SPR Distance Computation for *Unrooted* Trees

Glenn Hickey [*] Frank Dehne [†] Andrew Rau-Chaplin [‡] Christian Blouin [§]

Abstract: The subtree prune and regraft distance (d_{SPR}) between phylogenetic trees is important both as a general means of comparing phylogenetic tree topologies as well as a measure of lateral gene transfer (LGT). Although there has been extensive study on the computation of d_{SPR} and similar metrics between *rooted* trees, much less is known about SPR distances for *unrooted trees*, which often arise in practice when the root is unresolved. We show that unrooted SPR distance computation is NP-Hard and verify which techniques from related work can and cannot be applied. We then present an efficient heuristic algorithm for this problem and benchmark it on a variety of synthetic datasets. Our algorithm computes the *exact* SPR distance between unrooted tree, and the heuristic element is only with respect to the algorithm's computation time. Our method is a heuristic version of a fixed parameter tractability (FPT) approach and our experiments indicate that the running time behaves similar to FPT algorithms. For real data sets, our algorithm was able to quickly compute d_{SPR} for the majority of trees that were part of a study of LGT in 144 prokaryotic genomes. Our analysis of its performance, especially with respect to searching and reduction rules, is applicable to computing many related distance measures.

Keywords: Unrooted trees, SPR distance, lateral gene transfer, phylogenetic tree metrics

1 Introduction

Phylogenetic trees are used to describe evolutionary relationships. DNA or protein sequences are associated with the leaves of the tree and the internal nodes correspond to speciation or gene duplication events. In order to model ancestor-descendant relationships on the tree, a direction must be associated with its edges by assigning a root. Often, insufficient information exists to determine the root and the tree is left unrooted. Unrooted trees still provide a notion of evolutionary relationship between organisms even if the direction of descent remains unknown.

The phylogenetic tree representation has recently come under scrutiny with critics claiming that it is too simple to properly model microbial evolution, particularly in the presence of lateral gene transfer (LGT) events (Doolittle 1999). A LGT is the transfer of genetic material between species by means other than inheritance and thus cannot be represented in a tree as it would create a cycle. The prevalence of LGT events in microbial evolution can, however, still be studied using phylogenetic trees. Given a pair of trees describing the same sets of species, each constructed using different sets of genes, a LGT event corresponds to a displacement of a common subtree, referred to as a SPR operation. The SPR distance is the minimum number of SPR operations, denoted by d_{SPR}, that explain the topological differences between a pair of trees. It is equivalent to the number of transfers in the most

[*]**Corresponding Author.** School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6. Tel: +1 (613) 520-2600 ext.4588, Email: ghickey@scs.carleton.ca

[†]School of Computer Science, Carleton University, Ottawa, Canada, http://www.dehne.net

[‡]Faculty of Computer Science, Dalhousie University, Halifax, Canada, http://users.cs.dal.ca/∼arc

[§]Faculty of Computer Science, Dalhousie University, Halifax, Canada, cblouin@cs.dal.ca

parsimonious LGT scenario (Beiko & Hamilton 2006). In general, d_{SPR} can be used as a measure of the topological difference between two trees, e.g. for comparing the outputs of different tree construction algorithms.

Tree bisection and reconnection (TBR) is a generalization of SPR that allows the pruned subtree to be rerooted before being regrafted. Computation of the TBR distance (d_{TBR}) was shown to be NP-hard by (Allen & Steel 2001), who also provided two rules that reduce two input trees to a size that is a linear functions of d_{TBR} without altering their distance. These rules, which reduce common chains and subtrees, also form the basis of algorithms that compute the SPR distance between rooted trees (d_{rSPR}) (Bordewich & Semple 2004) as well as hybridization number (h) (Bordewich et al. 2007), see Section 3.3. Such algorithms proceed as follows. First the distance problem is shown to be equivalent to counting components of a maximum agreement forest, and then it is shown that the application of the rules do not alter the number of components in the forest. These steps have been successfully applied to d_{TBR}, d_{rSPR} and h but not d_{SPR}, for which no equivalent agreement forest problem is known. As a consequence, the computational complexity of d_{SPR} has remained an open problem. We provide a proof of NP-Hardness in Section 2. In Section 3, we present an efficient algorithm that relies only on the subtree reduction rule to compute the SPR distance of unrooted trees. An implementation of this algorithm was tested on a variety of data, and the results are analyzed in Section 4. In particular, we show that the conjecture that chain decomposition is d_{SPR}-preserving for unrooted trees (Allen & Steel 2001) is strongly supported by our data.

The contributions of this paper can be summarized as follows: (1) We show that SPR distance computation is NP-hard for unrooted trees. (2) We present an efficient heuristic algorithm for this problem and benchmark it on a variety of synthetic datasets. Our algorithm computes the *exact* SPR distance between unrooted trees, and the heuristic element is only with respect to the algorithm's computation time. Our method is a heuristic version of a fixed parameter tractability (FPT) approach (Downey & Fellows 1998) and our experiments indicate that the running time behaves similar to FPT algorithms. For real data sets, our algorithm was able to quickly compute d_{SPR} for the majority of trees that were part of a study of LGT in 144 prokaryotic genomes. (3) Our analysis of its performance, especially with respect to searching and reduction rules, is applicable to computing many related distance measures. (4) In (Bordewich et al. 2007), a decomposition by common clusters was used with significant practical success. We show that such a decomposing by common clusters cannot be used to compute exact SPR distance for *unrooted* trees (Figure 4) which is somewhat counterintuitve.

