Approximating the True Evolutionary Distance Between Two Genomes

Krister M. Swenson* Mark Marron† Joel V. Earnest-DeYoung†
Bernard M.E. Moret* † ‡

Abstract

As more and more genomes are sequenced, evolutionary biologists are becoming increasingly interested in evolution at the level of whole genomes, in scenarios in which the genome evolves through insertions, duplications, deletions, and movements of genes along its chromosomes. In the mathematical model pioneered by Sankoff and others, a unichromosomal genome is represented by a signed permutation of a multiset of genes; Hannenhalli and Pevzner showed that the edit distance between two signed permutations of the same set can be computed in polynomial time when all operations are inversions. El-Mabrouk extended that result to allow deletions and a limited form of insertions (which forbids duplications); in turn we extended it to compute a nearly optimal edit sequence between an arbitrary genome and the identity permutation. In this paper we generalize our approach to compute distances between two arbitrary genomes, but focus on approximating the true evolutionary distance rather than the edit distance. We present experimental results showing that our algorithm produces excellent estimates of the true evolutionary distance up to a (high) threshold of saturation; indeed, the distances thus produced are good enough to enable the simple neighbor-joining procedure to reconstruct our test trees with high accuracy.

1 Introduction

Gene-content and gene-order data are becoming more common and are increasingly used in the study of evolution (see [14]) and in comparative genomics (see [1]). We can compare genomes from various species under the assumption that certain biologically plausible operations have, through time, shaped their current conformation from a single common original genome. Changes to genomic content or to gene order are of particular interest, as they arise infrequently and so offer the potential for reconstructing very old evolutionary events as well as computing pairwise evolutionary distances between distantly related modern genomes (see [6, 16, 17, 19]).

Biologists can observe the ordering and strandedness of genes on each chromosome, thereby producing a *gene order* for each chromosome, a sequence of signed integers in which each integer represents a gene (the same gene may appear multiple times in the genome) and the sign indicates the strandedness. In turn, evolutionary events can be couched in terms of operations on such signed orders: inversions, insertions, duplications, and deletions all have simple representations in this model. The model then leads naturally to the problem of defining the *distance* between two genomes in terms of these operations. The distance one would want is simply the actual number of evolutionary events (from the list of allowed operations) that took place to evolve one genome into the other—what is known as the *true evolutionary distance*. Not only is that distance of biological interest, but knowledge of the pairwise true evolutionary distances is sufficient to reconstruct the true phylogeny. Since that value cannot be computed exactly, however, computational biologists have instead developed algorithms to compute the *edit distance*, i.e., the smallest number of evolutionary events needed to transform one genome into the other. An edit distance has the advantage of presenting a clearly defined minimization problem but it also underestimates the true evolutionary distance. Thus com-

*School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne, CH 1015, Lausanne, Switzerland, {krister.swenson,bernard.moret}@epfl.ch

†Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, USA

‡Swiss Institute of Bioinformatics

putational biologists have developed methods for *correcting* the edit distance (according to empirical data) in order to produce an estimate of the true evolutionary distance. Such correction methods introduce problems of their own, however. In particular, the variance in the estimator grows as the distance grows, to the point where, beyond a certain threshold known as the *saturation* value, the estimate is too noisy to be useful.

However, even computing an edit distance is a very complex problem for whole genomes. Simply finding the edit distance between two *unsigned permutations* with only inversions allowed (no change in content and no duplicate genes) is NP-hard [3]; the same problem with signed permutations is solvable in polynomial time thanks to the results of Hannenhalli and Pevzner [9]—in fact, the edit distance can then be computed in linear time [2]. On the other hand, finding an ancestral permutation that minimizes the sum of distances over a tree of just three taxa under this model (the so-called median problem) is NP-hard even for signed permutations [4]. Computing the edit distance for genomes with unequal content is barely touched but known to be NP-hard [5], even when one genome is the identity [18]; El-Mabrouk [8] showed how to extend the theory of Hannenhalli and Pevzner to include deletions, but her results assume no duplicate genes, obviously a major limitation in practice.

In previous work [12], we gave a polynomial-time approximation algorithm for the (NP-hard) computation of the minimum edit distance from any genome to the identity permutation $1, 2, \ldots, n$. Our method relied on the construction of a mapping between duplicates in the genomes, yielding a partial map we called a *cover*. This cover associates substrings of genes that exist in both subject and target genomes (modulo an inversion); any genes not included in the cover are then treated as deletions from the subject or insertions into the target. A minimal cover is one that uses the fewest substrings. While Section 2.2 formally defines the cover, Figure 1 illustrates the concept: the target is the identity permutation $1, 2, 3, 4, 5, 6$, while the subject, -2,-1, $2, 3$,-6,-5, $3, 4$, includes duplicates of genes "2" and "3". The cover consists of three substrings, namely "$1, 2$", "$3, 4$", and "$5, 6$"; concatenated, these substrings produce the target string, while separately they can be found in the subject string as "-2,-1", "$3, 4$", and "-6,-5". The third and fourth genes in the subject string (forming the substring "$2, 3$") are then viewed as lost in the evolution to the target.

subject $\boxed{\text{-2-1}}$ 23 $\boxed{\text{-6-5}}$ $\boxed{34}$

target $\boxed{12}$ $\boxed{34}$ $\boxed{56}$

Figure 1: A minimal cover.

Here we generalize this approach to compute the distance between two arbitrary genomes and show through extensive simulations that we reconstruct a sequence of operations that reflects the true evolutionary distance. Our algorithm computes distances between two genomes in the presence of insertions (including duplications), deletions, and inversions; in our simulations, the distance computed very closely approximates the true evolutionary distance up to a (high) saturation level. The approximation is in fact good enough that it can be used in conjunction with a distance-based phylogenetic reconstruction method (we used the most common one, neighbor-joining) to reconstruct trees of reasonable sizes (up to 100 genomes) and very large pairwise distances with high accuracy.

