Maastricht University

Fast heuristics for computing TBR distance

August 8, 2022

Sébastien Boxho

***Abstract*—There exist various distances to compare two phylogenetic trees. Some are easy to compute, others are NP-hard. The Tree Bisection and Reconnection (TBR) distance is an NP-hard distance which has some good exact algorithms and some fast lower bounding algorithm, but not yet fast upper bounding algorithms. Hence, this paper aims to introduce heuristic algorithms to compute a good upper bound on TBR-distance. Thus, we will leverage the known relationship between TBR-distance and Maximum Agreement Forests to produce fast algorithms capable of computing good upper bounds. This paper will present three different approaches, a greedy algorithm, a Monte Carlo Tree Search algorithm, and a randomized search based on an existing branching algorithm. The conducted experiments will give insights into which algorithm performs the best upper bounds.**

Keywords: Phylogenetic Tree – TBR-distance – Maximum Agreement Forest

I. Introduction

Phylogenetic trees or phylogenies are the standard model for representing evolutionary processes. All extant taxa are assumed to descend from the same common ancestor and diverge in a tree-like fashion through speciation events. Phylogenies can be rooted or unrooted unordered trees with unlabeled internal nodes but uniquely labeled leaves. Here we will focus on unrooted bifurcating trees, where each internal node has exactly three neighbors.

There are multiple methods and formats to construct or represent a phylogenetic tree. Throughout this paper we will refer to a standard text such as Felsenstein (2004) [1] for such construction methods and Newick (1990) for the format in which phylogenetic trees are often represented.

For this paper we will not be concerned with inferring phylogenetic trees, or their representation, but rather with computing the distance between two phylogenetic trees. There are various techniques and metrics to compute the distance, such as the Subtree Prune and Regraft (SPR), Nearest Neighbor Interchange (NNI), Maximum Agreement Forest (MAF) and the Tree Bisection and Reconnection (TBR) distance [2]. These different techniques have their own underlying semantics, but their aim is the same, to evaluate the distance between different phylogenetic trees with the same taxa. The purpose of this research is to build fast methods for computing upper bounds on TBR-distance, as such algorithms may generate useful information regarding the TBR distance itself or they may be used to accelerate already existing exact algorithms. Furthermore, the TBR-distance helps the understanding of the space of all phylogenetic trees on a fixed number of leaves $|X|$ where X is the set of leaves. The problem of computing the TBR-distance is known to be NP-hard [2] but fortunately, fixed parameter tractable which means that it can be solved in time $f(k)*|X|^{O(1)}$ where f is a single-exponential function that depends only on k, k is the TBR-distance and as mentioned above X the set of Taxa. This property allows exact algorithms to run quickly as long as the TBR-distance is not too large. However, what do we do when the TBR distance is large and exact algorithms slow down too much to be feasible? We then need a set of heuristics to induce an answer to this question. In this paper, we attempt to answer three questions: 1. Which algorithm design strategies work well for constructing fast, high-accuracy heuristics for computation of TBR distance? 2. How does one incorporate trade-off in the design of such heuristics: How far can more running time be traded for higher accuracy? 3. How do the fast heuristics compare to existing (exact) methods?

Part of the answer for the first question was inspired by the insights provided by Allen and Steel (2001) [2] as the TBR-distance between two unrooted phylogenetic trees can be characterized by a maximum agreement forest, informally, a forest with a minimum

number of components that covers the set of leaves of both trees. In particular, the components have to induce disjoint subtrees in both trees, and the subtrees must have the same topology in both trees. This characteristic has facilitated the development of fixed parameter tractable algorithms and approximation algorithms but here we will take advantage of it to build fast heuristics. To answer the second question, we will investigate the evolution of the results given by the three different algorithms depending on the times allotted. For the third question we will compare the accuracy of the different results produced in the second answer to a dataset containing the exact TBR-distance which were computed in van Wersch, R., Kelk, S., Linz, S. et al. [3]. These exact TBR-distances were computed with the only publicly available TBR solver uSPR [4] (available here https://github.com/cwhidden/uspr) or their Integer Linear Programming (ILP) based *Tubro* solver. The uSPR solver implements an exact TBR solver using iterative deepening as described in Chen's algorithm Chen et al. 2015 [5].

The rest of this paper is subdivided into the following sections: Section 2 will define and describe the vocabulary and methods, which will be used throughout the paper. Section 3 will describe the methods and the framework of the different techniques used to calculate the TBR-distance. Section 4 will present the experiment setup and its results. Section 5 offers a high-level discussion followed by the conclusion.

II. Preliminaries

This section provides the notation and terminology used throughout this paper and a brief description of the TBR and MAF models.

A. Unrooted Phylogenetic Tree

An unrooted phylogenetic tree is a simple, connected, acyclic, and undirected graph, where each leaf vertex has degree one and is bijectively labeled by a set X of labels. As there is no root, there are $n-2$ inner vertices which all have degree 3 and where n is equal to the number of leaves. For the purpose of this paper we will assume without loss of generality that $|X| \geq 4$ as the TBR-distance is 0 if $|X| < 4$

B. Tree Rearrangements operation

A tree rearrangement operation transforms a phylogenetic tree into another tree by making a small graph theoretical change. For example, a TBR-operation involves detaching a subtree and attaching it elsewhere. The following section will give a formal definition to this abstract explanation.

C. Tree Bisection and Reconnection - Distance

Let T and T' be two unrooted trees on X. The edges are not directed, and their weights or value will not be considered, so they are assumed to be 0.

A TBR-operation (see figure 1) is defined as a three-step operation applied on T [6]:

1) **Step 1:** Pick an edge in T, delete it, and then suppress any resulting degree-2 vertex, as the resulting phylogenetic trees T_1 and T_2 need to follow the constraints mentioned above.
2) **Step 2:** If T_1 (respectively T_2) has at least one edge, subdivide an edge in T_1 (resp. T_2) with a new vertex v_1 (resp. v_2) and otherwise set v_1 (resp. v_2) to be the single isolated vertex of T_1 (resp. T_2).
3) **Step 3:** Add a new edge between v_1 and v_2 to obtain a new phylogenetic tree T' with X the same set of taxa and a new edge $\{v_1, v_2\}$.