2 SPR Distance Computation is NP-Hard for Unrooted Trees

In (Hein et al. 1996), it was shown that computing the size of a the Maximum Agreement Forest (MAF) of two trees is NP-Hard by reducing it from Exact Cover of 3-Sets (X3C). Later, (Allen & Steel 2001) proved that this result is insufficient to show the hardness of unrooted SPR distance because there is no direct relationship between MAF size and d_{SPR}, as was previously claimed. Similar techniques have since been used in (Bordewich & Semple 2004) to show that rooted SPR distance is NP-Hard via reduction from X3C to a rooted version of MAF. We show that although d_{SPR} cannot be used to compute $|MAF|$ in general, it can for the trees used in the polynomial-time reduction from X3C and this is sufficient to show that d_{SPR} is NP-Hard as well. We begin with two preliminary definitions

Definition 2.1. An *agreement forest* for two trees is any common forest that can be obtained from both trees by cutting the same number of edges from each tree, applying forced contractions after each cut. A *maximum agreement forest* (MAF) for two trees is an agreement forest with a minimum number of components. (Hein et al. 1996)

Definition 2.2. The *exact cover by 3-sets* (X3C) problem is defined as follows (Garey & Johnson 1979): Given a set X with $|X| = n = 3q$ and a collection C of m 3-element subsets of X. Does C contain an exact cover for X, ie, a sub-collection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

NOTE: This problem remains NP-Complete if no element occurs in more than three subsets. Also note that this problem remains NP-Complete if each element occurs in *exactly* three subsets. This second property is implied by (Hein et al. 1996) though never explicitly stated. A supplemental proof is provided in Appendix A.

We now review the polynomial-time reduction from X3C to MAF provided by (Hein et al. 1996), clarifying their notation to reflect that each element of X belongs to *exactly* three subsets in C, ie $|X| = |C| = 3q = m = n$, a fact implied but not clearly stated in their paper. An instance of X3C is transformed into two rooted phylogenetic trees shown in Figure 1. Each element of X is represented by a triplet of the form $\{a, u, v\}$ and each triplet appears 3 times in each tree, once for each occurrence in a subset in C. Tree T_1 is illustrated in Figure 1(a). Each subtree $A_i \in T_1$, shown in Figure 1(b) corresponds to a subset $c_i \in C$. Each subtree of A_i induced by the triple $\{a_{i,j}, u_{i,j}, v_{i,j}\}$ where $j \in \{1, 2, 3\}$ corresponds to a single element of X.

Tree T_2, shown in Figure 1(c), has the same leaf set as T_1 but a different topology. Each D_i subtree of T_2, as seen in Figure 1(e), corresponds to a subset in C except only the a-part of each triplet is present. Each B_i subtree of T_2, as seen in Figure 1(d), corresponds to an element in X. Each such element $x = \{a, u, v\}$ in the set X appears in three different subsets of C: c_j, c_k, and c_l. Without loss of generality, assume it consists of the first element of c_j, second element of c_k, and third element of c_l. The corresponding B tree would have leaves $\{u_{j,j'}, u_{k,k'}, u_{l,l'}, v_{j,j'}, v_{k,k'}, v_{l,l'}\}$ where $j' = 1, k' = 2, l' = 3$.

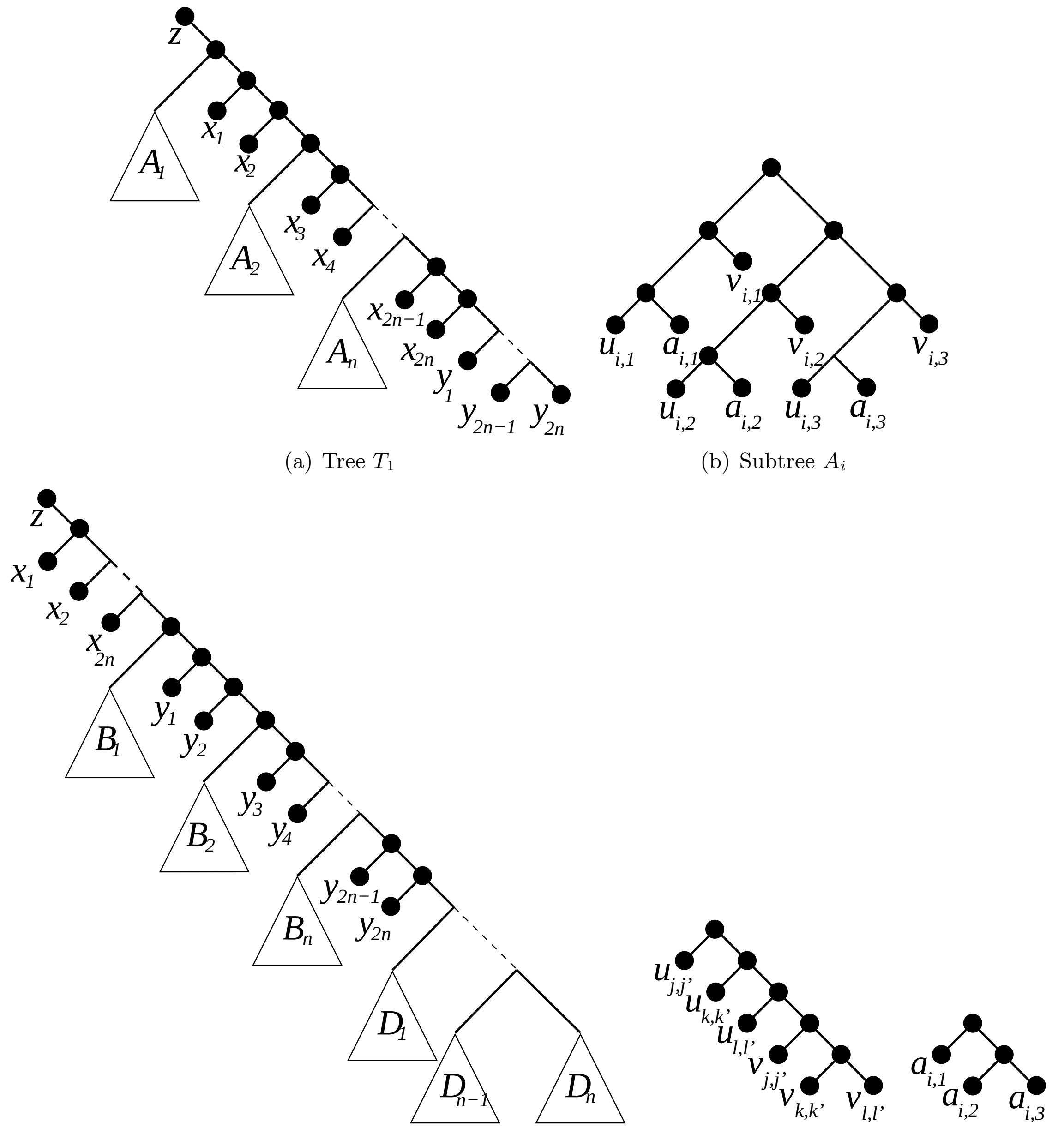
(Hein et al. 1996) show that $|MAF(T_1, T_2)| = 20q + 1$ if and only if C contains an exact cover of X. Note that we have added the z leaf to these trees, unrooting them. This does not have any affect on the $|MAF|$ as z can remain attached to x_1 in the agreement forest without the addition of any new components.

