Approximating the True Evolutionary Distance Between Two Genomes

Krister M. Swenson, Mark Marron, Joel V. Earnest-DeYoung, and Bernard M.E. Moret

Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, USA
`ma_luen@eece.unm.edu`, {`kswenson`, `joeled`, `moret`}`@cs.unm.edu`

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 extend and improve our previous work in two major ways. We generalize our approach to handle duplications as well as insertions and thus enable the computation of distances between two arbitrary genomes; and our new algorithm computes true evolutionary distances, as opposed to the less useful edit distances. 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 a simple neighbor-joining procedure to reconstruct our test trees with high accuracy.

Keywords: inversion distance, reversal distance, genomic distance, Hannenhalli-Pevzner

1 Introduction

Gene-content and gene-order data are becoming more common and are being more commonly used in the study of evolution (see [9]) 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. Moreover, because changes to genomic content or gene order arise infrequently, this data offers the potential for reconstructing very old evolutionary events as well as computing pairwise evolutionary distance between distantly related modern genomes (see, e.g., [5, 10, 11, 12]).

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 sequence) and the sign indicated the strandedness. In turn, evolutionary events can be couched in terms of manipulations of 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 *evolutionary distance* between two genomes. In phylogenetic work, the distance between two taxa is usually an *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 of course nature is not necessarily optimal—the *true evolutionary distance* may involve many more events than the edit distance. Moreover, knowledge of the true evolutionary distance is sufficient to reconstruct the true phylogeny [13].

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 [7]—in fact, the edit distance can then be computed in linear time [2]. On the other hand, computing the minimal edge lengths for 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 conjectured to be NP-hard; El-Mabrouk [6] 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 [8], we gave a polynomial-time approximation algorithm for the 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 inversions); any genes not included in the cover are then treated as deletions from the subject or insertions into the target. Here we generalize this approach to compute the distance between two arbitrary genomes, aiming to reconstruct a sequence of operations that reflects the true evolutionary distance (as ascertained through simulations) rather than the edit 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. As we show, the approximation is so good that we have been able to use the distance matrix produced on a set of genomes with the simple neighbor-joining method to reconstruct trees of reasonable sizes (up to 100 genomes) and very large pairwise distances with high accuracy.

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 Terminology and Review

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.

2.2 The cover

The cover is a mapping relates substrings of genes in the target to the same substring of genes (possibly inverted) in the subject. 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. Computing such a cover appears to require exponential time, however, so we define a *minimum cover* to be a cover that maps the subject to the target with the fewest substrings.

2.3 Difficulties with an arbitrary target

The main difference between our previous work [8] and our new algorithm is the presence of duplications (unrestricted insertions). When building the cover with the identity permutation as the target [8], 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. For example, with subject $1, 2, 3, -5, -2$ and the identity permutation $1, 2, 3, 4, 5$ as target, we have a cover for indices 1 through 3, one for index 5, and one for index 2 of the target; 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 of the target.

3 Constructing a (Nearly) Minimum Cover

The algorithm used in [8] looks for the longest match. 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.

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 1: Choosing a nearly minimal cover.

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 sequences) 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.

To find the longest match, we begin by finding all possible maximal matching substrings and then repeatedly pick the next largest substring to match, 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 matched. 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, each 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 1.

Theorem 1 *At the end of each iteration, M contains all unmatched maximal substrings common to both sequences.*

Proof: By induction on the number of iterations. After 0 iterations this is true from the definition of M. Assume that this is true after i iterations. In the $i + 1^{st}$ iteration the longest match, say m, is removed from M. Only those portions of the matches in M that overlap with m are removed. Since all portions of matches that have not been covered remain in M, we know that all remaining maximal matches are in M. *Q.E.D.*

Theorem 2 *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 of lists, one for each index of 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 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. *Q.E.D.*

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

Lemma 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 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. *Q.E.D.*

Lemma 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. *Q.E.D.*

We are finally ready to prove Theorem 2.
Proof: (of Theorem 2) Initialization takes quadratic time (Lemma 2). Each match in each index list is visited a constant number of times (Lemma 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 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 1 and 3), the running time is $O(n^2)$. *Q.E.D.*

Theorem 3 *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 [8] or [6] can be applied. Both methods run in $O(n^2)$ time. *Q.E.D.*

4 Experimental Design

We used two types of tests to asses 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 topology of the tree matters little 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 a simplistic method for choosing the matches for the duplicated substrings of genes. 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. 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.