These steps are the definition of a single TBR-operation. The Tree Bisection and Reconnection distance between two trees T and T' is equal to the minimum number of tree rearrangement operations required, in this case the so-called TBR-operation. The distance is known as NP-hard to compute and often denoted as $d_{TBR}(T, T')$.

D. Maximum Agreement Forest

Let T and T' be two unrooted phylogenetic trees on the same set of taxa X. Let $F = \{B_0, B_1, B_2, \dots\}$ be a partition of X, where each element of F is denoted by B_i with $i \in \{0, 1, 2, \dots, k\}$ and is known as a component. The term $T[X']$ denotes the unique, minimal subtree of T that connects all elements in X' where $X' \subset X$ and we denote $T|X'$ as the restriction of T to X' obtained after the suppression of the degree 2 vertices in $T[X']$. We say that F is an agreement forest for T and T' if these two conditions hold:

1) For each i contained in $\{0, 1, 2, \dots, k\}$, we have $T|B_i = T'|B_i$.
2) For each pair i, j contained in $\{0, 1, 2, \dots, k\}$ with $i \neq j$, we have that $T[B_i]$ and $T[B_j]$ are vertex-disjoint in T, and $T'[B_i]$ and $T'[B_j]$ are vertex-disjoint in T'.

The first condition ensures that the subtrees induced by each element B_i of F have the same topology in both T and T'. While the second condition ensures that the subtrees obtained in 1. are disjoint in both trees.

Finally, let F be an agreement forest for T and T'. The size of F is denoted by $d_{MAF}(T, T')$ and is simply the number of components. The maximum agreement

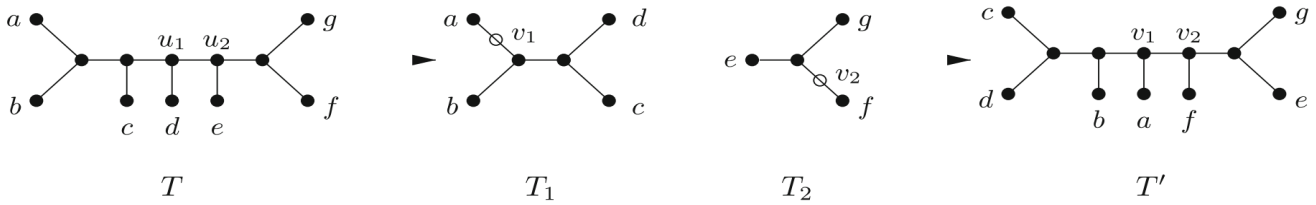

Fig. 1: An example of a single TBR operation. We first removes an edge in T to obtain T1 and T2. Then, generate T' by reconnecting both subtrees by introducing an edge between the open circles [3]

forest (MAF) for T and T' is defined as an agreement forest with a minimum number of components. It is well known that the number of components d_{MAF} is equal to $d_{TBR} + 1$ [2]. We will use this equivalence when computing upper bounds on the TBR-distance. Computing a MAF is known to be fixed-parameter tractable and NP-hard. The techniques used to compute a MAF are often a combination of these four techniques: kernelization, exponential search, depth-bounded search, and cluster partitioning [7].

E. Cherry Reduction

Two distinct leaves x, y form a cherry $\{x, y\}$ of T if they are adjacent to a common vertex i.e. they share a common parent. If x,y is a cherry in both trees T and T', then it is called a common cherry. It is well-known that replacing a common cherry with a single, new leaf does not alter the TBR distance [6]. Replacing common cherries can thus reduce the number of leaves in the trees making it easier to compute the TBR distance.

F. Quartets

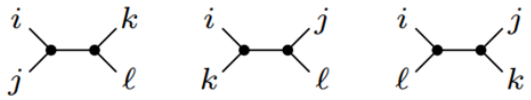

Fig. 2: Three possible quartet topologies

Let T be an unrooted binary phylogenetic tree on X, the set of taxa. A quartet is an unrooted binary tree for a quadruple of taxa, it represents the smallest informative element in the tree. Let $\{i, j, k, l\} \in X$, then there are only three kinds of topologies possible $ij|kl$, $ik|jl$ and $il|jk$ (see figure 2) where only one of the three possibilities holds. The character $|$ represents the half-way point on the two sides of the quartet. For example, $ab|cd$ is a quartet of T if the path from a to b does not intersect with the path from c to d. Two trees T and T' on the same set of leaves are topologically equivalent if and only if they can be decomposed in exactly the same set of quartets.

G. Dataset

To perform the experiments necessary for this research the dataset used in van Wersch, R., Kelk, S., Linz, S. et al. [3] was taken. The dataset consists of 735 tree pairs, where each pair (T, T') consists of two phylogenetic trees on the same set of taxa. The trees are partitioned into seven sets of size $|X| \in \{50, 100, 150, 200, 250, 300, 350\}$ and with different skewness with $s \in \{50, 70, 90\}$. The higher the skew, the further the trees are from being balanced. For each set of taxa they applied $k \in \{5, 10, 15, 20, 25, 30, 35\}$ TBR-operations to T to obtain T'. This ensures that $d_{TBR}(T, T') \leq k$. A second dataset containing 90 larger tree pairs was also generated with the parameters $|X| \in \{500, 1000, 1500, 2000, 2500, 3000\}, k \equiv 35$ and the same s. This dataset will give insights when scaling our algorithms.

III. Methods

This section includes a brief description of each algorithm used to compute the upper bounds on TBR-distance.

A. Greedy Search

The greedy algorithm computes upper bounds on the TBR-distance between two phylogenetic trees T and T' by randomly picking leaves to add them to an existing or new component of an agreement forest.