It is worthwhile to note that although we consider only inversions (aside from duplicating insertions and deletions), the properties of a minimum cover discussed in Section 2.2 imply that it would likely perform well with other operations such as transpositions: the cover is a model-independent method. However, due to the fact that transpositions distances are yet to be well understood we do not consider them in this exposition.

The rest of the paper is organized as follows. Section 2 reviews the problem and past results and establishes notation. Section 3 discusses the difficulties faced when using two arbitrary genomes and how we solve them to recover a solution in the spirit of our earlier results; it outlines our method for producing a cover in quadratic time. Finally, Section 4 presents the design of our two studies while Section 5 shows how our constructed cover performs when estimating pairwise tree distances and how these distances can be used in tree reconstruction.

2 Background

2.1 The problem We consider the problem of approximating the true evolutionary distance (as determined through simulations) from an arbitrary *subject* genome to an arbitrary *target* genome. The operations that we consider are *inversions*, in which the order of a substring of genes is reversed and the sign of each gene in the substring flipped; *deletions*, in which a substring of genes is removed; and *insertions* (including *duplications*), in which substrings of genes (including entirely new genes not found anywhere else) are added. All duplicates of one gene form a *gene family*; and all genes bearing the same identifier are known as *homologs*—that is, they are considered to have been derived from a common ancestral gene through various cascades of evolutionary events (that include both duplications and nucleotide-level changes).

2.2 The cover Our solution attempts to assign each duplicate gene in the subject to a particular homolog in the target; that is, it creates a maximum matching between the genes in corresponding gene families of the two genomes. However, some matchings are clearly preferable to others because they reduce the number of insertions, deletions, and rearrangement operations required to transform one genome into the other. We say that a cover is *optimal* if the correspondence it establishes leads to a minimum number of operations (inversions, insertions, and deletions) in the shortest sequence required to transform the subject into the target while respecting the map. However, computing such a cover is NP-hard [18], so we define a *minimum cover* to be a cover that maps the subject to the target with the fewest common substrings. The effect of renaming according to a minimal cover is to yield a breakpoint graph [9] with maximum number of cycles of length 2, minimizing the number of breakpoints between the renamed genomes.

2.3 Difficulties with an arbitrary target The main difference between our previous work [12] and our new algorithm is the presence of duplicate genes in the target. When building the cover with the identity permutation as the target [12], all candidate cover elements from the subject are immediately apparent because of the unique correlation between their identity and their index in the target genome. In the case of an arbitrary target, however, this correlation no longer exists. Moreover, a cover may no longer cover all genes from one or the other genome: clearly, if genome A has more duplicates of gene x than genome B, and genome B has more duplicates of gene y than genome A, then any matching between these two genomes must leave some duplicates of gene x unassigned in A and some duplicates of gene y unassigned in B. For example, with subject $1, 2, 3$,-5,-2 and the identity permutation $1, 2, 3, 4, 5$ as target, we have a cover, using indices in the target, for indices 1 through 3, one for index 5, and one for index 2; but for the same subject and for target -7, $1, 2, 3, 5$,-3, we obtain partial covers for indices 2 through 4 or for indices 5 through 6.

3 Constructing a (Nearly) Minimum Cover

The algorithm used in [12] looks for the longest matching substring. As long as such a longest match is unique, there is no difficulty beyond identifying such matches as quickly as possible. (A naïve cubic-time algorithm will do, although, as we shall see, the same job can be done in quadratic time.) When the longest match is not unique, however, finding a minimum cover may require an exploration of the alternatives and thus exponential time. Instead, we use a greedy heuristic to break ties.

We have tried several tie-breaking heuristics (and compared them to breaking ties at random). One heuristic is based on identifying a possible extension of the match (to one or the other side). If the substring to one side of the match is the inverse of the substring to the same side of the match in the other genome, for instance, if we had substrings "1, 2,-4,-2" in the target and "$1, 2, 2, 4$" in the subject, we may prefer to match these substrings to each other (even though there may be another "$1, 2$" elsewhere in both genomes) because they are only a single inversion from each other.

Another heuristic is to minimize the interaction between matches. The longer the match we make at each iteration, the fewer potential matches may be needed overall, so we may want to choose the match with a range of indices that crosses the smallest number of other match ranges. Section 6 contains some conclusions about the effectiveness of these heuristics.

To find the longest match, we begin by finding all possible maximal matching substrings and then repeatedly pick the next largest substring, doing necessary bookkeeping to reflect our successive choices. Let M be the set of all maximal matching substrings between the subject and the target that have not yet been picked. For instance, if we start with target genome $1, 2, 1, 3, 4, 5, 6, 7, 8$ and subject genome $6, 7, 3, 4, 5, 6, 1, 2, 3, 6, 7, 8$, we initially have $M = \{$"$6, 7$", "$3, 4, 5, 6$", "$1, 2$", "$6, 7, 8$"$\}$. We say that two matches *overlap* if their indices in the target intersect. By picking the longest match l, we cover a part of the target that may overlap with some number s of other matches, call them $o_1, o_2, \ldots, o_s \in M$. In our example, match "$3, 4, 5, 6$" would be chosen first, covering the 6 from matches "$6, 7$" and "$6, 7, 8$" and the 3 from match "$1, 2, 3$". The overlapping portion of each match $o_i, 1 \leq i \leq s$ is then removed, resulting in shorter matches. Thus, two of those matches in our example will be shortened by 1 yielding "7", "$7, 8$", and "$1, 2$". The resulting algorithm is described in Figure 2.

We proceed to show that COVER can be implemented to run efficiently, first stating the theorem and then providing the necessary background to prove it.