4.2 Tree Reconstruction

We tested the performance of our distance functions using neighbor-joining, the canonical distance-based tree reconstruction method. Due to the dearth of real-world trees reconstructed using biological techniques, we were forced to generate trees that would exercise our algorithm over a wide range of plausible models of gene-order evolution. We generated 970 trees using a minor variation on the birth-death model that yielded the requisite diversity of tree topologies. 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 sequences within a general 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 970 trees the percentage of inversions varied from 50% to 90%. The remaining percentages were split evenly between the insertion (duplicating and non-duplicating) and deletion operations. The non-duplicating insertion and duplication percentages were varied over three different tests where 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 upon an edge was 20 with a Gaussian distributed variance of 10 operations.

To generate a tree we began with the identity sequence on 800 genes and performed 200 evolutionary operations on it using the same parameters that are specified for generating the tree. This sequence was then used as the root of the tree. For each node we checked if it should become a child, 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 sequence.

5 Experimental Results

5.1 Pairwise Error

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. 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. We created four such trees with 10, 20, 40, and 60 expected operations per tree edge; these

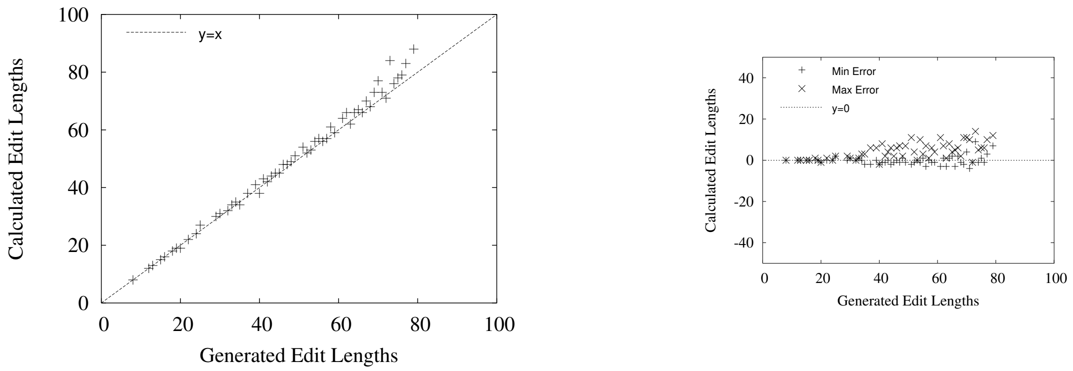

Figure 2: Experimental results for 800 genes with expected edge length 10. Left: generated edit length vs. reconstructed length; right: the variance of computed distances per generated distance.

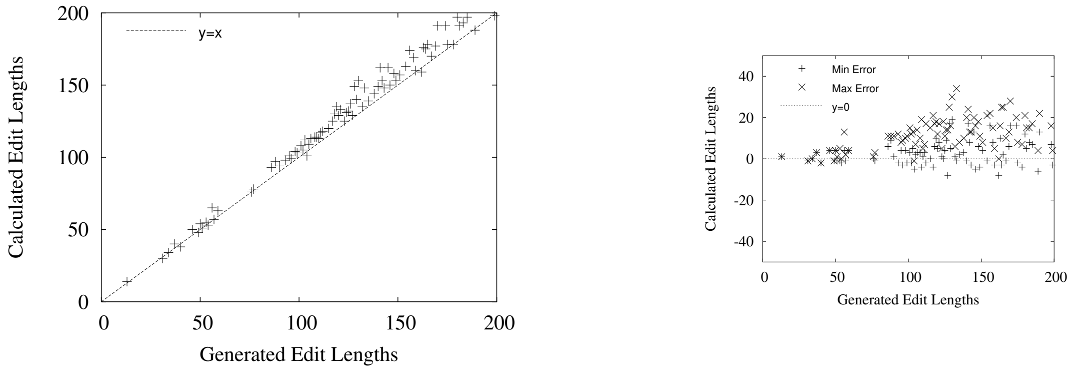

Figure 3: Experimental results for 800 genes with expected edge length 20. Left: generated edit length vs. reconstructed length; right: the variance of computed distances per generated distance.