Proving that $d_{SPR}(T_1, T_2) = |MAF(T_1, T_2) - 1|$ is sufficient to transform any instance of X3C where $|X| = |C| = 3q$ to an instance of d_{SPR}. In fact, it is sufficient to show that the inequality $d_{SPR}(T_1, T_2) \leq |MAF(T_1, T_2) - 1|$ is true as $d_{SPR}(T_1, T_2) \geq |MAF(T_1, T_2) - 1|$ follows from Lemma 2.7(b) and Theorem 2.13 from (Allen & Steel 2001). We proceed much in the same way as the original proof, noting that each SPR operation used to transform to T_1 to T_2 corresponds to a cut required to form their MAF.

$MAF(T_1, T_2)$ is formed by the cutting edges from A_i subtrees (and the corresponding subtrees in T_2) in either of two possible ways (Hein et al. 1996):

1. Cut leaves $u_{i,1}, v_{i,1}, u_{i,2}, v_{i,2}, u_{i,3}, v_{i,3}$ and then prune the remaining subtree formed by leaves $\{a_{i,1}, a_{i,2}, a_{i,3}\}$. Such a procedure contributes 7 components to the MAF.

2. Cut the leaves $a_{i,1}, a_{i,2}, a_{i,3}$ then cut each of the remaining two-leaf subtrees: $\{u_{i,1}, v_{i,1}\}$, $\{u_{i,2}, v_{i,2}\}$, and $\{u_{i,3}, v_{i,3}\}$. These operations contribute 6 components to the MAF



(a) Tree T_1

(b) Subtree A_i

(c) Tree T_2

(d) Subtree B_i

(e) Subtree D_i

Figure 1: Reduction of an instance of X3C to $|MAF(T_1, T_2)|$ from an $\{a, u, v\}$ triplet. The instance of X3C has a solution if and only if $|MAF(T_1, T_2)| = 20q + 1$ (where $n = 3q$).

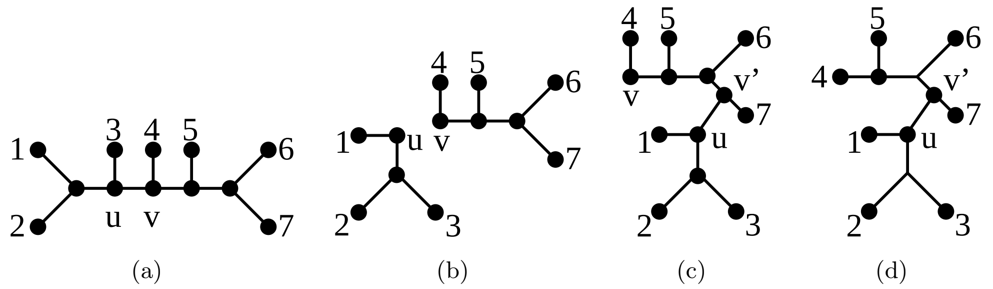


Figure 2: (a) Original tree. (b) Edge uv is removed, pruning subtree rooted at u. (c) Subtree is regrafted, creating new vertex v'. (d) Degree-2 vertex v is contracted.

We now show that given two trees T_1 and T_2 and their MAF, which was created using the above cut operations, there exists $|MAF| - 1$ SPR operations that can transform T_1 to T_2. In particular, for each set of cut operations, there exists an equivalent set of SPR operations.

1. Prune leaves $u_{i,1}, v_{i,1}, u_{i,2}, v_{i,2}, u_{i,3}, v_{i,3}$ from A_i and regraft them onto the chain, forming B_i subtrees in the required positions. Prune the subtree $\{a_{i,1}, a_{i,2}, a_{i,3}\}$ and regraft into the position of D_i. In this case, 7 SPR operations are performed.

2. Prune the leaves $a_{i,1}, a_{i,2}, a_{i,3}$ and regraft them onto the chain, forming a D_i subtree in the proper position. Prune the remaining two-leaf subtrees: $\{u_{i,1}, v_{i,1}\}$, $\{u_{i,2}, v_{i,2}\}$, and $\{u_{i,3}, v_{i,3}\}$ and regraft them onto the chain, forming B_i subtree components in the required position. 6 SPR operations are used.

There is a one-to-one correspondence between cuts formed when creating the MAF and SPR operations that can transform T_1 to T_2. Thus $d_{SPR}(T_1, T_2) \leq |MAF(T_1, T_2)| - 1$ and the proof is completed.