THEOREM 3.1. *Algorithm* COVER *can be implemented to run in quadratic time.*

We represent M by a list arranged by match length. We keep an auxiliary data structure, the *index reference*, to maintain the set M through each iteration. This index reference is an array (0 indexed) of lists, one for each index of the target; each such list, an *index list*, contains the matches that have an endpoint on that target index. For instance, in our example three such matches would be "$3, 4, 5, 6$", "$6, 7$", and "$6, 7, 8$". These matches are associated with indices 3 through 6, 6 through 7, and 6 through 8 of the target. Thus index 6 of the target would have three members to its index list, because the matches "$3, 4, 5, 6$", "$6, 7$", and "$6, 7, 8$" all have "6" as an endpoint. Index 7, however, would have a single match "$6, 7$", because "$6, 7, 8$" does not have "7" as an endpoint. A simple way to find all possible maximal matches in quadratic time is to slide the subject over the target, comparing all possible combinations of indices between the two. Each match found is placed in M and the index lists for its endpoints. The key to this implementation is the efficient update of overlapping matches. With the index lists we can find all $o \in M$ that overlap a given $m \in M$ by examining each list that corresponds to an index that m spans. When the match m that spans indices i through k is chosen, we can shorten each o_i that overlaps from the left by relocating it from the index list for $j, i \geq j \geq k$, to the index list for $i - 1$. Similarly, each o_k that overlaps m from the right can be relocated to the index list for $k + 1$.

LEMMA 3.1. *The maximum number of matches that can have an endpoint at a given index of the target is bounded by $4n$, where n is the length of the longer genome.*

Proof. Each index in subject or target can be of two types: a left or right endpoint of a match. All four combinations of endpoint types can occur for a given pair of indices. If there were more than one match per pairing of endpoint types then one of them could not be maximal, therefore there can be at most four distinct maximal matches associated with every pair of indices. Since there are n indices in the subject, there can be at most $4n$ matches associated with a single index of the target.

It follows immediately that the number of maximal matches between two genomes, the larger of which has size n, is $O(n^2)$.

LEMMA 3.2. *Initialization of M and of the index reference takes quadratic time.*

Proof. We know that the number of maximal matches is $O(n^2)$ and that the length of a match is

ALGORITHM COVER:

$C = \emptyset$.
$M = \{$s : s is a maximal substring of the subject and target$\}$.
WHILE C cannot cover the Target DO:
    Add longest $l \in M$ to C.
    $M = M \backslash \{l\}$.
    FOREACH $o \in M$ that overlaps l DO:
        $u = o$ without the substring common to o and l.
        $M = M \backslash \{o\} \cup \{u\}$.
RETURN C

Figure 2: Choosing a nearly minimal cover.

bounded by the size of the genomes. We can add a match to a list organized by length in constant time through direct indexing. Likewise, addition to the end of a given index list can be done in constant time. Since there are $O(n^2)$ matches and placement into the index reference is $O(1)$, we can build these lists in quadratic time.

LEMMA 3.3. *A match can be relocated between index lists at most twice before being removed from consideration.*

Proof. It is sufficient to show that a match e will not be encroached upon from the same side twice. Assume that e is shortened from one direction by match m and later from the same direction by match m' without being covered. Because m was picked by the algorithm first, m' must not stretch past the opposite end of m. Therefore, either m' covers less than e or e must now be removed from consideration—a contradiction in either case.

We are finally ready to prove Theorem 3.1.

Proof. (of Theorem 3.1) Initialization takes quadratic time (Lemma 3.2). Each match in each index list is visited a constant number of times (Lemma 3.3). When visited, each match is shortened, removed from consideration or relocated to the index list at the edge of the most recently chosen match, and then relocated in the length list. Since each of these operations runs in constant time, the running time is bounded by a constant times the total number of matches visited. Since each index list is visited at most once and the length of that list is at most linear (Lemmata 3.1 and 3.3), the running time is $O(n^2)$.

THEOREM 3.2. *The distance function can be computed in $O(n^2)$ time.*

Proof. The cover can be generated and applied in $O(n^2)$ time. Then the algorithm presented in [12] or [8] can be applied. Both methods run in $O(n^2)$ time.

4 Experimental Design

We used two types of tests to assess the accuracy and utility of our tree distance algorithm. The first set of tests were designed to determine if our distance function accurately modeled the true pairwise tree (true evolutionary) distances. The second set of tests were used to evaluate the effectiveness of our distance function within the most simple distance-based phylogenetic reconstruction algorithm.

4.1 Pairwise Error For this experiment, we generated evolutionary trees with known edge lengths and compared the pairwise distances between the leaves with those computed by our algorithm. Variance in tree shape does not matter here;

in fact, since we want a large range of pairwise tree distances, a perfectly balanced tree is best.

In the following tests we used the simplest version of the method described earlier. The algorithm picks the largest match to make and in the case of ties picks one of the tied matches at random. Clearly other information is present in the genomes that could provide a better choice of match and thus lead to a more accurate distance score. However, all of the heuristic methods that we used failed to have a noticeable impact on the accuracy of the distance value returned. Furthermore, in experiments with a large number of random restarts, we found that most of the values clustered around the true value with a small number of outliers; we also found that averaging over a smaller number of random restarts and discarding any substantially outlying points provided a distance estimate that was nearly indistinguishable from the distance estimate computed with the use of our best heuristics (see Section 6). While the use of biological information to select the best match could prove effective in generating more biologically plausible evolutionary paths, the current method seems to perform quite well in terms of distance computations.

Not enough is known about inversions, deletions, insertions, and duplications to enable one to set good parameters (such as lengths of inversion, for instance) *a priori*, so we chose values so as to ensure that a single operation would not completely alter the genome. Most of our tests were conducted with a root genome of 800 genes on a tree of depth 4; such a tree has 16 leaves and thus 120 pairs of genomes with paths from 2 to a maximum of 8 edges between genomes.