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 2 through 5 show the results 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 clearly 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 specific to small trees. We ran another series of tests involving trees of around 50 leaves; while the main purpose of these tests was to assess the quality of tree reconstruction using our distance computations, we naturally checked the computed distances against the true distances for these trees as well. Figure 6 shows the same two plots 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

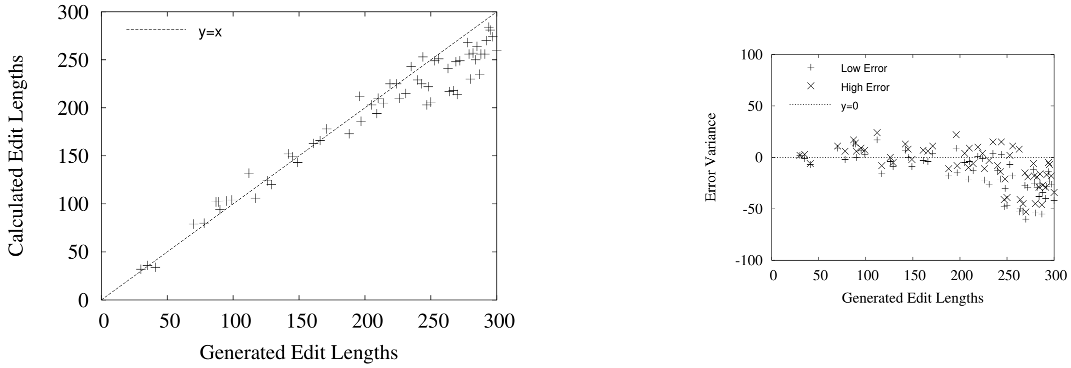

Figure 4: Experimental results for 800 genes with expected edge length 40. Left: generated edit length vs. reconstructed length; right: the variance of computed distances per generated distance.

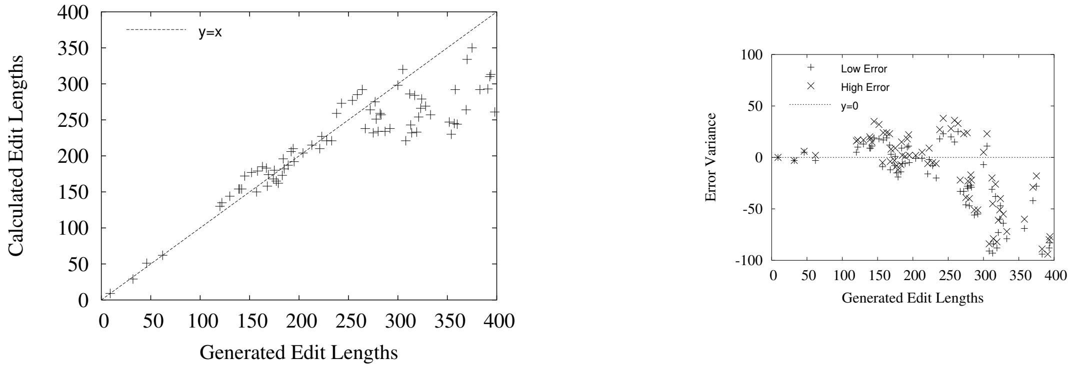

Figure 5: Experimental results for 800 genes with expected edge length 60. Left: generated edit length vs. reconstructed length; right: the variance of computed distances per generated distance.

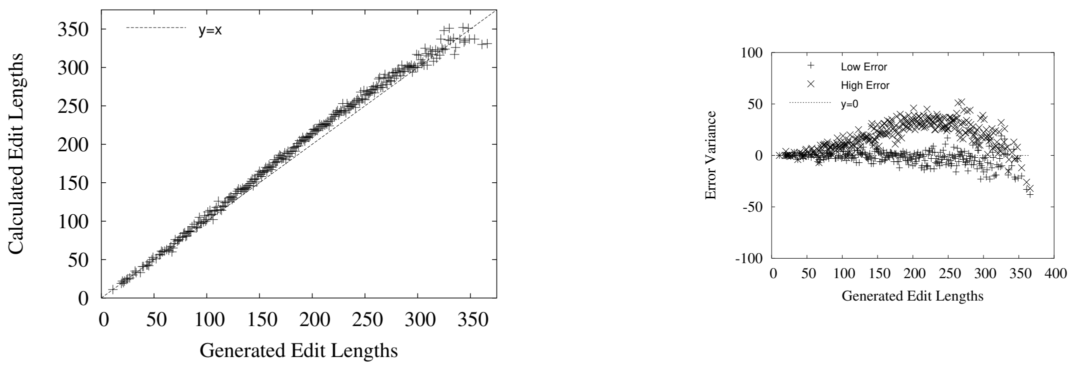

Figure 6: Experimental results for 1200 genes with expected edge length 20. Left: generated edit length vs. reconstructed length; right: the variance of computed distances per generated distance.