Firstly, a random starting leaf from the tree T is selected. This same leaf is then found in the second tree T'. Both leaves are added to the first component of the agreement forest. Then, for each iteration a random leaf is selected in the tree T and found in T' and the path connecting the leaf to any leaf in the component is marked. The paths must be marked

to make sure that the internal nodes connecting the leaves from the same component are not used by any other path connecting two leaves from another component. Otherwise, the components would not be disjoint therefore the agreement forest would not hold. If the condition holds for both trees, then the leaf can be added to the first component. Once, the component of the agreement forest has a size equal or greater than 4 the topology must be verified which is not necessary for components of size 3 or smaller as they always have the same topology. Since, the components are not only sets of taxa, but they represent a subtree of the phylogenetic trees T and T' it is crucial to constantly check the equivalence of topology in both subtrees. Thus, for every size-4-subset in the component it is necessary to verify that the induced quartets have the same topology in both trees. As long as the components of each tree T and T' have the same topology they represent the same subtree. If a leaf cannot be added to an existing component without violating any condition, then it is added to a new component. If the adjacent vertex of a leaf is already in a component and the leaf cannot be added to this component without violating any condition, then this leaf is designated as a singleton and will require a component by its own. Thus, the algorithm continues to randomly pick leaves and add them to the desired component if the path and topology conditions are both fulfilled. The greedy algorithm runs until every taxon of both trees has been assigned to a component of the agreement forest.

B. Iterative Greedy

The iterative greedy algorithm is an advanced version of the greedy algorithm with slight improvements. The issue with the greedy algorithm is that it will most likely not find the best solution after its first run. So, the iterative greedy performs multiple runs of the greedy algorithm and keeps the best distance. The greedy algorithm selects the leaves to add to the component in a randomized manner therefore by performing multiple iteration of the greedy algorithm one gets various TBR-distances. The more iterations we try, the more likely we are to find the optimal solution or a good approximation. But as the objective is to find good upper bounds, we cannot allow too many iterations as the aim is to be quicker than the already existing exact algorithms. So, to keep reasonable running times the iterative greedy algorithm runs $n/2$ iterations, where n is equal to the number of taxa in the tree. Moreover, as mentioned this algorithm is an advanced version of the simple greedy approach hence the first step of this algorithm is to reduce the common cherries in both trees unlike the greedy approach. This allows to reduce the dimensionality of the tree without impacting the TBR-distance and improve the approximation since there are fewer leaves to randomly pick.

C. Monte Carlo Tree Search

For clarity reasons in the following section the leaves and nodes of the MCTS search tree will be named MCTS_leaf and MCTS_node while the phylogenetic leaves and nodes are kept unchanged.

Monte Carlo Tree Search is a search algorithm based on a heuristic approach [8] it was implemented using the Upper Confidence bounds applied to Trees (UCT) algorithm for the node selection. The MCTS algorithm consists of four phases the selection, expansion, simulation, and backpropagation.

The selection, keeps selecting a child MCTS_node in the search tree until it reaches a MCTS_leaf node using the UCT algorithm for node selection. The idea is to create a new branch in the search tree for each unused leaf, leaf that is not assigned to a component yet.

The expansion, the MCTS_leaf node is then expanded to the next tree level by constructing all possible child nodes, again referring to the leaves not assigned yet.

The simulation, for the rest of the search tree, the leaves are selected at random until the end of the tree. Until there are no unattributed leaves left, and the TBR-distance of this branch is known.

The backpropagation, the final TBR-distance computed in the previous phase is then backpropagated to the current state and will serve as an indicator if the evaluated leaf is favorable.

The leaf selected is added to its according component and then the MCTS starts again from this state it is currently in, until every leaf of the phylogenetic tree has been added to a component.

D. Branching Algorithm

The last algorithm presented in this paper is a stochastic variation of a known exact algorithm by Whidden et al. [9]. At the beginning we have two trees T and T' on the same set of taxa X. As part of the algorithm consists of cutting edges in T' we will refer to it as F', a forest composed of subtrees of T' where $F' = T'$ at the beginning. In the first step of the algorithm, we reduce the common cherries of both trees T and F', then if there are no common cherries left find any cherry $\{x, y\}$ in tree T. As $\{x, y\}$ is not a cherry in tree F' there are two possibilities:

1) **x and y are in the same component of F'**
   If both x and y are contained in the same component then there exists a path from x to y, this path has at least 3 edges otherwise it would have been a cherry. Here comes the first stochastic part; there are three possibilities for the next step so we will guess one of them and never try out the two others. The first two guesses are similar as one of them deletes x and the other one y from both trees T and F'. After deleting x (resp. y) the adjacent vertex which has degree 2 is deleted as well while the TBR-distance is increased by 1 and then we go back to the cherry detection. The third possibility implies a second stochastic choice as we will randomly select an adjacent edge from the path connecting x to y to keep while deleting all the others adjacent edges. The TBR-distance will be increased by the number of edges deleted and both trees T and F' are tidied up in order to fulfill the conditions of phylogenetic trees.
2) **x and y are in different components of F'**
   If both x and y are in different components of F' then, one of the two following possibilities must be stochastically guessed. Either delete x from both T and F' or delete y. After deleting one of these leaves one needs to delete the adjacent vertex as well and increase the TBR-distance by one. Finally, go back to the cherry detection and continue until the tree T as at most 3 leaves left.

IV. Experiments

The experiments were conducted on a Windows 11 x64-based machine equipped with an Intel Core i7-9750 processor, clocked at 2.60GHz and with 16GB of RAM.

The experiments consisted of running the three algorithms (Iterative greedy, MCTS, Branching algorithm) on both datasets mentioned above, the set containing the 735 tree pairs and the set containing the 90 larger tree pairs. As the objective of this paper is to find good upper bounds, we mainly focused on comparing TBR-distance computed by these algorithms to the exact distance given in the dataset. To keep the tests within an acceptable time frame the execution times allocated for each algorithm were bound to 30 seconds for the trees with 50 taxa, 1 minute for 100 taxa, 2 minutes for 150 taxa and 3 minutes for the rest of the sets belonging to the first dataset. For the 90 larger tree pairs each algorithm was allotted 10 minutes. If the algorithm did not finish before the time limit, it was counted as a failed attempt.

V. Results

In the first dataset containing 735 trees a total of 29 trees (3.9%) could not be investigated as due to inconsistencies some trees did not follow the binary tree structure or they caused the program to crash. Moreover 15 other trees could not be computed as they exceeded the time limits for the Iterative Greedy and MCTS algorithms. These trees were removed from the further analysis.

