Implementation and Optimisations for Computing Maximum Agreement Forests for Rooted Multifurcating Trees

Ben Lee
Faculty of Computer Science
Dalhousie University
Halifax, Canada
Ben.Lee@dal.ca

Chris Whidden
Faculty of Computer Science
Dalhousie University
Halifax, Canada
cwhidden@dal.ca

***Abstract*—We implement and optimise a fixed parameter tractable (FPT) algorithm proposed by Whidden et al. (Algorithmica, 2016) for computing maximum agreement forests (MAFs) for pairs of multifurcating trees. The sizes of MAFs give the subtree-prune-and-regraft (SPR) distance and has uses in determining when and how often lateral gene transfer (LGT) takes place in phylogenetic trees. We show the running time based on synthetic data and confirm the optimisations speed up the algorithm. The main motive for this algorithm is to study LGT in antimicrobial resistant bacteria.**

***Index Terms*—subtree-prune-regraft, phylogenetic trees, multifurcating, non-binary**

Introduction

Lateral gene transfer (LGT) can rapidly spread antimicrobial resistance (AMR) in pathogenic bacteria, making it difficult to treat patients and requiring the expensive development of new antibiotics [2]. One method of studying and identifying LGT is to compare individual evolutionary gene trees to a reference "species tree" and reconcile differences between the trees [5]. The publicly available rspr software[1] computes such distances, using the biologically informative subtree prune-and regraft distance [3]. Computing maximum agreement forests (MAFs) is one way of calculating the subtree prune and regraft (SPR) distance [1]. However, the current implementation of rspr relies on an assumption that the reference tree is binary, that is, every node of the tree has at most two clear descendants. it is necessary to relax this assumption to analyze large AMR datasets of closely related bacteria, where the exact order of descendants may be unclear. This uncertainty is represented in the tree with nonbinary nodes. In this paper we extended rspr to handle nonbinary-nonbinary comparisons, by implementing a proposed algorithm from Whidden et al. (Algorithmica, 2016) [6]. We then show a set of optimisations that experimentally decrease the running time on synthetic trees.

This project was funded by the Natural Sciences and Engineering Research Council

[1]https://github.com/cwhidden/rspr

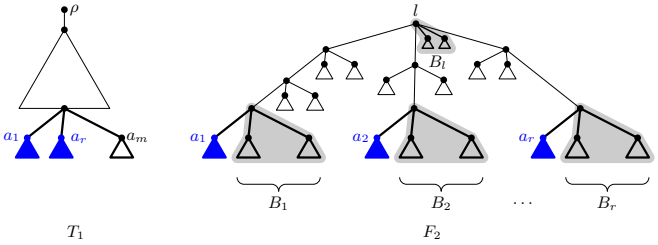

Fig. 1. Figure of a sibling group on T_1 (left) and relative nodes in a subtree of Forest F_2, created from previous cuts of T_2 (right)

Methods

This algorithm is based on finding MAFs because the size of an MAF gives the SPR distance between two trees. In order to compute an MAF, edges between nodes are cut until each forest are identical, and the goal is to cut as few edges as possible so the forests have maximum agreement. Let the reference tree be denoted by T_1 and the target tree denoted by T_2. The algorithm works by identifying sibling groups in T_1, where a sibling group is a set of leaves that share a parent, then using the relative position of those siblings in T_2 to identify possible cuts that lead to an optimal solution. Proven in [6] and shown in Fig. 1, there are four possible cuts that reduces the distance by at least 1 while working on two siblings. These cuts are: cutting the edge above a_1, a_2, expanding B_1 out to a single node and cutting that, and doing the same for B_2

This algorithm is recursive with a fixed parameter k, starting at 0 and increasing until the SPR distance is less than or equal to k. This does not significantly increase the running time because the algorithm is exponential on k [6].

As mentioned in [6], the naïve implementation would simply recurse on the 4 different cuts. However, there may be a case where in one branch edge e_a is cut, then e_b; then in another branch edge e_b is cut, then e_a. These two branches result in the same tree and it would be best to only go down one of them. A set of special cases were proposed to reduce recursive calls like this [6]. Cases 8.1-8.7 reduce the number of recursive calls made by making a series of cuts all at once.

TABLE I

TEST PAIRS OF TREES WITH THE STATED SPR DISTANCE (DSPR) THAT CAN BE BROKEN INTO (CLUSTER COUNT) CLUSTERS. RUNNING TIME OF THE NEW ALGORITHM IN SECONDS IS SHOWN USING BOTH OPTIMISATIONS OF THE SPECIAL CASES 7/8 AND CLUSTERING, JUST THE SPECIAL CASES, JUST CLUSTERING OR NEITHER OPTIMISATION. IF THE ALGORITHM DID NOT FINISH IN 30 MINUTES IT IS LABELED N/A.

dSPR	Cluster Count	Special cases w/ cluster(s)	Special cases w/o cluster(s)	No Special cases w/ cluster(s)	No special cases w/o cluster(s)
10	1	0.032	0.031	0.159	0.162
15	1	1.117	1.122	14.338	14.087
17	3	0.199	0.318	1.698	6.265
24	1	3.503	4.025	868.792	838.446
46	25	0.146	N/A	0.728	N/A

Cases 7.1-7.4 help prevent redundant branching by marking an edge not to be cut in the next iteration. Similar cases were implemented for binary trees [7] and were shown to improve the running time by several orders of magnitude, and we expect it to have the same effect on multifurcating trees. The multifurcating cases are explained in detail in [6].

Another optimisation we implemented was cluster reduction developed by Linz and Semple [4]. This involves finding pairs of subtrees in T_1 and T_2 that have the same subset of leaves but not necessarily the same branching. We can solve these subtrees independently of the rest of the tree. Since the algorithm is exponential on k, this can greatly decrease the running time by solving many small trees with low SPR distance instead of one large tree.