3 Algorithm for $\mathbf{d_{SPR}}$ Computation

3.1 Definitions

All trees referred to in this paper, unless otherwise stated, are unrooted binary phylogenetic trees. Such trees have interior vertices of degree 3 and uniquely labeled leaves. Given a tree T, let $V(T)$, $L(T)$ and $E(T) \in \{V(T) \times V(T)\}$ be the vertex, leaf, and edge sets of T respectively. A tree can be rooted by adding a root vertex of degree 2. A pendant subtree of T is any rooted tree T' such that $V(T') \subseteq V(T), L(T') \subseteq L(T)$ and $E(T') \subseteq E(T)$. A SPR operation on a tree T is defined by the following three steps, illustrated in Figure 2. First, an edge $\{u, v\} \in E(T)$ is removed, effectively pruning a pendant subtree rooted at u from T. A new interior vertex w is created by subdividing an edge in T and the subtree is then regrafted by creating edge $\{u, w\}$. Finally, the degree-2 vertex v is contracted by identifying its incident edges. The SPR distance between T_1 and T_2, denoted $d_{SPR}(T_1, T_2)$, is the minimum number of SPR operations required to transform T_1 into T_2. Furthermore, d_{SPR} is a metric (Allen & Steel 2001).

3.2 Exhaustive Search

The reduction rules referred to above only serve to transform the original problem into smaller subproblems. These subproblems must still be solved with an exhaustive search as the problem is NP-Hard (see proof in Appendix). Let $G_{SPR}(n)$ be the graph such that each vertex in the graph is associated with a unique tree topology with n leaves, and all possible topologies are in the graph. A pair of vertices in this graph are connected if their SPR distance is 1. Computing $d_{SPR}(T_1, T_2)$ is therefore equivalent to finding the length of the shortest path between T_1 and T_2 on $G_{SPR}(n)$ and can be computed through an exhaustive breadth-first search beginning at T_1. In (Allen & Steel 2001), it was shown that each tree will have $O(n^2)$ neighbors in the graph and it follows that the search will visit $O(n^2)$ trees of distance 1 from T_1, $O(n^4)$ trees of distance 2, up to $O(n^{2k})$ trees of distance k. A hash table is kept to ensure the same tree is not visited twice. Assuming that it can be updated in constant time, each tree can be processed in $O(n)$ bringing the time and space complexity of the search to $O(n^{2k+1})$.

While it is still an open problem to determine if reduction rules can be found to reduce n to k in the asymptotic complexity above, the value of the exponent can be reduced significantly. Observe that there must be some tree T' such that $d_{SPR}(T_1, T') = \lfloor k/2 \rfloor$ and $d_{SPR}(T_2, T') = \lceil k/2 \rceil$ because d_{SPR} is a metric and therefore satisfies the triangle inequality. T' and, correspondingly, k can be computed by performing two breadth-first searches, with origins at T_1 and T_2 simultaneously. During the i^{th} iteration of the search, all trees of distance i from first T_1 then T_2 are explored and updated into the same hash table. T' is the first tree to be found by both searches and $d_{SPR}(T_1, T_2)$ is $2i - 1$ if T' is found in the search for T_1 or $2i$ otherwise. Pseudocode is given in Algorithm 1. The time complexity of this algorithm is $O(n^{\lfloor k/2 \rfloor + 1}) + O(n^{\lceil k/2 \rceil + 1}) = O(n^{k+2})$. This is a significant reduction from the simple search but the complexity is still prohibitive. Fortunately, heuristics can greatly speed up many instances of the problem while still guaranteeing an exact answer.

3.3 Heuristic Improvements

A subtree reduction replaces any pendant subtree that occurs in both input trees by a single leaf with a new label in each tree as as shown in Figure 3(a). A chain reduction, illustrated in 3(b), replaces any chain of pendant subtrees that occur identically in both trees by three new leaves with new labels correctly oriented to preserve the direction. (Allen & Steel 2001) showed that maximum application of both of these rules reduces the size of the input trees to a linear function of d_{TBR}. This result also holds for d_{SPR} as $d_{SPR} \leq 2d_{TBR}$ for two trees since each TBR operation can be replaced by 2 SPR operations. It is trivial to show that subtree reductions do not alter d_{SPR} but, unlike d_{TBR} it is presently unknown whether or not chain reductions affect d_{SPR}, therefore they can not be used in an exact algorithm. However, our experimental results, further described in Section 4, do support the conjecture that chain reductions do not affect SPR distance.

In addition to applying reductions on the input trees, intermediate trees visited during the breadth-first search can be likewise reduced. For example, if T^* is a tree found on the i^{th} iteration from T_1 that has a common pendant subtree with T_2, then that subtree can be reduced to a leaf in T^* and T_2 without affecting $d_{SPR}(T^*, T_2)$. Accordingly, the shortest path from T_1 to T_2 will still be found by a search that applies subtree reductions to the intermediate trees. For ease of maintaining the hash table of trees visited, in our implementation we flag common subtrees rather than remove them and use these flags to

Algorithm 1 SPRDIST(T_1, T_2)

```
 1: if $T_1 = T_2$ then
 2:     return 0
 3: end if
 4: Apply subtree reductions to $T_1$ and $T_2$
 5: $d \leftarrow 0$
 6: $H \leftarrow$ empty hash table
 7: $\mathcal{L}_1, \mathcal{L}_A \leftarrow$ empty lists
 8: Insert $T_1$ into $\mathcal{L}_1$
 9: Insert $T_2$ into $\mathcal{L}_A$
10: loop
11:     $\mathcal{L}_2, \mathcal{L}_B \leftarrow$ empty lists
12:     if ITERATE($\mathcal{L}_1$, $\mathcal{L}_2$, $H$, $T_2$) = TRUE then
13:         return $d$
14:     else
15:         $\mathcal{L}_1 \leftarrow \mathcal{L}_2$
16:         $d \leftarrow d + 1$
17:     end if
18:     if ITERATE($\mathcal{L}_A$, $\mathcal{L}_B$, $H$, $T_1$) = TRUE then
19:         return $d$
20:     else
21:         $\mathcal{L}_A \leftarrow \mathcal{L}_B$
22:         $d \leftarrow d + 1$
23:     end if
24: end loop
```

