

# Reconstruction of Ancestral Gene Order Following Large Scale Genome Duplication and Gene Loss

Jun Huan<sup>1</sup>      Jan F. Prins<sup>1</sup>      Wei Wang<sup>1</sup>      Todd J. Vision<sup>2,3</sup>

Departments of <sup>1</sup>Computer Science and <sup>2</sup>Biology  
University of North Carolina, Chapel Hill, NC 27599

<sup>3</sup>Correspondence: tjv@bio.unc.edu, ph: (919) 843-4507, fax: (919) 962-1625

## Abstract

Gene order evolves through gross chromosomal rearrangements, small scale inversions and transpositions, gene duplication, and gene loss. Much research has been done on the calculation of edit distance and on sorting algorithms under a variety of rearrangement models in which the genome may be represented as conserved segments with permuted order and orientation. However, gene loss within otherwise conserved segments, as typically occurs following large scale genome duplication, has not been well studied algorithmically. This has been a major impediment to comparative genomics in certain taxa, such as plants and fish. When large scale genome duplication and gene loss are occurring, how well can we infer both the true gene order within ancestral chromosomal segments and the ancestral ordering of those segments?

We propose a heuristic algorithm for the inference of ancestral gene order in a set of genomes for which at least some genomic segments are partially related by common ancestry to two or more different segments. It does not require gene content and order to be perfectly conserved among segments. First, conserved chromosomal regions are identified using existing pairwise genomic alignment algorithms. Second, segments are iteratively clustered under the control of two parameters, (1) the minimal required number of shared genes between two segments or clusters and (2) the maximal allowed number of rearrangement breakpoints along the lineage leading to each descendant segment. Finally, we compute the estimated ancestral gene order for each cluster.

We evaluate the performance of this algorithm on simulated data that models a genome evolving by large-scale duplication, duplicate gene loss, transposition, translocation, and inversion. The results suggest that ancestral gene orders may be estimated with sufficient accuracy to substantially improve the detection sensitivity of pairwise genomic alignment algorithms.

**Keywords:** gene order, genome rearrangement, genome evolution, comparative mapping

## 1 Introduction

Genome maps can be used to infer common ancestry among segments within the same genome or among different genomes, a practice known as *comparative mapping* [16]. Comparative mapping allows one to leverage the known gene content of model organisms by extrapolating this information to relatives with less tractable genomes, particularly those of economic importance. The wide variety of plant species grown as crops, and the expansion of many plant genomes due to non-genic DNA, makes comparative mapping a particularly important tool for plant geneticists.

In previous work, we provided a mathematical framework and algorithm for identifying homologous segments in a pairwise genome alignment [4]. However, application of this approach is difficult when genomes are distantly related, primarily due to the occurrence of lineage-specific genomic duplication (*i.e.* *polyploidy*) and gene loss (*i.e.* *diploidization*) events. Such events are now known to have occurred fairly regularly in eukaryotic genome evolution [12, 24, 27, 29].

In describing this work, the following terminological conventions will prove useful. The features on a genome map, such as genes, which have a relative ordering on one or more linear chromosomes, are referred to as *markers*. To compare two maps, it is also necessary to identify *homologous* markers that are descended from a common feature in an ancestral genome. Homology may be determined by experimental cross-hybridization in the laboratory, by sequence analysis, or other means [14]. Two segments that share a number of homologous markers with similar ordering can themselves be inferred to be homologous [4]. A pair of such homologous segments is referred to as a *block*. Note that it is possible for a pair of markers to be homologous without the same being true of the segments in which they are situated due to duplication and transposition events that operate on individual markers. Those markers shared between the two members of a block are referred to as *anchors* while those that are unique to only one member are called *singletons*.