4.2 Tree Reconstruction We tested the performance of our distance functions using neighbor-joining, the standard distance-based tree reconstruction method. Due to the dearth of real-world trees reconstructed using biological techniques, we had to generate model trees that would exercise our algorithm over a wide range of plausible models of gene-order evolution. (We conducted one study using real data with very large numbers of insertions and deletions; partial results to date show promise [7].) We generated one thousand trees using a variation of the birth-death model that produces a larger variation in tree topologies, especially imbalanced ones that are known to be insufficiently represented in a pure birth-death model [10]. The only constraint that was placed on the operations was that the expected number of inserted elements was equal to the expected number of deleted elements, in order to keep all genome sizes within a reasonable range. (Cases where certain genomes are much smaller than others, due, e.g., to symbiosis, certainly exist, but the variation generated by our mechanism nearly encompasses that case already.) Three random restarts of our distance algorithm were used for each pair of nodes to produce the pairwise distance matrix.

Within the thousand trees the percentage of inversions varied from 50% to 90%. The remaining percentages were split evenly between insertions (duplicating and non-duplicating) and deletions. Non-duplicating insertion and duplication percentages were varied over three different tests, in which each received a quarter, a half, and three quarters of the percentage. The expected Gaussian distributed length of each operation filled a range of combinations from 5 to 30 operations per operation type. Finally, the expected number of event per edge was 20 with a Gaussian distributed variance of 10 operations.

To generate a tree we began with the identity genome on 800 genes and performed 200 evolutionary operations on it using the same parameters that are specified for generating the tree. This genome was then used as the root of the tree. For each node we checked if it should become a leaf, based on the maximum depth allowed and a random choice, if not we stopped. Otherwise we created each of the two children by performing the randomly selected operations (as specified in the previous paragraph) on the parent. Each type of operation (inversion, non-duplicating insertion, duplication, and deletion) was selected at random according to a fixed distribution. The interval over which an operation acts is produced with one endpoint selected at random and a length drawn from a Gaussian distribution. For duplications, the interval to be duplicated is selected and then inserted at an index chosen uniformly at random in the genome.

5 Experimental Results

5.1 Pairwise Error We present results for one of the many mixes of operations used in our simulations; other mixes gave very similar results. This particular data set used a mix of 70% inversions, 16% deletions, 7% insertions, and 7% duplications. The inversions had a mean length of 20 and a standard deviation of 10. The deletions, insertions, and duplications all had a mean length of 10 with a standard deviation of 5. We used four trees of 16 leaves as described earlier, with 10, 20, 40, and 60 expected operations per tree edge; these choices can result in very large pairwise distances—up to an expected 480 operations (on just 800 genes) for the most distant pairs. For these four trees, our algorithm was run with 10 random restarts and simple randomization for the selection of the matchings.

Figures 3 through 6 show the results (as a scatter plot of the 120 data points for each experiment) for these four datasets. In each figure, the left-hand plot shows the estimated tree distance on the ordinate against the true evolutionary distance (from the simulation) on the abscissa. A perfect result would simply trace the 1:1 diagonal, which is lightly marked on each plot to aid in evaluating the results. The right-hand plot displays the deviation from the 1:1 ideal as a function of the true evolutionary distance, plotting largest and smallest differences between computed values and the true value, for each true value.

These plots show that our distance estimator tracks the true evolutionary distance very closely up to a saturation threshold, where it starts lagging seriously behind the true value. Such saturation is of course expected; what is surprising is how high that saturation threshold is. On genomes of roughly 800 genes, saturation appears to occur only around 250 evolutionary events and our estimator tracks very accurately to at least 200 events. Moreover, the smaller plots indicate that the variance is very small up to 200 events and remains reasonable up to 250 events.

These results are not limited to small trees. We ran another series of tests involving trees of 50 leaves; while the main purpose of these tests was to assess the quality of tree reconstruction using our distance computations, we checked the computed distances against the true distances for these trees as well. Figure 7 shows the same two scatter plots (this time on roughly 1,250 data points) for one such tree. For these larger trees, we used a root genome of 1,200 genes in order to prevent early saturation; the example reported in the figure used an expected edge length of 20 evolutionary events. With the larger number of genes, saturation now does not occur until we reach at least 350 evolutionary events. The error plot shows that the error remains sharply bounded throughout the range of values tested.

5.2 Tree Reconstruction Since our distance computation tracks tree distances so accurately and since distance-based methods are guaranteed to do well when given distances that are close to the true evolutionary distances, we also ran a series of tests designed to ascertain the quality of tree reconstruction obtained with the most commonly used distance-based reconstruction method, neighbor-joining (NJ). The NJ method runs in low cubic time and thus is applicable to large datasets, but, like all distance-based methods, it is known to produce poor results when the range of tree distances gets large (see, e.g., [15].

Recall that we generated a very large number of diverse tree topologies, producing a population of trees that more closely matches the observed balance statistics [10] than would be the case with a pure birth-death process. We evaluated results using the standard *Robinson-Foulds (RF) distance* [20], which is simply (in the case of binary trees, as in our series of experiments) the number of edges (or bipartitions) present in one tree, but not in the other. In several cases, we present the *RF error rate*, which is the ratio of the RF distance to the number of taxa in the tree. In terms of the latter measure, most systematists will consider rates above 10% to be unacceptable and rates below 5% to be very good.

The tree reconstruction performed very well on the generated trees, as shown in Figure 8. Approximately 65% of the reconstructed trees had a Robinson-Foulds error rate of less than 5% and only 15% of the trees had an error above 10%.