Algorithm 2 ITERATE($\mathcal{L}_{in}$, $\mathcal{L}_{out}$, H, T)

```
1: for all $t \in \mathcal{L}_{in}$ do
2:     if $t \in H$ then
3:         return TRUE
4:     else
5:         Append set of SPR neighbors of $t$ to $\mathcal{L}_{out}$
6:         Insert $t$ into $H$
7:     end if
8: end for
9: return FALSE
```

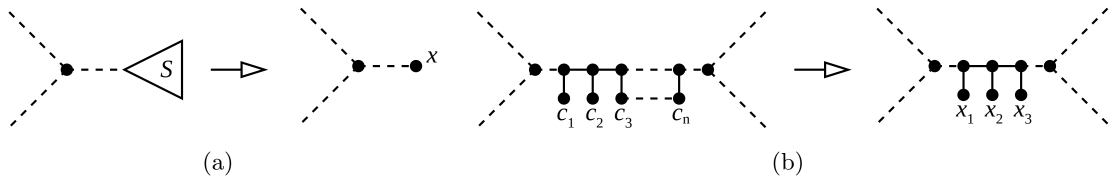


Figure 3: Reduction rules applied to a tree. (a) A subtree is reduced to a leaf. (b) A chain of length n is reduced to a chain of length 3.

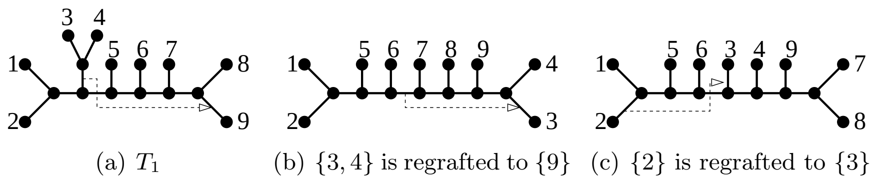
avoid performing SPR operations that would prune from or regraft to flagged subtrees. This process has no adverse effect on the asymptotic complexity of the search as common subtrees and chains can be detected in $O(n)$. It is expected that performing reductions on intermediate trees will lessen the total number of trees searched but we are unable to show that it will affect the worst case complexity.

Because the number of trees visited in each iteration of the exhaustive search increases exponentially, the asymptotic complexity is bounded by the number of trees explored in the final iteration. It follows that the order in which these trees are searched can have a critical impact on the running time. We attempt to increase the probability that the tree upon which the search is completed is visited near the beginning of an iteration by sorting the trees in each iteration according to how many leaves are eliminated in by subtree reduction. Our hypothesis is that trees with larger common subtrees are more likely to be near the destination tree. Since at most n leaves can be eliminated by subtree reductions, the trees can be bucket sorted in $O(n)$ time, leaving the asymptotic complexity unchanged. These last two heuristics are employed by replacing the call to ITERATE in SPRDIST to a call to SORT_ITERATE, shown in Algorithm 3.

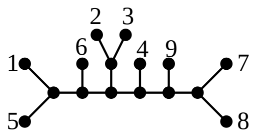
Algorithm 3 SORT_ITERATE($\mathcal{L}_{in}$, $\mathcal{L}_{out}$, H, T)

```
1: for all $t \in \mathcal{L}_{in}$ do
2:   Flag all subtrees in $t$ that also occur in $T$
3: end for
4: Bucket Sort $L_{in}$ in decreasing order by number of vertices flagged
5: for all $t \in \mathcal{L}_{in}$ do
6:   if $t \in H$ then
7:     return TRUE
8:   else
9:     Append set of SPR neighbors which preserve flagged subtrees of $t$ to $\mathcal{L}_{out}$
10:    Insert $t$ into $H$
11:  end if
12: end for
13: return FALSE
```

A cluster is the leaf set of a pendant subtree. T_1 and T_2 share a common cluster C if they contain pendant subtrees S_1 and S_2 respectively such that $L(S_1) = L(S_2) = C$. In (Baroni et al. 2006), it was shown that the hybridization number of two trees is equal to the total of the hybridization numbers of all their pairs of maximal common clusters. In (Beiko & Hamilton 2006), the authors made a similar assumption in their heuristic algorithm to measure LGT. Such a decomposition makes intuitive sense for exact SPR distance as well, as it would seem that any SPR operation that affects more than one common cluster would not reduce the distance and therefore not be part of an optimal solution. Unfortunately, this is not the case as evidenced by the counterexample given in Figure 4 which presents T_1 and T_2 that share the common cluster $\{7, 8, 9\}$. $d_{SPR}(T_1, T_2) = 3$ and 3 SPR operations are shown that transform T_1 into T_2, the first of which breaks the common cluster. Indeed an exhaustive simulation showed that no 3 sequential SPR operations exist to transform the trees that do not break the common clusters. This can be more easily seen by observing that any such sequence would have to regraft 7 to 9 and only 2 operations would be left to transform the cluster $\{1, 2, 3, 4, 5, 6\}$ which is clearly insufficient.



(a) T_1 (b) $\{3, 4\}$ is regrafted to $\{9\}$ (c) $\{2\}$ is regrafted to $\{3\}$



(d)

Figure 4: Example of trees whose common clusters cannot be maintained by a minimal SPR path. T_1 (a) and T_2 (b) have a SPR distance of three but all possible sequences of SPR operations of this length (one is shown by the dotted lines) break the common cluster $\{7, 8, 9\}$.