The three diagrams shown in the figures 3, 4 and 5 display the results of the exact TBR-distance which is already known minus the TBR-distance computed by the algorithms. For example, every time the line touches the X-axis the difference is equal to 0 which means that the upper bounding algorithm computed the exact distance. Formally, the closer the line is to the X-axis, the closer the computed TBR-distance is to the known TBR-distance. The average difference for the Iterative Greedy algorithm is equal to 17.43 with a standard deviation of 14.15. The MCTS algorithm has an average distance of 23.35 with standard deviation 16.22 and the Branching algorithm has an average of 37.48 with standard deviation 20.73.

The next figures 6, 7 and 8 displays the frequency of the computed difference between the exact TBR-distance and the TBR-distance determined by the three algorithms. These histograms help to see which algorithm performed the best as the more it is right skewed the lower is the upper bound of the distance computed.

The final figure 9 plots two diagrams into one to visualize the relation between running time and TBR-distance computed. Only the trees with 50 taxa are taken into consideration and the running time of the Iterative Greedy algorithm.

For the second dataset the only approach able to compute an upper bound distance for every tree was the Branching Algorithm. The two other approaches were only able to compute 29 and 8 trees for the Iterative Greedy respectively MCTS out of the 90 trees.

VI. Discussion

First of all, the overall performance of the three algorithm can be summarized to; Iterative Greedy being the best upper bounding approach then the MCTS and finally the Branching Algorithm. However, each algorithm has its strength and weakness, the Iterative Greedy had the best results overall with an average

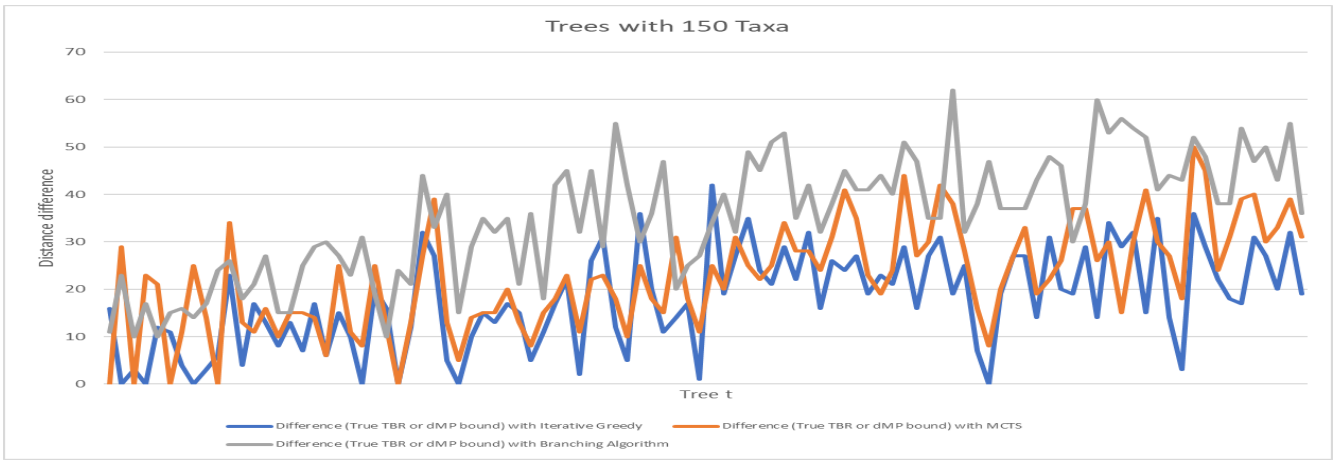

Fig. 3: Displays the difference (known TBR-distance minus computed TBR-distance) of each algorithm for the trees with 150 taxa. The trees are in order from lowest skew s and lowest TBR-operation performed k to highest s and highest k. The time allotted for these trees was set to 2 minutes.

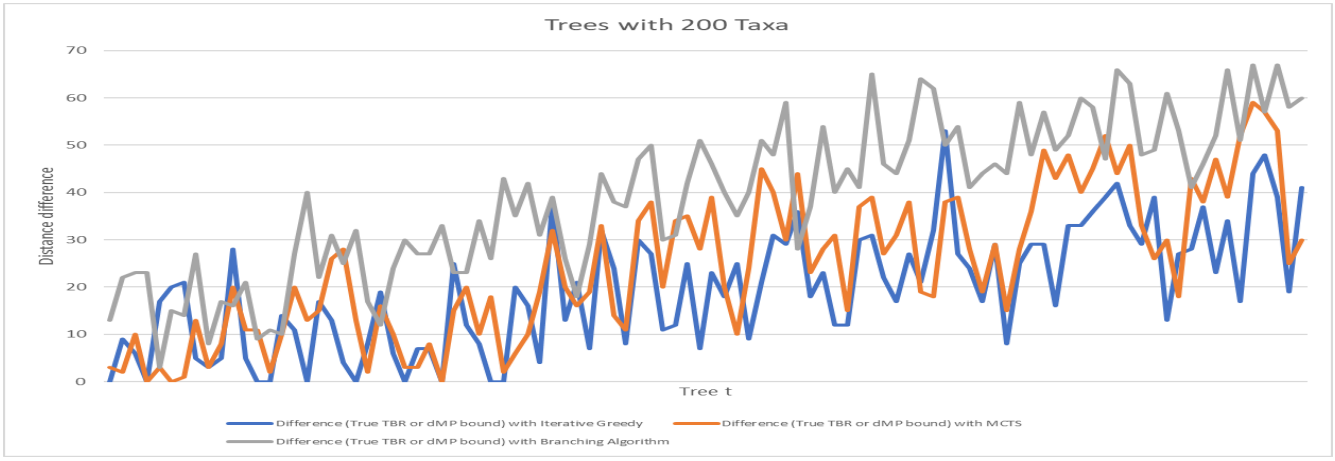

Fig. 4: Displays the difference (known TBR-distance minus computed TBR-distance) of each algorithm for the trees with 200 taxa. The trees are in order from lowest skew s and lowest TBR-operation performed k to highest s and highest k. The time allotted for these trees was set to 3 minutes.