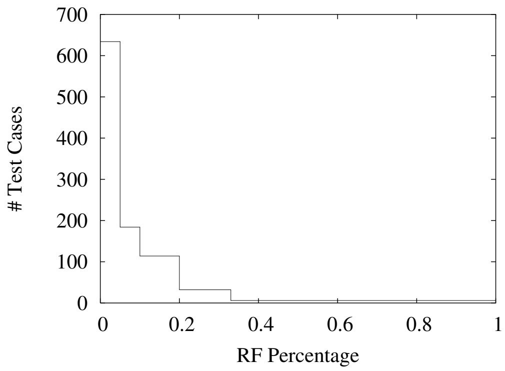

Figure 7: The distribution of RF rates.

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 simplest distance-based reconstruction method, the popular neighbor-joining (NJ) method. 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.

The tree reconstruction performed very well on the generated trees. Approximately 65% of the reconstructed trees had a Robinson-Foulds error of less than 5% and only 15% of the trees had an error above 10%. 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, the results in tree reconstruction can be refined to further improve the results.) As an additional check, we also compared how well our method performs with respect to simply equalizing copied content and applying El-Mabrouk's method. This comparison also gives us an indication of how we handle duplicated gene content and how important it is to handle duplication in estimating true tree distances. We computed a distance matrix for each tree by pairwise removal of all duplicate content and running El-Mabrouk's method on each pair of taxa; these data were then passed to the NJ solver.

Over all 970 trees the equalized El-Mabrouk reconstruction had a lower RF error than ours on only 14%. Furthermore, in three quarters of those cases, the overall RF error was lower than 10%—relatively easy cases; thus our method does better on the harder cases. The average difference in RF error on the trees that our method did worse on was 1.2; while the average difference in RF error on the trees our method did better on was 3.5. Our method thus makes significant improvements on the state of the art. Further, the low error rate in most fo the 14% of cases where our method did not improve results suggests that our randomized selection may be to blame, so that further work on heuristics will reduce this problem.

To examine how well our technique handled copies we compared the RF error of our reconstruction with that of the equalized content El-Mabrouk method as a function of copied content. The graph below indicates that as the number of duplicates increases our method does correspondingly better at reconstructing the tree.

6 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 evolutionary events and reasonably accurate to 250 such operations.

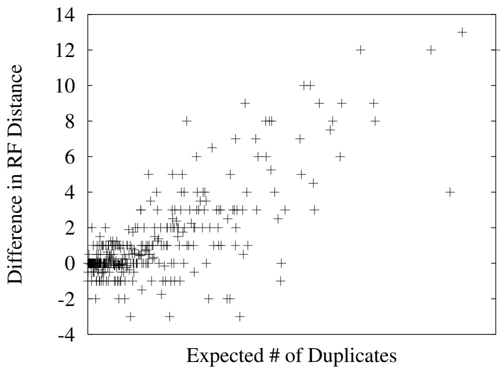

Figure 8: As the expected number of duplicates rise the difference in RF error between El-Mabrouk's and our method rises as well.

Indeed, these distances are accurate enough that the simplest distance-based method for phylogenetic reconstruction, neighbor-joining, reconstructed our test trees with very high accuracy. These findings open up the possibility of reconstructing phylogenies from whole-genome nuclear data, as opposed to the organellar data that has 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 shows that our distance computation is accurate, the accompanying sequence of evolutionary events is only one of many possible sequences (it uses a "canonical form" [8]); hence our level of confidence in the correctness of reconstructed ancestral genomes is low. In order to reconstruct good ancestral genomes, we will need to use additional biological information, including sequence data around each gene.

7 Acknowledgments

This work is supported by the National Science Foundation under grants ACI 00-81404, DEB 01-20709, EIA 01-13095, EIA 01-21377, and EIA 02-03584.

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*, 2004 (in press).

[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. RECOMB97*, 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. RECOMB99*, pages 84–93. ACM Press, 1999.

[5] 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.

[6] 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.

[7] 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.

[8] M. Marron, K. Swenson, and B.M.E. Moret Genomic distances under deletions and insertions *Proc. 9th Int'l Combinatorics and Computing Conf. COCOON'03, in Lecture Notes in Computer Science 2697*, 2003, 537-547.

[9] 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*. Kluwer Acad. Publ., 2004.

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

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

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

[13] D.L. Swofford and G.J. Olsen and P.J. Waddell and D.M. Hillis. Phylogenetic inference. In D.M. Hillis and B.K. Mable and C. Moritz, editors, *Molecular Systematics*, pages 407–514. Sinauer Assoc., 1996.

[14] Wen-Hsiung Li and Dan Graur Fundamentals of Molecular Evolution *Sinauer and Associates*

[15] 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