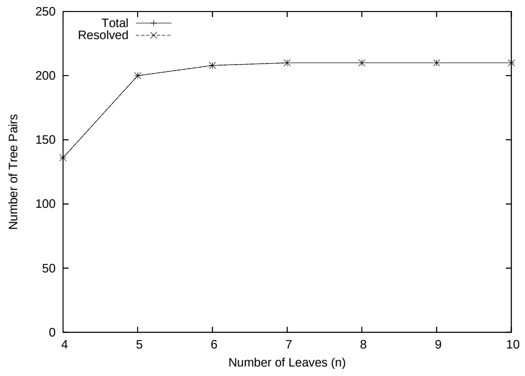
4 Experimental Results

4.1 Datasets

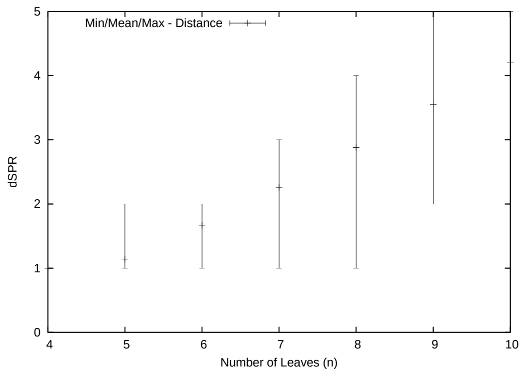
The datasets were chosen to analyze the merits of the heuristics discussed in the previous section as well as evaluate our algorithm in a realistic setting. To these ends, we benchmarked our algorithm on a variety of randomly generated trees, as well as trees created by (Beiko et al. 2005) in the course of analyzing the proteins from the 144 sequenced prokaryotic genomes available at the time. Two sets of random trees were generated, one by the Yule-Harding model and the other by random walks. Yule-Harding trees are constructed by first creating an edge between two randomly selected leaves, then randomly attaching the remaining leaves to the tree until none are left. The random walk dataset consists of pairs of trees such that one of which is generated by the Yule-Harding model and the other is created from the first by applying a sequence of between 2 and 8 random SPR operations (Beiko & Hamilton 2006). The size of the datasets, along with the average distances computed by our algorithm are presented in Figure 5. In some cases, the program ran out of memory before finding the solution. The fraction of instances successfully resolved for each type of data is listed in the "% Resolved" column.

4.2 Performance

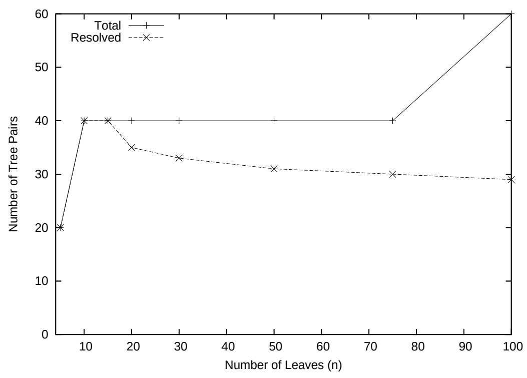
The algorithm described in Section 3 was implemented in C++ and benchmarked on a 2.6Ghz Pentium Xeon System with 3G of RAM. The source code is available at `http://morticia.cs.dal.ca/lab_public/?Download`. This program was executed for all pairs of trees described in Figure 5 with and without the various heuristic optimizations discussed previously. Graphs (a), (c) and (e) in Figure 6 display the effectiveness of the reduction rules' ability to reduce the input trees. As could be expected, the trees in the protein and random SPR walk datasets are reduced more than the two random datasets as their ratios of size to distance are much higher. In all cases, the amount of reduction increases in correlation to the mean distance rather than n. Our method is essentially a fixed parameter tractability (FPT) approach (Downey & Fellows 1998) and our experiments indicate that



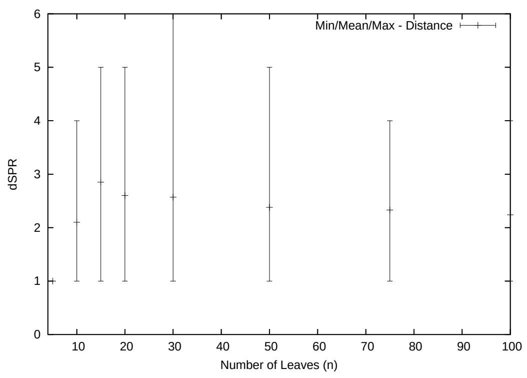
(a) Yule-Harding Random Pct. Resolved



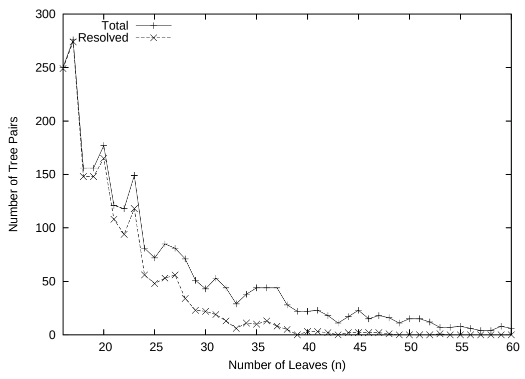
(b) Yule-Harding Random Distances



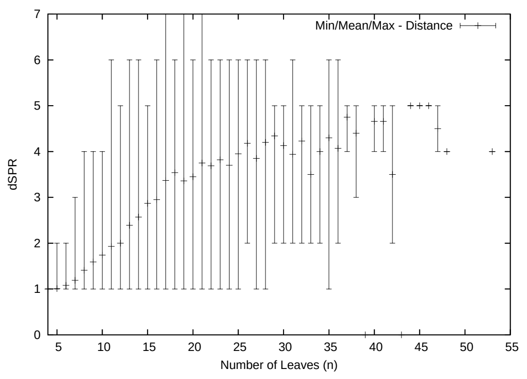
(c) Simulated SPR Walk Pct. Resolved



(d) Simulated SPR Walk Distances



(e) Protein Pct. Resolved



(f) Protein Distances

Figure 5: Size, success rate and distance distributions for each dataset. For the protein data, no trees of size greater than 60 were resolved.