The loss of one or the other copy of a large fraction of duplicated gene pairs following segmental or global duplication serves to obscure the presence of many blocks of homology [10]. Such highly diverged blocks, which lack a sufficient number or density of anchors to be identified by pairwise genome alignment, have been referred to as *ghosts* [22]. A number of strategies for the identification of ghosts have been proposed. One is to use multiple, rather than pairwise, genome comparisons [22, 26]. Another is to incorporate reconstructed ancestral genomes directly into pairwise genome alignment algorithms [1, 3]. However, methods for inferring ancestral gene order have not been well studied and little is known about the accuracy of the reconstructions that can be obtained.

In the present work, we study the problem of reconstructing ancestral marker order in the presence of global duplication, marker loss and other rearrangement events that permute marker order. We have developed an algorithm for solving this problem called eAssembler (for *evolutionary Assembler*). Our approach differs from previous work in this area by taking advantage of the overlap among blocks to assemble ancestral segments, or *contigs*, that contain more distinct markers than any single block. The eAssembler algorithm both clusters segments for assembly and derives a reconstructed marker order for each cluster. We assess the performance of the algorithm with simulation experiments and by analysis of real data.

The paper is organized as follows. In Section 2.1, we present a model of genome evolution for multiple chromosomes that incorporates global duplication and gene deletion. In Section 2.2, we consider the choice of a distance function for comparing gene orders. We then describe eAssembler algorithm in Section 3, including several optimizations in our implementation of a procedure proposed by Sankoff and colleagues for solving the so-called *breakpoint median* problem [20]. In Section 4, we report on the quality of the results obtained applying eAssembler to simulated genomes and a dataset from *Arabidopsis thaliana*. We close in Section 5 with a discussion of related research in this area and directions for future work.

## 2 Background

### 2.1 Modeling Genome Rearrangement

In this paper, we are interested in providing a general model for an evolutionary process in which the following five genome rearrangements are involved: deletion, inversion, transposition, translocation and whole genome duplication. These types of events appear to be the major ones affecting gene order evolution in eukaryotes.

For our purpose, we view a chromosome  $p$  as a linear unsigned sequence of markers  $p_1 \dots p_n$  and a genome  $G$  as a list of chromosomes. The genome rearrangements are defined below:

- a *deletion* on  $p$  between indices  $i$  and  $j$  where  $1 \leq i \leq j \leq n$  yields the chromosome  $p' = p_1 \dots p_{i-1} p_{j+1} \dots p_n$ .
- an *inversion* between indices  $i$  and  $j$  of  $p$  where  $1 \leq i < j \leq n$  yields the chromosome  $p' = p_1 \dots p_{i-1} p_j p_{j-1} \dots p_i p_{j+1} \dots p_n$ .

- an *intrachromosomal transposition* with indices  $i, j, k$  where  $1 \leq i \leq j \leq n$  and  $k \notin [i, j]$  places the interval  $p_i \dots p_j$  immediately after  $p_k$ . For example, when  $k > j$ , an intrachromosomal transposition on  $p$  with indices  $i, j, k$  yields the chromosome  $p' = p_1 \dots p_{i-1} p_{j+1} \dots p_k p_i p_{i+1} \dots p_j p_{k+1} \dots p_n$ .
- an *interchromosomal transposition* between two chromosomes  $p = p_1 \dots p_n$  and  $q = q_1 \dots q_m$  with indices  $i, j, k$  where  $1 \leq i \leq j \leq n$  and  $1 \leq k \leq m$  places the interval  $p_i \dots p_j$  immediately after  $q_k$ .
- a *transposition* may be either an intrachromosomal or interchromosomal transposition.
- a *reciprocal translocation* between two chromosomes  $p = p_1 \dots p_n$  and  $q = q_1 \dots q_m$  with indices  $1 \leq i \leq n$  and  $1 \leq j \leq m$  creates two new chromosomes  $p' = p_1 \dots p_{i-1} q_j q_{j+1} \dots q_m$  and  $q' = q_1 \dots q_{j-1} p_i p_{i+1} \dots p_n$ .
- a *whole genome duplication* creates two identical copies of every chromosome  $p$  in genome  $G$ .