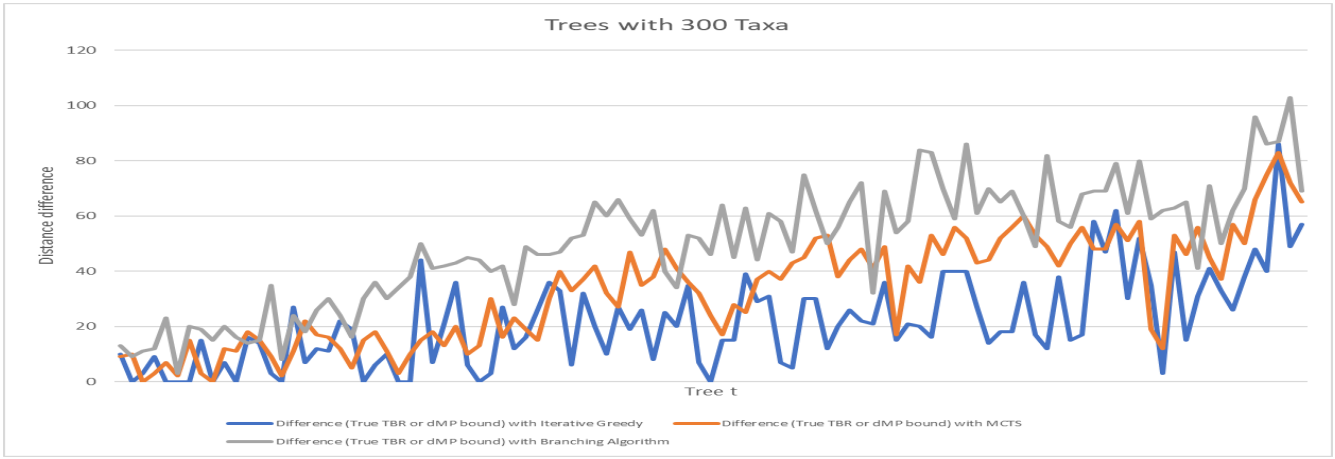

Fig. 5: Displays the difference (known TBR-distance minus computed TBR-distance) of each algorithm for the trees with 300 taxa. The trees are in order from lowest skew s and lowest TBR-operation performed k to highest s and highest k. The time allotted for these trees was set to 3 minutes.

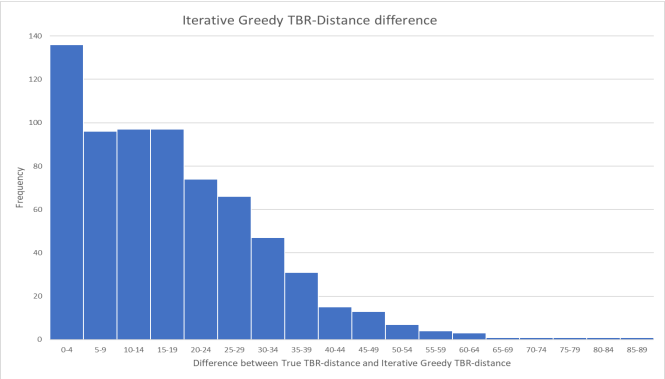

Fig. 6: Shows the frequency of the difference determined by the true TBR-distance minus the computed Iterative Greedy TBR-distance. This takes every tree from the first dataset into consideration.

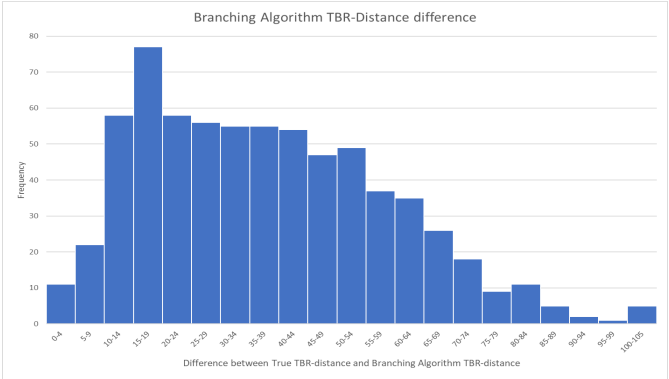

Fig. 8: Shows the frequency of the difference determined by the true TBR-distance minus the computed Branching Algorithm TBR-distance. This takes every tree from the first dataset into consideration.

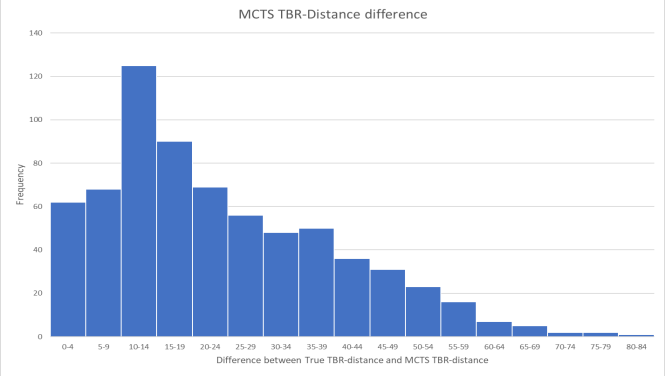

Fig. 7: Shows the frequency of the difference determined by the true TBR-distance minus the computed MCTS TBR-distance. This takes every tree from the first dataset into consideration.

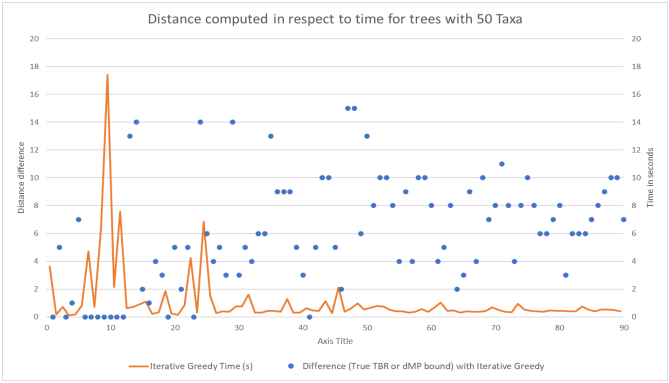

Fig. 9: Shows the relation between two plots, distance computed by the Iterative Greedy algorithm and the running time needed to compute it. Only considers trees with 50 taxa.

difference in distance of 17.43. The trees with the lowest skewness and the most effective cherry reduction, where the reduced tree had less taxa than the initial amount, the TBR-distance given by this approach was really competitive, often in the range $0-9$, as shown in the figure 6. For most of the trees the distance could be calculated in a time frame of 0.3 seconds and 120 seconds for larger trees. But for some trees (as shown in figure 9) the running time jumped when the algorithm seemed to find an exact or a relatively good distance. But for some rare trees with $|X| \geq 200$ taxa, the Iterative Greedy approach ran for longer then its time limit of 3 minutes because one of the iterations got stuck and found a good solution. So, after this iteration the algorithm was stopped even if it did not finish its $n/2$ iterations and 4 trees had to be stopped after 10 minutes. For the larger trees the iteration was limited to $n = 10$ and the time limit 10 minutes whatever happens so, only 29 trees out of the 90 could be computed and the average TBR-distance was 73.1. Here we output the distance and not the difference as the true TBR-distance is not known for this dataset but is ≤ 35 as the parameter to generate the trees was set to $k = 35$, k being the number of random TBR-operations performed on T to obtain T'.