the running time behaves similar to FPT algorithms. Also encouraging is the fact that the reduction rules perform much better in practice than the worst-case analysis in (Allen & Steel 2001), which predicts a reduction in size to a factor of 28 times the distance. For example, in the random SPR walk dataset whose mean distance is roughly 2, the reductions are effective for $n > 4$ whereas in the worst case it is only guaranteed to work for $n >= 56$. Similar results are visible in the protein dataset graphs as well. As can be seen in these graphs, chain reductions accounts for only a small portion (well under 10%) of the overall gain with subtree reductions making up the rest. We also note that of the roughly 20,000 pairs of trees tested, application of the chain reduction rule did not once affect the SPR distance.

The performance of the remaining heuristics is displayed in terms of running time in graphs (b), (d) and (f) in Figure 6. Applying the reductions to intermediate trees provided very little performance gain, implying that the search space is dominated by trees with few common subtrees and chains. However, sorting the trees visited in each iteration of the search by the number of leaves reduced had a significant impact on the running time for all of the harder cases ($d_{SPR} \geq 4$), speeding up the computation by nearly a factor of 6 for some of the larger protein tree pairs.

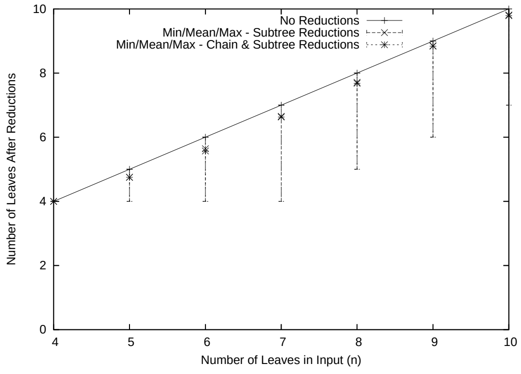
5 Conclusion

The computation of SPR distances between unrooted phylogenetic trees can be used to compare the evolutionary histories of different genes and provide a lower bound on the number of lateral transfers. Little previous work has been done on this problem though many related tree metrics have been relatively well studied in the literature. The reason for this appears to be primarily due to less insight into the problem's structure (no known MAF reduction) rather than lack of interest. In this paper we revisited the problem of unrooted SPR distance, showing that it is NP-Hard and providing an optimized algorithm and implementation to solve it exactly. The algorithm is based on dividing the problem into two searches and making use of heuristics such as subtree reductions and reordering. This algorithm was able to quickly compute the exact distance between the majority of proteins belonging to 144 available sequenced prokaryotic genomes and their supertree. Our method can also be used to improve the brute force search component of TBR and rooted SPR distance computation.

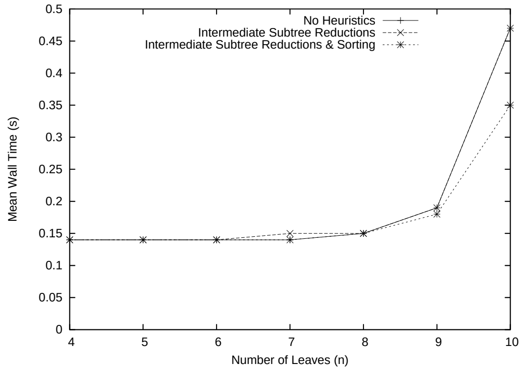
Though a polynomial time solution is unlikely due to its NP-Hardness, some possible avenues of future work on this problem remain. One is to show that chain reductions do not affect the distance, a conjecture that is supported by our experimental results but for which an analytical proof remains absent. This result would be sufficient to show that unrooted SPR distance is fixed parameter tractable, being exponential only in terms of the distance and not the size of the trees. In (Bordewich et al. 2007), a decomposition by common clusters was used with significant practical success. We showed that such a technique cannot be directly applied to the problem of unrooted SPR distances but perhaps a variation of this technique can.

Acknowledgment

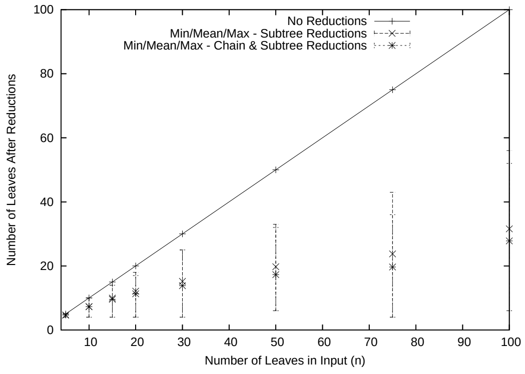
This research partially supported by the Natural Sciences and Engineering Research Council of Canada and Genome Atlantic.



