally likely performed. To achieve a desired similarity of the resulting input trees, 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, whereas the initial tree itself is not included in the input tree collection. 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. In case of these instances the name itself holds the information about the number of trees and taxa and also the TreeRank score upper bound used.

The results of the re-implemented version of the basic EA of [5] and those of the three hybrid variants are shown in Table 3. We set a CPU time limit per instance (time[s]), which was determined by running the pure EA for 500000 iterations. Further we state the TreeRank score of the best input tree (input), for the artificially created instances additionally the initial tree as a reference solution (init), and following TreeRank scores per algorithm and instance: the best result (best) and 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.

As can be seen the hybrid variants achieve better results for all instances than the pure EA, which was also statistically verified by performing Wilcoxon rank sum tests with error levels of less than 5%. Looking at the different hybrids, the sequential variant tends to yield better results for the real-world instances whereas the intertwined variants seem to be better suited for the artificial instances. Of the latter variants, the intertwined MA/VNS hybrid shows for many instances a better performance when compared to the same variant utilizing the pure EA only, thus the additional intensification paid off.

Further it is to be noted that the artificially generated instances allow more room for improvement (ca. 5%) in contrast to our real instances (ca. 2%), and that in case of the former only the hybrid algorithms are consistently able to find consensus trees being better than or equal to the initial trees.

Instance	time[s]	input	init	EA				Sequential EA/VNS				Intertwined EA/VNS				Intertwined MA/VNS			
				best	mean	sdv.	med.	best	mean	sdv.	med.	best	mean	sdv.	med.	best	mean	sdv.	med.
M877	228	49.99	-	51.85	51.68	0.10	51.70	51.97	**51.89**	0.04	51.89	52.00	**51.89**	0.06	51.88	51.99	51.88	0.07	51.89
M971	283	62.41	-	63.38	63.28	0.04	63.28	63.41	**63.36**	0.03	63.36	63.40	63.35	0.03	63.36	63.41	63.33	0.05	63.34
Onco9	391	89.96	-	91.21	90.98	0.10	90.97	91.29	**91.21**	0.04	91.22	91.26	91.20	0.04	91.22	91.28	**91.21**	0.03	91.22
Onco10	420	90.87	-	91.01	90.98	0.02	90.97	91.13	**91.09**	0.02	91.09	91.13	**91.09**	0.02	91.09	91.13	**91.09**	0.02	91.09
5x150_70	297	64.54	67.24	68.86	67.81	0.40	67.70	69.48	68.97	0.32	68.97	69.52	69.06	0.26	69.05	69.53	**69.16**	0.23	69.23
5x150_80	310	72.65	78.27	78.56	77.95	0.28	77.98	78.85	78.64	0.20	78.70	78.84	**78.67**	0.12	78.69	78.82	**78.67**	0.17	78.73
5x150_90	314	84.34	88.96	88.65	87.99	0.31	88.01	88.96	88.78	0.18	88.82	88.96	**88.84**	0.12	88.88	88.96	88.82	0.14	88.87
5x175_70	382	64.75	69.23	69.57	68.66	0.38	68.65	70.99	70.37	0.24	70.38	71.20	70.70	0.19	70.67	71.27	**70.94**	0.16	70.99
5x175_80	384	72.10	77.35	76.94	75.98	0.48	75.96	77.56	77.18	0.24	77.21	77.50	77.21	0.18	77.22	77.53	**77.27**	0.12	77.26
5x175_90	380	81.34	86.44	86.44	85.71	0.47	85.62	86.44	**86.37**	0.10	86.42	86.44	86.33	0.12	86.39	86.44	86.25	0.18	86.27
Avg.		73.29		76.65	76.10			77.01	76.79			77.03	76.83			77.04	76.86		

Table 3: Results of the metaheuristic approaches

6. Conclusions

In this work we proposed metaheuristic approaches utilizing the fine-grained TreeRank measure and exact integer linear programming models and solution methods. In general, the use of lazy constraints is the most successful approach which has been investigated to solve the problem by means of ILP techniques. It has also been investigated that by pruning the available variables, other near-optimal solutions can be obtained. Therefore, the assumption that triplets not appearing in any input tree are not likely to appear in an optimal solution can be considered as a reliable pruning criterion. The use of the heuristic variable generation technique is clearly beneficial for this problem, since both execution times and quality of the final solution are improved. More effort should be spent on finding new and efficient variable generation heuristics able to identify which variables should be added to the formulation in order to obtain the optimal solution. We further introduced an extension of a previously presented EA to an MA as well as a VNS with an embedded VND based on four specific tree neighborhood structures to better solve larger CTP instances. Of the metaheuristics, the hybrid approaches clearly exploit the benefits of the individual algorithms they combine and always outperform the pure EA. Especially the intertwined hybrids yield consistently excellent and in most cases the overall best solutions. Nevertheless, further tests on other and more diverse instances, a refined VND (e.g. regarding the neighborhood order) and a closer look at the hybridization might be future work on the metaheuristic side.

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]C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, 35(3):268–308, 2003.

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

[5]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.

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

[7]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.

[8]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.

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

[10]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

[11] 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.

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

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

[14]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.

[15]M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. Operations Research, 53(6):1007–1023, 2005.

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

[20]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.

[21]J. Weyer-Menkhoff. New Quartet Methods in Phylogenetic Combinatorics. PhD thesis, Universität Bielefeld, Germany, 2003.