The MCTS algorithm computed a TBR-distance which was mostly between the TBR-distance computed by the Iterative Greedy and the Branching algorithm as it can be deduced in the figures 3, 4 and 5. For this approach some running times did the same jump as for the previous one even though most of them did finish in the allotted time frame. These jumps in time could be explained by the fact that the expansion phase of the MCTS uses the greedy algorithm to perform a move when it selects a random leaf as the move needs to follow the same conditions when adding a leaf to a component. For the larger trees this approach performed quite poorly in terms of results found in the

10 minutes time frame as only 8 trees with an average distance of 79.5 could be computed.

The final algorithm, which is based on a stochastic Branching Algorithm was by far the quickest in terms of performance as every tree in the 735 dataset was computed in less than 0.1 seconds and for the larger trees with $|X| = 3000$ it ran in 0.3 seconds. But, it was also the worst performing one with an average difference in distance of 37.48 for the 735 trees and an average distance of 149.02 for the larger trees. Both of these means are quite far from the true TBR-distance as a consequence one could say that the Branching Algorithm is most of the time thrice if not more than the true TBR-distance.

VII. Conclusion

To conclude, this paper showed that if one wants good upper bounds the insights given by [2] about the Maximum Agreement Forest and the Tree Bisection and Reconnection distance was quite useful. An Iterative approach also permits to try out multiple agreement forests and helps to find the maximum agreement forest in some cases when the tree has the lowest skew and the lowest amount of taxa due to the cherry reduction. The reason why it might get stuck and run longer for some trees might be because of the topology check. As every subset of size 4 has to be verified and the lower the TBR-distance the larger are the subtrees with exponentially many subsets. This might also explain why the stochastic Branching Algorithm is the quickest in terms of performance as there are no subtrees to verify, every iteration is just a random choice which leads to a smaller tree. Also there is clearly a relation between cherry reduction and TBR-distance for the three algorithms so using other reduction techniques such as the one presented in [3] might improve these upper bounds but we suggest this as future work.

VIII. Acknowledgment

I would like to thank Associate Professor Steven Kelk for his time and continued support in this thesis and also for both datasets

References

[1] Joseph Felsenstein and Joseph Felenstein. *Inferring phylogenies*. Vol. 2. Sinauer associates Sunderland, MA, 2004.

[2] Benjamin L Allen and Mike Steel. "Subtree transfer operations and their induced metrics on evolutionary trees". In: *Annals of combinatorics* 5.1 (2001), pp. 1–15.

[3] Rim van Wersch et al. "Reflections on kernelizing and computing unrooted agreement forests". In: *Annals of Operations Research* 309.1 (2022), pp. 425–451.

[4] Chris Whidden and Frederick A. Matsen. *Calculating the Unrooted Subtree Prune-and-Regraft Distance*. 2015. DOI: 10.48550/ARXIV.1511.07529. URL: https://arxiv.org/abs/1511.07529.

[5] Jianer Chen, Jia-Hao Fan, and Sing-Hoi Sze. "Parameterized and approximation algorithms for maximum agreement forest in multifurcating trees". In: *Theoretical Computer Science* 562 (2015), pp. 496–512.

[6] Steven Kelk and Simone Linz. "New reduction rules for the tree bisection and reconnection distance". In: *CoRR* abs/1905.01468 (2019). arXiv: 1905.01468. URL: http://arxiv.org/abs/1905.01468.

[7] Estela Maris Rodrigues, Marie-France Sagot, and Yoshiko Wakabayashi. "The maximum agreement forest problem: Approximation algorithms and computational experiments". In: *Theor. Comput. Sci.* 374 (2007), pp. 91–110.

[8] Guillaume Chaslot et al. "Monte-Carlo Tree Search: A New Framework for Game AI". In: *Proceedings of the Fourth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AIIDE'08. Stanford, California: AAAI Press, 2008, pp. 216–217.

[9] Chris Whidden, Robert G Beiko, and Norbert Zeh. "Fixed-parameter algorithms for maximum agreement forests". In: *SIAM Journal on Computing* 42.4 (2013), pp. 1431–1466.

Appendix

The figures 10 to 14 are charts omitted from the main paper (but for which other, representative variants appear in the main paper). And figure 15 shows a correlation matrix that did not produce meaningful results and thus was omitted from the paper.