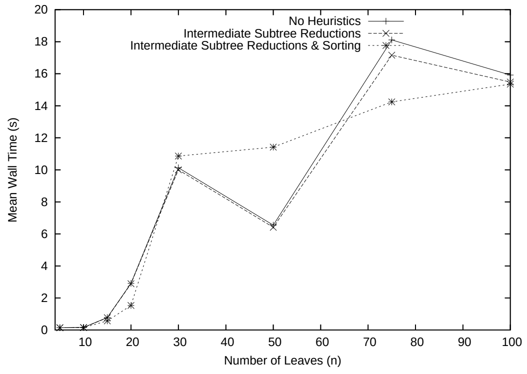
(a) Yule-Harding Random



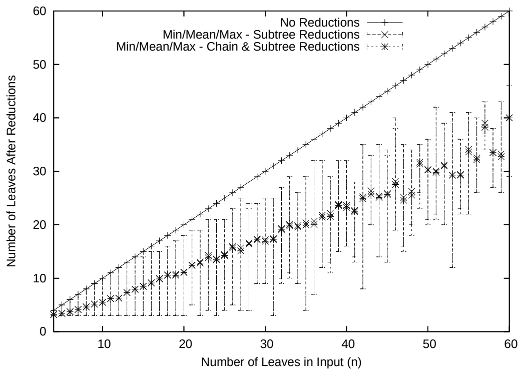
(b) Yule-Harding Random



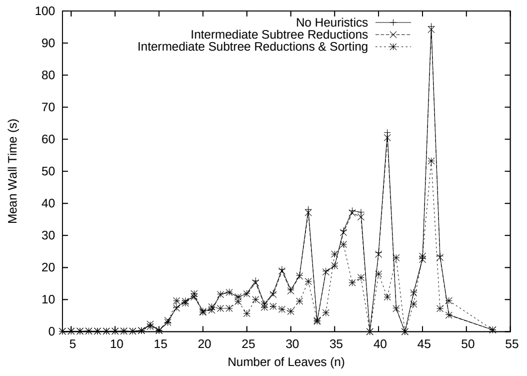
(c) Simulated SPR Walk



(d) Simulated SPR Walk



(e) Protein



(f) Protein

Figure 6: Experimental evaluation of the different heuristics on the three datasets. The effect of the reduction rules on the input tree sizes is displayed on the left. The improvements to the running time made by reducing and sorting intermediate trees are displayed on the right.

References

Allen, B. L. & Steel, M. 2001. Subtree transfer operations and their induced metrics on evolutionary trees, *Annals of Combinatorics* 5(1):1 – 15.

Baroni, M., Semple, C. & Steel, M. 2006. Hybrids in real time, *Systematic Biology* 55(1):46–56.

Beiko, R. G. & Hamilton, N. 2006. Phylogenetic identification of lateral genetic transfer events, *BMC Evolutionary Biology* 15(6).

Beiko, R. G., Harlow, T. J. & Ragan, M. A. 2005. Phylogenetic identification of lateral genetic transfer events, *Proc. Natl. Acad. Sci. USA* 102:14332–14337.

Bordewich, M., Linz, S., John, K. S. & Semple, C. 2007. A reduction algorithm for computing the hybridization number of two trees, *Evolutionary Bioinformatics* 3:86–98.

Bordewich, M. & Semple, C. 2004. On the compuational complexity of the rooted subtree prune and regraft distance, *Annals of Combinatorics* 8(4):409 – 423.

Doolittle, W. F. 1999. Phylogenetic classification and the universal tree, *Science* 284:2124–2128.

Downey, R. & Fellows, M. 1998. *Parameterized Complexity*, Springer-Verlag.

Garey, M. R. & Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company.

Hein, J., Jiang, T., Wang, L. & Zhang, K. 1996. On the complexity of comparing evolutionary trees, *Discrete Applied Mathematics* 71:153–169.

Appendix

A X3C Remains NP-Complete when Each Element Occurs in Exactly 3 Subsets

In this appendix we verify that X3C remains NP-Complete in the special case where each element occurs in *exactly* three subsets. Consider an instance of X3C in which no element occurs in more than three subsets. We provide a polynomial time reduction from such an instance, known to be NP-Complete, into an instance in which each element occurs in exactly three subsets. Let:

$Y_1 \subseteq X$: Elements of X that appear in exactly one subset
$Y_2 \subseteq X$: Elements of X that appear in exactly two subsets
$Y_3 \subseteq X$: Elements of X that appear in exactly three subsets

So $|Y_1| + 2|Y_2| + 3|Y_3| = |X| = 3q$

For each element to appear in exactly three subsets, we must add $2|Y_1| + |Y_2|$ elements to subsets in C.

Let multiset $Z = \{z_0, z_1, \ldots, z_{3p-1}\} = Y_1 + Y_1 + Y_2$ be these elements we have to add. Note that $|Z| = 3p$ where $p = 2(q - |Y_3|) - |Y_2|$.

Let $X' = \{x'_0, x'_1, \ldots, x'_{3p-1}\}$ be a set of new elements such that $|X'| = 3p$ and $X \cap X' = \emptyset$.

We now create a collection C' of new subsets out of Z and X' so that each element in $X \cup X'$ appears in a subset in $C + C'$ exactly three times.

For each $i = 0, 3, \ldots, 3p-1$, we add four subsets to C':
$c'_{4i} = \{x'_i,\ x'_{i+1},\ x'_{x+2}\}$
$c'_{4i+1} = \{z_i,\ x'_i,\ x'_{i+1}\}$
$c'_{4i+2} = \{z_{i+1},\ x'_{i+1},\ x'_{i+2}\}$
$c'_{4i+3} = \{z_{i+2},\ x'_{i+2},\ x'_i\}$

We now show that $X \cup X'$ and $C + C'$ form an instance of X3C such that every element of $X \cup X'$ appears in 3 subsets in $C + C'$ and X has a cover in C if and only if $X \cup X'$ has a cover in $C + C'$.

(if): If X has a cover in C, then $X \cup X'$ has a cover in $C + C'$: Let $S \subseteq C$ be the cover of X. Then $S + c'_0 + c'_4 + c'_8 + \ldots + c'_{12p-1}$ is a cover $X \cup X'$.

(only if): If $X \cup X'$ has a cover in $C + C'$, then X has a cover in C: Similar to above, the only way to cover X' is with $c'_0 + c'_4 + c'_8 + \ldots + c'_{12p-1}$ and no other elements of C' can be part of an exact cover. This means that X is covered entirely by subsets in C so X is exactly covered by C.