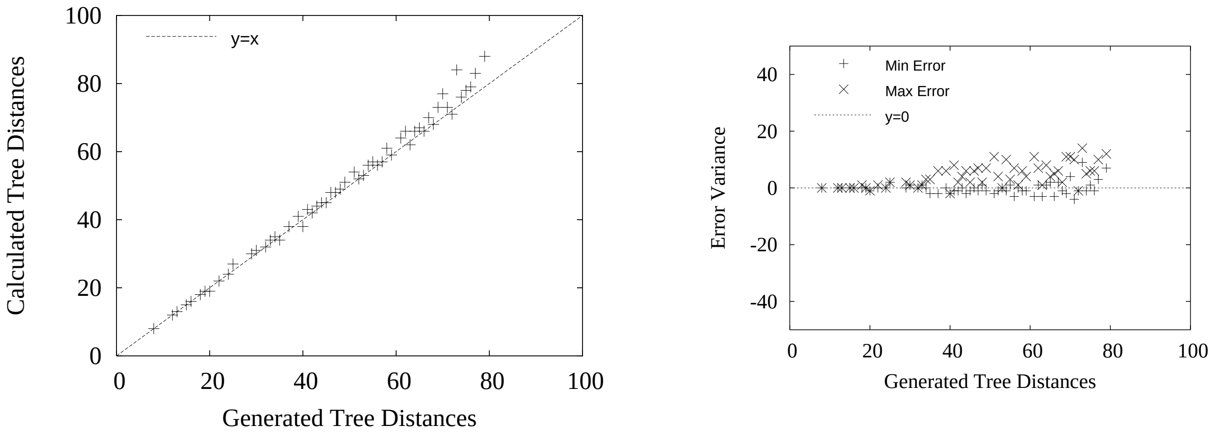


Figure 3: Experimental results for 800 genes with expected edge length 10. Left: generated distance vs. reconstructed distance; right: the variance of computed distances per generated distance.

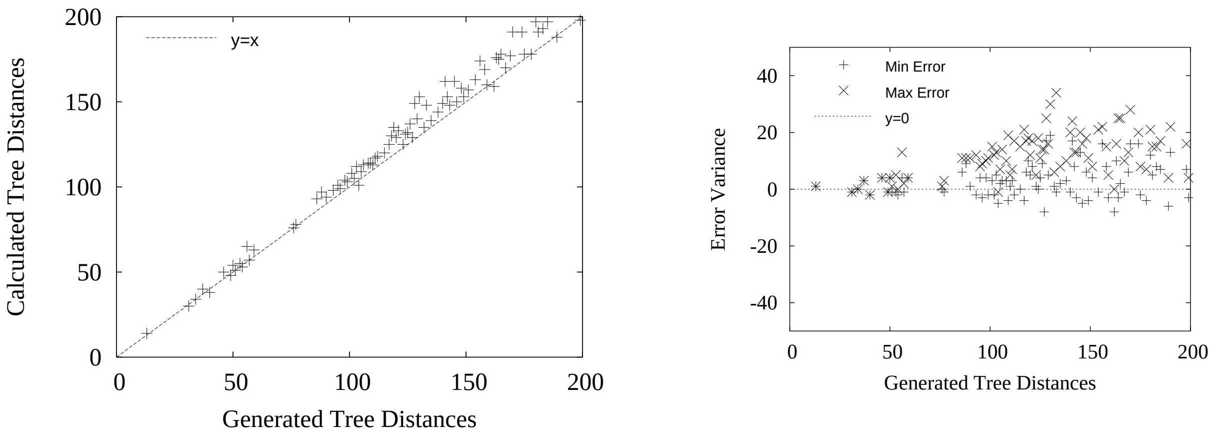


Figure 4: Experimental results for 800 genes with expected edge length 20. Left: generated distance vs. reconstructed distance; right: the variance of computed distances per generated distance.

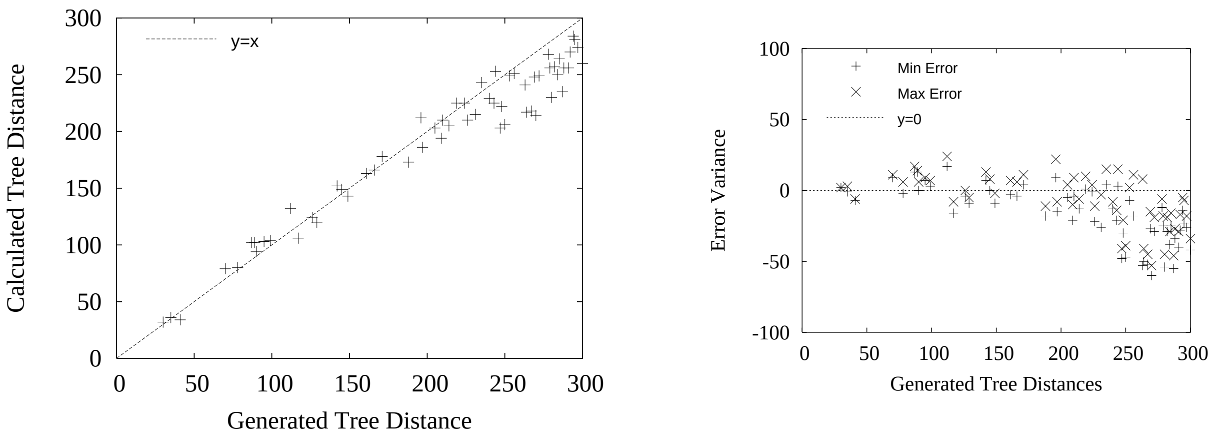
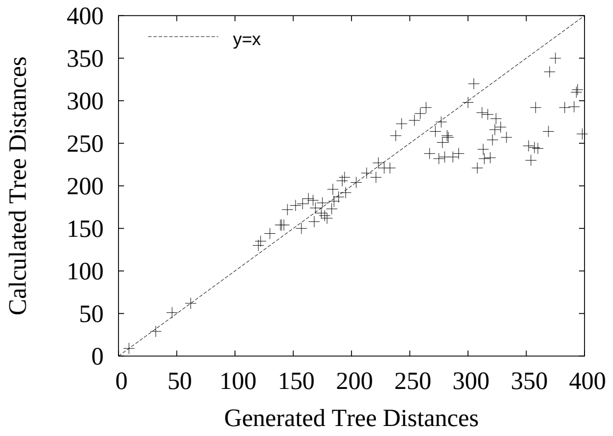


Figure 5: Experimental results for 800 genes with expected edge length 40. Left: generated distance vs. reconstructed distance; right: the variance of computed distances per generated distance.