We assume that these events occur at random sites within the chromosome(s) with a fixed probability of involvement per gene. For our purposes, the frequency of the event types are held in a fixed ratio. Additional details are provided in Section 4.

The model we present here is generally relevant to genomes having one or more linear chromosomes. Similar formalisms include the *generalized Nadeau-Taylor model* (a single chromosome genome model) described in [13, 15, 28] and the multichromosomal genome model in [2, 18, 25]. The inclusion of whole genome duplication and deletion is a distinguishing feature of the present work.

## 2.2 Distance Functions

Several algorithms have been proposed to measure the evolutionary distance between two genome segments with unequal marker content [7, 23]. Here we use the induced breakpoint distance of [20], defined below, and apply it between a pair of arbitrary segments. There are a number of reasons for this choice. First, it is a generic measure that does not rely on a specific rearrangement cost model. Second, it tolerates a substantial level of deletion between segments, which is not handled well by the brute-force search algorithm proposed by [23]. Third, it is inexpensive to compute. Finally, the related median problem which is central to this work, and which is defined below, has been well studied for the induced breakpoint distance.

Given a segment  $p = p_1 \dots p_n$ , we define a symmetric adjacency relation for symbol set  $\{p_1, \dots, p_n\}$  such that  $p_i$  and  $p_{i+1}$  are *adjacent* for all  $1 \leq i < n$ . Given two segments  $p = p_1 \dots p_n$  and  $q = q_1 \dots q_n$  of identical marker content, if two symbols  $g$  and  $h$  are adjacent in  $p$  but not in  $q$ , this defines a *breakpoint* between  $p$  and  $q$ . The total number of breakpoints between  $p$  and  $q$  is the *breakpoint distance* between  $p$  and  $q$ . Given two segments  $p = p_1 \dots p_n$  and  $q = q_1 \dots q_m$  (possibly with differing marker content), the *reduced segment*  $p'$  of  $p$  is the segment obtained by removing all singletons from  $p$ . The *induced breakpoint distance* (or *distance* for short) between  $p$  and  $q$  is the breakpoint distance between reduced segments  $p'$  and  $q'$ . We denote by  $d(p, q)$  the distance between two segments  $p$  and  $q$ . A *median* of a set  $P$  of segments is a string  $m$  composed of all the distinct markers in the segments in  $P$ . The *cost* of the median  $m$  is the sum of the pairwise distances between  $m$  and each segment in  $P$ . The *cost variance* of  $m$  is the variance of the distances between  $m$  and each segment in  $P$ .

## 3 Algorithm Design and Implementation

### 3.1 Overview

The eAssembler algorithm is designed to infer the ancestral order of markers in a set of descendant genome segments. It does this by clustering the segments. Initially the algorithm places each segment in its own cluster. The algorithm iteratively joins two existing clusters  $P$  and  $Q$  that satisfy two conditions, governed by parameters  $t$  and  $k$ . First,  $P$  and  $Q$  must share at least  $k$  markers. Second, there must exist a median

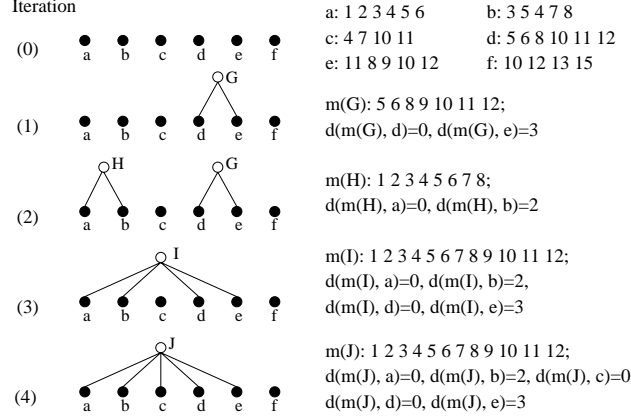


Figure 1: Application of eAssembler to a set of six segments. Four iterations of the clustering process are shown from top to bottom. Filled circles represent input strings (with sequences given to the right), and open circles represent clusters. A median that satisfies the assembly parameters ( $t = 3, k = 3$ ) for each cluster is shown to the right. Also shown are the distances from the median to each segment in the cluster.