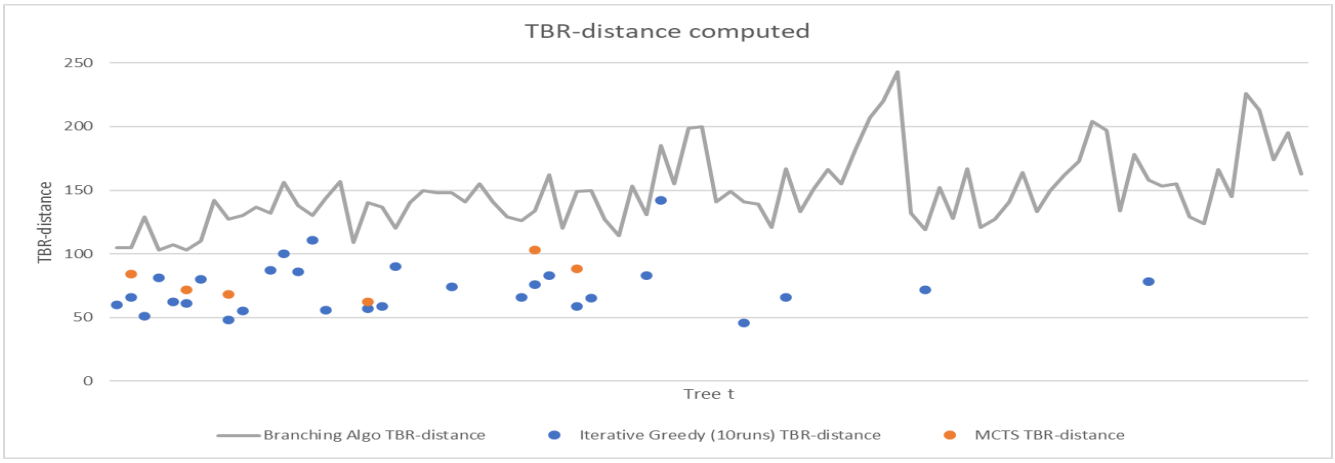

Fig. 10: Shows the TBR-distance computed by the three algorithm for the large trees with a time limit of 10 minutes. All 90 trees are in order from lowest skew s and lowest TBR-operation performed k to highest s and highest k

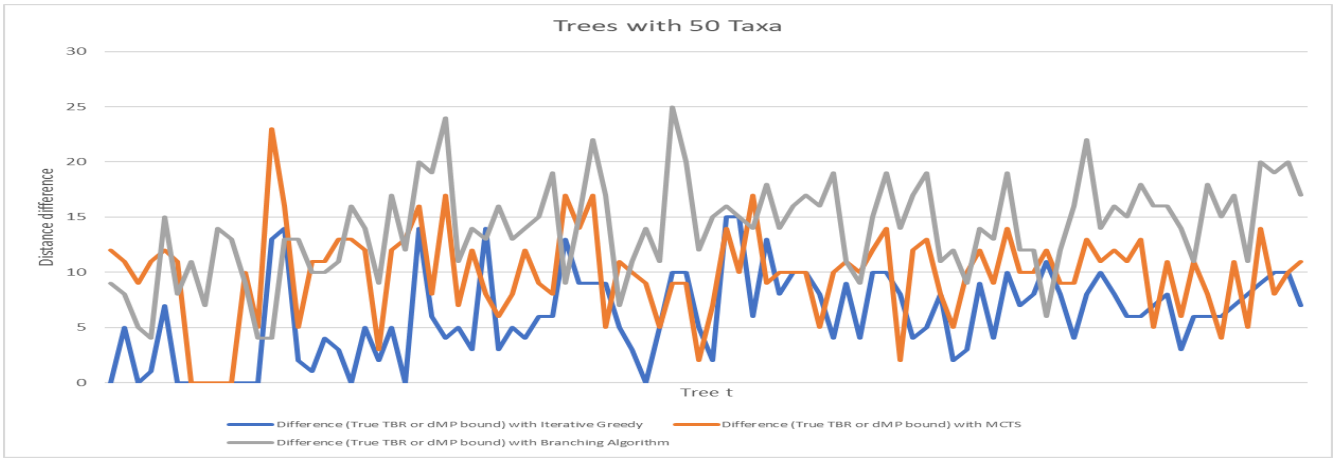

Fig. 11: Displays the difference (known TBR-distance minus computed TBR-distance) of each algorithm for the trees with 50 taxa. The trees are in order from lowest skew s and lowest TBR-operation performed k to highest s and highest k

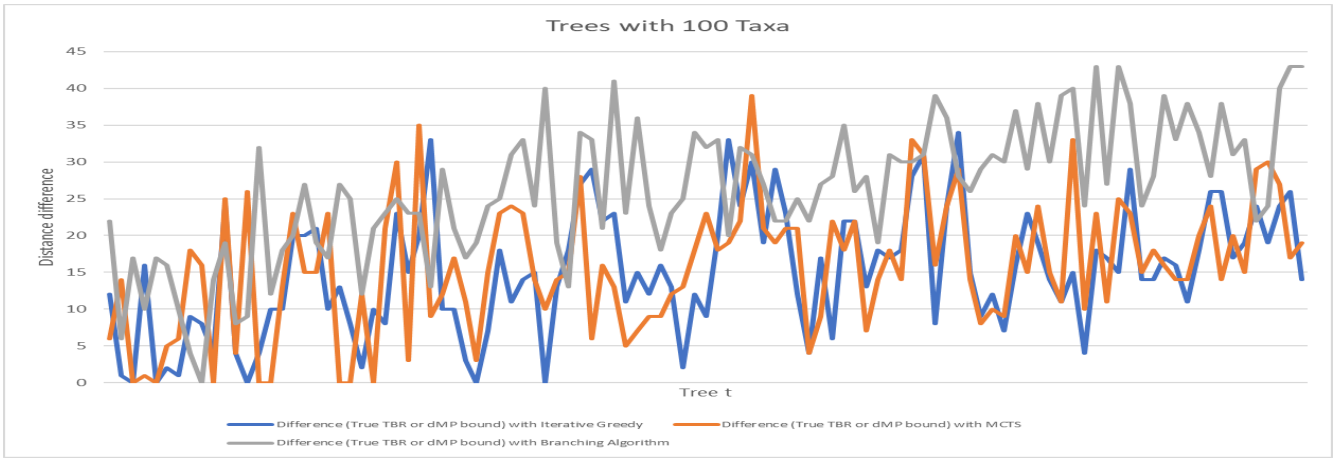

Fig. 12: Displays the difference (known TBR-distance minus computed TBR-distance) of each algorithm for the trees with 100 taxa. The trees are in order from lowest skew s and lowest TBR-operation performed k to highest s and highest k