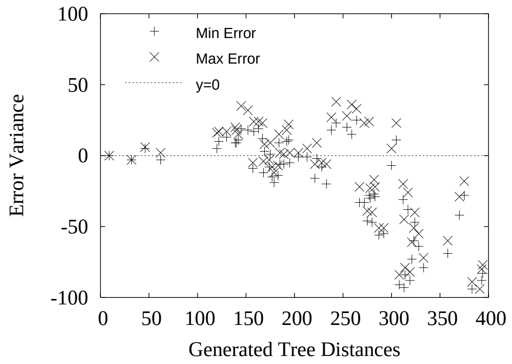
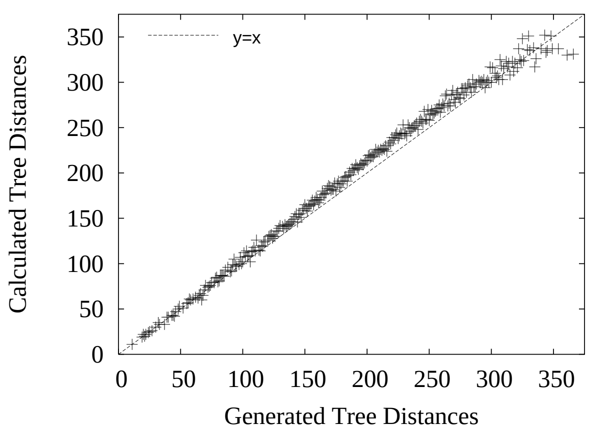


Figure 6: Experimental results for 800 genes with expected edge length 60. Left: generated distance vs. reconstructed distance; right: the variance of computed distances per generated distance.



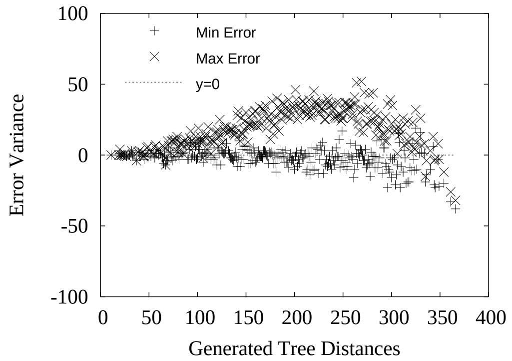


Figure 7: Experimental results for 1,200 genes with expected edge length 20. Left: generated distance vs. reconstructed distance; right: the variance of computed distances per generated distance.

This reconstruction was done without any use of error correction, variances, or knowledge of the underlying model that generated the trees; it also used the simplest form of neighbor-joining. Thus, it would be easy to improve these results by refining the reconstruction method.

As an additional check, we also compared how well our method performs with respect to simply removing duplicate content and applying El-Mabrouk's exact method [8]. This comparison gives us an indication of how important it is to handle duplication in estimating true tree distances. We computed a distance matrix for each tree where a single entry of a matrix was obtained by pairwise removal of all duplicate content and subsequent computation using El-Mabrouk's exact method. The NJ method was applied to each matrix to obtain a tree. Over all thousand trees the reconstruction without duplicates had a lower RF error rate than ours on only 14% of the trees; furthermore, in three quarters of those cases, the overall RF error rate for both methods was lower than 10%—that is, these were relatively easy cases. Thus, our method does better on the harder cases; the average difference in RF error rate on the trees where our method did worse on was 1.2, while the average difference in RF error rate on the trees our method did better on was 3.5. This is strong evidence that our method makes significant improvements on the state of the art. Furthermore, because of this low error rate in the 14% of cases where our method was not the best, there is good reason to believe that a slightly better tie breaker (see Section 6) will yield even more cases where the method presented here wins.

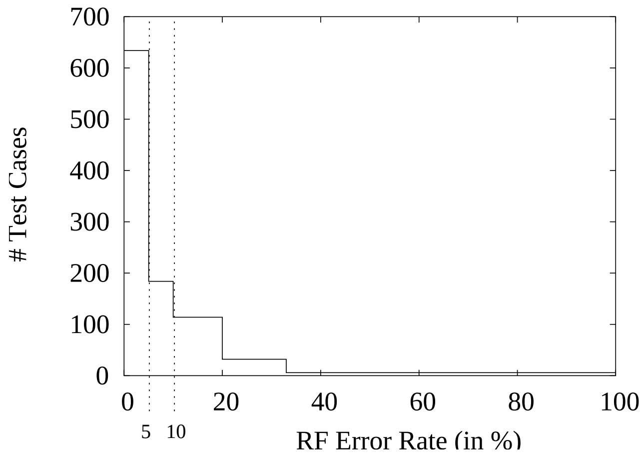


Figure 8: The histogram of RF error rates for reconstructions based on our distance computation.

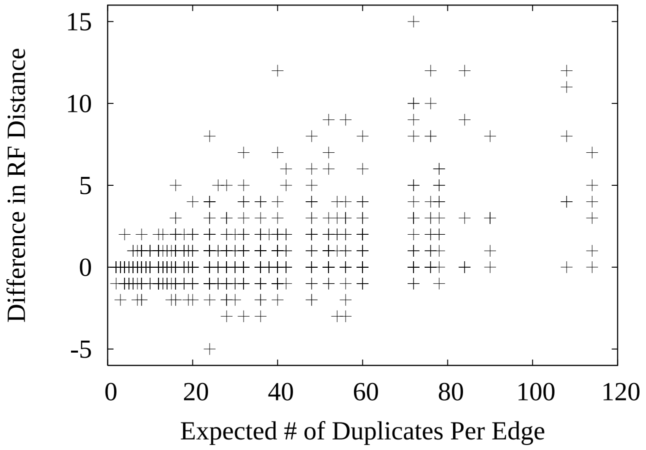


Figure 9: The difference in RF distance between our method and the method without duplicates as a function of the number of duplicates on an edge.