$m$  such that the distance between every segment in the two clusters and  $m$  is no greater than  $t$ . If there are multiple such joins available among the current clusters, a join with the maximal number of shared markers is chosen. The clustering algorithm stops when no further joins are possible. An important consequence of this clustering procedure is that a join may occur even in cases where no pair of segments (one from each cluster) have sufficient overlap to be joined by themselves. This contrasts with previously published approaches in which clusters can consist of only two segments [1, 22]. The final step of the algorithm is to compute the *optimal median* for each cluster. This is defined as one of the (potentially large number of) medians that satisfies the distance constraint and also has minimal cost. Among these, we choose a median having minimal cost variance. The optimal median represents the inferred ancestral string of marker symbols. By analogy with sequence assembly (e.g. [9]), we refer to the optimal median of each cluster as a *contig*.

Figure 1 illustrates the reconstruction of two contigs from six segments  $a, \dots, f$  with composition shown at the top of right of the figure. The reconstruction uses parameters  $k = t = 3$ . The six segments initially form singleton clusters  $A = \{a\}, B = \{b\}, \dots, F = \{f\}$ . At the first iteration, clusters  $D$  and  $E$  are joined to form cluster  $G$  because they share the largest number of markers (they share four markers which is greater than  $k$ ), and there exists a median  $m(G)$  for which the distance to each segment is no more than  $t$  (i.e.  $d(m(G), d) \leq t$  and  $d(m(G), e) \leq t$ ). In the next iteration,  $A$  and  $B$  have the greatest overlap among existing clusters and have a median satisfying the distance bound, so are joined to form cluster  $H$ . In the third iteration, clusters  $G$  and  $H$  have maximal overlap, and the overlap is sufficient for the clusters to be joined. Note that in the case of this proposed new cluster,  $I$ , we require of its median that  $d(m(I), a) \leq t, d(m(I), b) \leq t, d(m(I), c) \leq t$  and  $d(m(I), d) \leq t$ . The final join is between clusters  $C$  and  $I$ , forming cluster  $J$ . The only other remaining possibility would be to join  $F$  and  $J$ , but they share only two markers, which is less than  $k$ . Thus, the algorithm proceeds to the final step of constructing the contigs. The optimal median for cluster  $J$  is the sequence  $1, \dots, 12$ , which is at distance at most three from all segments in  $J$  and has minimal cost and minimal cost variance among all such medians. Hence, it is the contig assembled for cluster  $J$ . The cluster  $F$  contains only segment  $f$ , which is a trivial contig for the cluster.

### 3.2 eAssembler Algorithm

Here we present the eAssembler algorithm more formally.

#### Notation:

$p, q, s$	segments (strings of markers)
$P, Q, P'$	clusters (sets of segments)
$p \cap q$	markers common between segments $p$ and $q$ : $\{c \mid c \in p \wedge c \in q\}$
$P \cap Q$	markers common between clusters $P$ and $Q$ : $\{c \mid p, q \text{ such that } p \in P \wedge q \in Q \wedge c \in p \cap q\}$

**Algorithm: eAssembler** (high-level)

**input:** A collection of segments  $s_1, \dots, s_n$ , a distance limit  $t$ , and a similarity threshold  $k$ .

**output:** A set of contigs  $C$ .

```

 $S \leftarrow \{\{s_1\}, \dots, \{s_n\}\}$ 
repeat
  let candidates  $\leftarrow$  list of  $(P, Q)$  where  $P, Q$  are distinct clusters in  $S$  with  $|P \cap Q| \geq k$ ,
    in order of decreasing  $|P \cap Q|$  value
  find first  $(P, Q)$  in candidates for which  $\exists$  median  $m$  such that  $d(m, w) \leq t \forall w \in (P \cup Q)$ 
  if none found then break
   $S \leftarrow S - \{P, Q\} \cup (P \cup Q)$ 
end repeat
 $C \leftarrow \{\text{contig}(P) \mid P \in S\}$ 