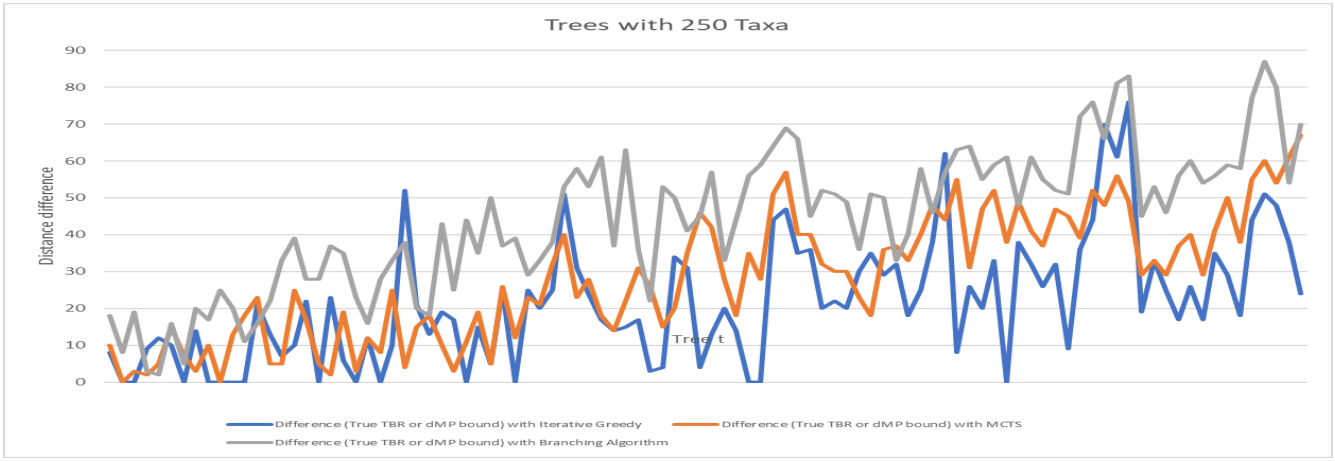

Fig. 13: Displays the difference (known TBR-distance minus computed TBR-distance) of each algorithm for the trees with 250 taxa. The trees are in order from lowest skew s and lowest TBR-operation performed k to highest s and highest k

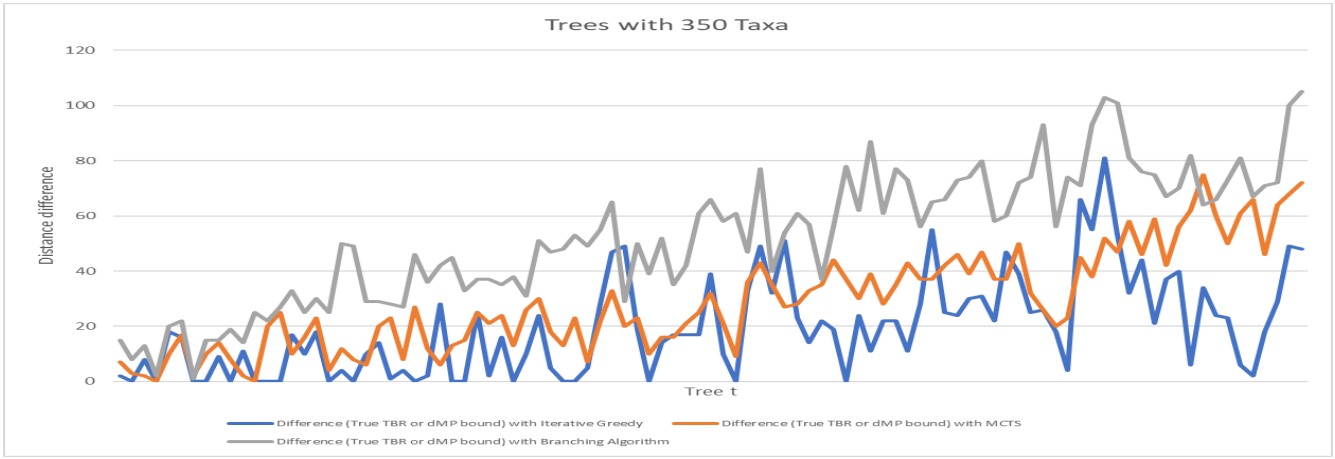

Fig. 14: Displays the difference (known TBR-distance minus computed TBR-distance) of each algorithm for the trees with 350 taxa. The trees are in order from lowest skew s and lowest TBR-operation performed k to highest s and highest k

	True TBR (if known), otherwise dMP bound	Iterative Greedy TBR-distance	Iterative Greedy Time (s)	MCTS TBR-distance	Branching Algo TBR	taxa	skew	taxa after cherry reduction	Difference TBR with Iterative Greedy	Difference TBR with MCTS	Difference TBR with Branching Algorithm
True TBR (if known), otherwise dMP bound	1	0.813276	0.030195	0.858166	0.834945	0.101454	4.38E-18	0.684259	0.533077	0.671776	0.67547
Iterative Greedy TBR-distance	0.813276	1	-0.03632	0.848837	0.817876	0.238119	0.197413	0.753461	0.925847	0.764091	0.735368
Iterative Greedy Time (s)	0.030195	-0.03632	1	0.16477	0.308057	0.660214	0.123788	0.485476	-0.07242	0.220645	0.39938
MCTS TBR-distance	0.858166	0.848837	0.16477	1	0.889339	0.334598	0.125262	0.823585	0.676898	0.956778	0.811222
Branching Algo TBR-distance	0.834945	0.817876	0.308057	0.889339	1	0.466661	0.129774	0.911982	0.646961	0.810258	0.969789
taxa	0.101454	0.238119	0.660214	0.334598	0.466661	1	0	0.683064	0.280343	0.425326	0.580305
skew	4.38E-18	0.197413	0.123788	0.125262	0.129774	0	1	0.194061	0.287044	0.180743	0.173883
taxa after cherry reduction	0.684259	0.753461	0.485476	0.823585	0.911982	0.683064	0.194061	1	0.651162	0.800742	0.918649
Difference TBR with Iterative Greedy	0.533077	0.925847	-0.07242	0.676898	0.646961	0.280343	0.287044	0.651162	1	0.674726	0.630563
Difference TBR with MCTS	0.671776	0.764091	0.220645	0.956778	0.810258	0.425326	0.180743	0.800742	0.674726	1	0.787882
Difference TBR with Branching Algorithm	0.67547	0.735368	0.39938	0.811222	0.969789	0.580305	0.173883	0.918649	0.630563	0.787882	1

Fig. 15: Displays the correlation matrix between different distances, running times and the parameters of the generated trees.