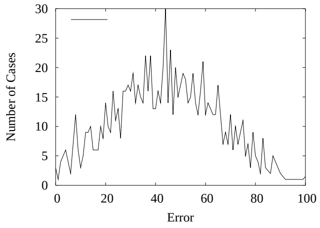
To examine how well our technique handled copies, we compared (for every test run) the RF distances of our reconstruction with those of the reconstruction without duplicates as a function of the total number of duplications. Figure 9, a scatter plot of the differences in RF distance, indicates that, as the number of duplicates increases, our method does correspondingly better at reconstructing the tree.

6 Improved Heuristics

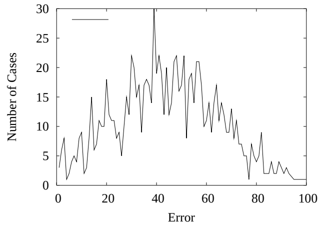
For distances used in tree reconstruction, the relative ordering of the values is more important than their absolute magnitude; it is most important to see computed distances increase as the simulated distances do. Our major goal with the introduction of more sophisticated heuristics is to reduce the variance of the scores so that the distance ordering will be more consistent and potentially result in more accurate trees.

The results presented earlier in the paper used a very simple heuristic; we selected the longest match for a cover element and then chose a match at random in the cases of ties. We investigate two more promising tie-breaking heuristics (introduced in Section 3): picking a match that has the smallest overlap with the other cover elements or picking a match by looking at the immediate context of the cover elements in the source genome. By choosing the match that has minimal overlap with all other matches, we maximize the number of longest-match candidates for the next round. To understand the motivation for the context driven heuristic suppose we are trying to find a cover element for a subsequence (of genes) s in the target. Also suppose that in the target, the subsequence to the left of s is s_l and to the right of s is s_r. Then we would like to pick a match in the source genome that has the context subsequences s'_l and s'_r that are as similar to s_l and s_r as possible.

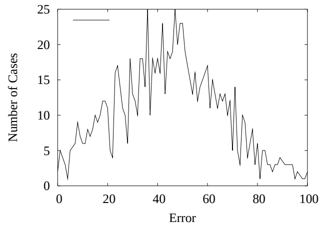
To assess the improvements when using these heuristics we ran two sets of pairwise distance comparisons. One set used genomes of length 800 with 200 operations from the identity to the first taxa and 200 operations between the taxa. The second set used genomes of length 1200 and took 400 operations between the identity genome and the first and between the first and the second taxa. In both data sets the probability of an operation being an inversion was 80%, of being a deletion 10%, of being a duplicating insertion 5%, and of being a non-duplicating insertion 5%. The distance between each pair was then computed using three heuristics, first the random selection was run, then the score was computed using overlap minimization, and finally the score was computed using the overlap minimization with context. Figure 10 indicates that there is little difference in the error values for the various methods. More importantly, the more sophisticated heuristics have very little impact on the variance. All methods resulted in a sample variance of about 22.6 for the genomes constructed with 400 operations.



(a) random tie-breaker



(b) overlaps tie-breaker



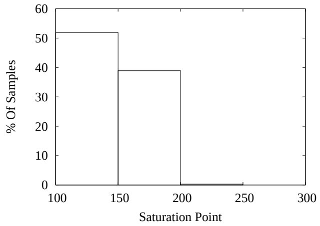
(c) overlaps with context tie-breaker (reduced scale)

Figure 10: Number of cases with error for each tie-breaker.

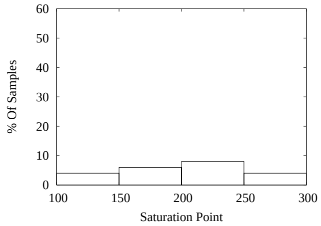
7 Saturation

Unsurprisingly, the high-error trees have arisen from saturation in the pairwise distance data. To this point, we have referred to saturation as being the point where the variance grows too large to make the calculated distances useful. We now use a numerical definition: saturation occurs whenever the true evolutionary distance exceeds the distance computed under our method (which, it should be recalled, is not an edit distance).

We compared reconstructed trees with an RF error greater than 10% to trees with RF errors of less than 5%. In the high RF error category over 91% of the distance matrices show saturation, whereas in the low RF error category 75.5% of the matrices are devoid of any saturation. The distribution of the number of operations where saturation occurs for the high and low RF error groups is shown in Figure 11. Further investigation into



(a) Worst RF Saturation Distribution



(b) Best RF Saturation Distribution

Figure 11: Saturation occurs early for those cases where the RF error is bad.

the properties of the trees in the high and low RF categories revealed little correlation between factors such as tree size, genome size (in genes), or distribution of operations. The major limiting factor in the accurate reconstruction of trees using this distance score is thus definitely the onset of saturation. Since the average genome size in our experiments was approximately 1000 elements, reconstruction is highly accurate when the computed edit distance does not exceed 10% of the genome size and in general performs well until the number of operations exceeds 25% of the genome size. Even in these cases the distance computation performs quite well up to the saturation point, as illustrated in Figure 12. The vertical axis is the difference between the actual and computed distances while the horizontal axis is the ratio of the computed distance to the genome size. Note that in a regime of saturation the computed distance stays the same while the actual distance is rising, so only the positive points should be considered when looking for saturation.

8 Conclusion and Future Directions

We have outlined a method that accurately computes tree distances (true evolutionary distances) under the full range of evolutionary operations between two arbitrary genomes. Our experimental results indicate that the accuracy is excellent up to saturation, which is reached remarkably late—for instance, with genomes of roughly 800 genes, our distance computation remains highly accurate up to 200–250 evolutionary events. Indeed, these distances are accurate enough that the simple neighbor-joining method applied to distance matrices computed with our algorithm reconstructs trees with high accuracy. These findings open up the possibility of reconstructing phylogenies from whole-genome nuclear data, as opposed to the organellar data that have been used so far. However, in order to use more sophisticated methods than neighbor-joining for such reconstructions, the problem of computing good medians must be addressed. While our experiments show that our distance computation is accurate, the accompanying sequence of evolutionary events is only one of many possible sequences (it uses a "canonical form" [12]); hence our level of confidence in the correctness of reconstructed ancestral genomes is low. In order to reconstruct good ancestral genomes, we will need additional biological information, such as boundary constraints (centromere, origin of replication, etc.), length distributions, and sequence data around each gene.