We made the following changes while porting the current binary-nonbinary rspr software to support nonbinary-nonbinary trees: 1. storing sibling groups, 2. finding lowest common ancestors (LCAs) of sibling groups and 3. finding identical sibling groups.

Firstly, in the binary implementation sibling pairs were stored as a pair of pointers to nodes. Instead of storing the siblings in a list, we save sibling groups by storing their parent. This way a single pointer to a node is stored instead of an entire list of pointers which saves storage and time.

Secondly, in order to determine which special case to consider (7.1-7.4 and 8.1-8.7), the LCA of the sibling group in T_2 must be found. We find LCA's in an efficient manner by traversing upward from each node of the sibling group in T_2. First, each node in T_2 is labeled with 0. Then we follow each node of the sibling group to its relative in T_2 and we traverse up, adding 1 to that node's label until a root is found. Then an LCA can be described as a node with a label value greater than or equal to 2, and all children with labels 0 or 1.

Thirdly, we implemented a way to find identical sibling groups. Identical sibling groups are subsets of the sibling group that share a parent in both T_1 and T_2. These are useful to find because it means these nodes can be contracted into a single node. Let twins be a pair of leaves, where one node is in T_1, one node is in T_2, and the node in T_2 represents the same organism as the node in T_1. These twins are maintained for all leaves throughout the algorithm. We create a map with the key as the parents in T_2 and the values as a the list of children that are the twins in T_2. Each sibling in T_1 is followed to its twin in T_2, if its parent is not in the map, it is added and the value list adds this twin. If it is in the map then this twin is appended into the list. After all the siblings are accounted for, these lists contain disjoint subsets of the sibling group that share a parent in T_2. If a list only has one node, then it is not part of an identical sibling group because it does not share a parent with any other sibling in T_2. If it has more than one node then it is part of an identical sibling group.

With these three major changes, we were able to implement the proposed nonbinary-nonbinary algorithm.

RESULTS

The following tests were run on a 2.3 GHz Dual-Core Intel i5 with 8GB of RAM and running macOS 11.0.1 Big Sur on a 2017 MacBook Pro. The code was compiled using clang 12.0.0 with optimisations -O2. Table I shows the running times of test trees with different optimisations enabled. Trees were made by generating a random tree with each branch containing 3 to 5 children.

From Table I, special cases 7 and 8 generally improved all of the test's running time. The most dramatic improvement was the cluster reduction, where a tree with a dSPR of 46 was able to be computed in negligible time with the cluster reduction. The cluster reduction understandably showed no difference when there was only one cluster because that cluster contains the whole trees.

Interesting to note from Table I is the 4th data point with a SPR distance of 24. The special cases gave a speed up of about 224 times. We expect to see similar or greater speed up from these cases in larger trees with larger SPR distances.

CONCLUSIONS

In this paper we present the results of implementing the proposed algorithm from Whidden et al. (Algorithmica, 2016) to compute the SPR distance between a pair of nonbinary trees [6]. We show that trees with an SPR distance of 46 can be calculated in under a second if they have many clusters, and trees with an SPR distance of 15 can be calculated in about a second with the special cases optimisation. We also show that the special cases can improve a tree with an SPR distance of 24 by approximately 224 times. In this project we created the first implementation of an algorithm for computing the SPR distance between two nonbinary trees.

FUTURE WORK

The next step for this project is to run the algorithm on real phylogenetic trees of AMR bacteria from the ARETE database

(http://arete-amr.ca). We would also like to work with ARETE researchers to analyse an AMR dataset using this algorithm.

The current software only shows the SPR distance, so another future work is to add an option that shows what the SPR operations were. Using the MAF calculated, it is possible to find the sequence of SPRs in polynomial time [1].

We would also like to try other optimisations that are not listed in [6], for example putting the sibling groups in a sorted queue based on number of siblings or depth then working on sibling groups in that order.

ACKNOWLEDGMENT

This project is supported by an NSERC Undergraduate Student Research Award.

REFERENCES

[1] Bordewich, M., Semple, C. On the Computational Complexity of the Rooted Subtree Prune and Regraft Distance. Ann. Comb. 8, 409–423 (2005). https://doi.org/10.1007/s00026-004-0229-z

[2] M. Gajdács, E. Urbán, A. Stájer, and Z. Baráth, "Antimicrobial Resistance in the Context of the Sustainable Development Goals: A Brief Review," European Journal of Investigation in Health, Psychology and Education, vol. 11, no. 1, pp. 71–82, Jan. 2021.

[3] Hein, J., Jiang, T., Wang, L., Zhang, K.: On the complexity of comparing evolutionary trees. Disc. Appl. Math. 71(1-3), 153–169 (1996)

[4] Linz,S., Semple,C.:Hybridization in nonbinary trees. IEEE/ACMTrans. Comput. Biol. Bioinform. 6, 30–45 (2009)

[5] Ravenhall M, Škunca N, Lassalle F, Dessimoz C. "Inferring horizontal gene transfer". PLoS Comput Biol. 2015;11(5):e1004095, 2015 May 28. doi:10.1371/journal.pcbi.1004095

[6] Whidden, C., Beiko, R.G. & Zeh, N. Fixed-Parameter and Approximation Algorithms for Maximum Agreement Forests of Multifurcating Trees. Algorithmica 74, 1019–1054 (2016). https://doi.org/10.1007/s00453-015-9983-z

[7] Whidden C., Beiko R.G., Zeh N. (2010) Fast FPT Algorithms for Computing Rooted Agreement Forests: Theory and Experiments. In: Festa P. (eds) Experimental Algorithms. SEA 2010. Lecture Notes in Computer Science, vol 6049. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13193-6_13