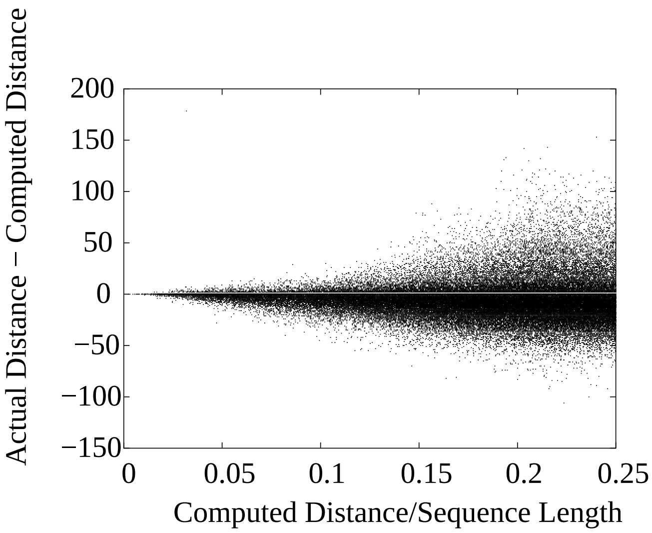


Figure 12: Histogram of the error (actual distance − computed distance) as the ratio of the computed distance to the genome size increases.

9 Acknowledgments

This work is supported by the National Science Foundation under grants DEB 01-20709 (on a subcontract to U. Texas at Austin), IIS 01-13095, IIS 01-21377, ANI 02-03584. and EF 03-31654, and by the National Institutes of Health under grant 2R01GM056120-05A1 (on a subcontract to U. Arizona).

References

[1] G. Andelfinger, C. Hitte, L. Etter, R. Guyon, G. Bourque, G. Tesler, P. Pevzner, E. Kirkness, F. Galibert, and D.W. Benson. Detailed four-way comparative mapping and gene order analysis of the canine ctvm locus reveals evolutionary chromosome rearrangements. *Genomics* 83:1053–1062, 2004.

[2] D.A. Bader, B.M.E. Moret, and M. Yan. A fast linear-time algorithm for inversion distance with an experimental comparison. *J. Comput. Biol.*, 8(5):483–491, 2001.

[3] A. Caprara. Sorting by reversals is difficult. In *Proc. 1st Int'l Conf. on Comput. Mol. Biol. RECOMB'97*, pages 75–83. ACM Press, 1997.

[4] A. Caprara. Formulations and hardness of multiple sorting by reversals. In *Proc. 3rd Int'l Conf. on Comput. Mol. Biol. RECOMB'99*, pages 84–93. ACM Press, 1999.

[5] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Trans. Comput. Biol. & Bioinf.* 2(4): 302–315, 2005.

[6] S. Downie and J. Palmer. Use of chloroplast DNA rearrangements in reconstructing plant phylogeny. In P. Soltis, D. Soltis, and J. Doyle, editors, *Plant Molecular Systematics*, pages 14–35. Chapman and Hall, 1992.

[7] J. Earnest-DeYoung and E. Lerat and B.M.E. Moret. Reversing gene erosion: reconstructing ancestral bacterial genomes from gene-content and gene-order data. In *Proc. 4th Workshop on Algs. in Bioinformatics WABI'04*, volume 3240 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2004.

[8] N. El-Mabrouk. Genome rearrangement by reversals and insertions/deletions of contiguous segments. In *Proc. 11th Ann. Symp. Combin. Pattern Matching CPM'00*, volume 1848 of *Lecture Notes in Computer Science*, pages 222–234. Springer-Verlag, 2000.

[9] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proc. 27th Ann. Symp. Theory of Computing STOC'95*, pages 178–189. ACM Press, 1995.

[10] S.B. Heard. Patterns in phylogenetic tree balance with variable and evolving speciation rates. *Evolution* 50:2141–2148, 1996.

[11] W.-H. Li and D. Graur. Fundamentals of Molecular Evolution *Sinauer and Associates*

[12] M. Marron, K. Swenson, and B.M.E. Moret. Genomic distances under deletions and insertions. *Theoretical Computer Science*, 325(3):347–360, 2004.

[13] B.M.E. Moret and J. Tang, and L.S. Wang and T. Warnow. Steps toward accurate reconstruction of phylogenies from gene-order data. *J. Comput. Syst. Sci.*, 65, 3(2002), 508-525

[14] B.M.E. Moret, J. Tang, and T. Warnow. Reconstructing phylogenies from gene-content and gene-order data. In O. Gascuel, editor, *Mathematics of Evolution and Phylogeny.* pages 321–352, Oxford U. Press, 2005.

[15] L. Nakhleh, B.M.E. Moret, U. Roshan, K. St. John, J. Sun, and T. Warnow. The accuracy of fast phylogenetic methods for large datasets In *Proc. 7th Pacific Symp. on Biocomputing PSB'02*, pages 211–222. World Scientific Pub., 2002.

[16] R. Olmstead and J. Palmer. Chloroplast DNA systematics: a review of methods and data analysis. *Amer. J. Bot.*, 81:1205–1224, 1994.

[17] J. Palmer. Chloroplast and mitochondrial genome evolution in land plants. In R. Herrmann, editor, *Cell Organelles*, pages 99–133. Springer Verlag, 1992.

[18] N.D. Pattengale, K.M. Swenson, and B.M.E. Moret. Approximation algorithms for orthology assignment from gene rearrangement data. Submitted.

[19] L. Raubeson and R. Jansen. Chloroplast DNA evidence on the ancient evolutionary split in vascular land plants. *Science*, 255:1697–1699, 1992.

[20] D.R. Robinson and L.R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.