```

### 3.3 Implementation and Optimizations

The key step in the eAssembler algorithm is the search to determine whether there exists a median that satisfies the distance constraint for a proposed join. In our implementation, we use the *furthest insert* heuristic proposed by Sankoff and colleagues [20] for computing the optimal induced breakpoint median among strings containing non-identical sets of symbols. The NP-hardness of that problem was established by [17].

In Sankoff's approach, the median for a cluster  $P$  is built iteratively by inserting one new marker with each iteration. First, an initial string of one marker is created by randomly selecting a marker from among all the segments in  $P$ . Second, an insertion of each of the remaining distinct markers is evaluated at each position in the current median. The insertion that minimizes the *cost* of the median is recorded for each marker. This is the *optimal insertion* for that marker. One of the markers with an optimal insertion having maximal cost is randomly chosen to be inserted into the median at the recorded position. The evaluation of each remaining marker is repeated until all markers have been inserted into the median. Remarkably, the practice of inserting the marker with the greatest cost actually leads to a median that itself has very nearly minimal cost [20]. Note that it is not necessary, with this scheme, to determine the relative orientations of the segments being assembled. In our implementation, we preferentially insert markers with low cost variance when choosing markers with equivalent optimal insertion costs. This serves to guide the search toward medians that are likely to satisfy the distance constraint for all component segments. We obtain many estimates of the optimal median by running the search algorithm multiple times and choose a string with minimal cost among the resulting set of medians.

The procedure above is the most time consuming step in eAssembler. We made four modifications in order to improve its performance. We denote by  $T$  the set of all segments that are to be assembled by eAssembler in the following discussion.

The runtime of the algorithm is quadratic in the number of distinct markers among the segments, so the first modification is simply to remove all singleton markers from the segments in  $T$ . Reinserting these markers at the end, without introducing additional breakpoints, is trivial, and hence the computed median is independent of these symbols.

We can further reduce the number of distinct markers by recoding as two markers any substrings of markers which are conserved in all segments in which any of the markers appear. More precisely, a string of markers  $p = p_1, \dots, p_n$  is a *run* in  $T$  iff for each  $c \in p$  and  $s \in T$ , we have  $c \in s \Rightarrow (p \text{ occurs in } s) \vee$

( $p$  occurs in  $s^R$ ) where  $s^R$  is the reverse of  $s$ . The run  $p$  can be replaced in a segment  $s$  with  $p_1p_n$  or  $p_np_1$ , depending on the direction of  $p$  in  $s$ .

The third modification takes advantage of a multiprocessor architecture to obtain multiple estimates of the median simultaneously, before extracting the optimal median.

Our last modification is based on the following observation. If two clusters, after being joined, fail to satisfy the distance constraint, then no pairs of clusters containing them could satisfy the condition. Thus, we may avoid testing the join between two clusters  $P$  and  $Q$  when any subsets of  $P$  and  $Q$  have previously failed. In our implementation, we used a boolean function  $\text{JoinCandidate}(P, Q)$  to keep whether cluster  $P$  and  $Q$  have been tested.

The fully optimized version of the program is presented below. It incorporates a compress function to reduce the number of markers in a cluster following the first two optimization steps discussed above. It also utilizes  $\text{ParBPMedian}$ , which is a parallel version of the search scheme for the distance constrained optimal median.  $\text{eAssembler}$  takes a set of blocks as input since segments are identified by some pairwise genome alignment algorithm and always pair with each other [4, 18]. With all the four modifications, we obtain approximately one order of magnitude of speedup.

**Algorithm: eAssembler** (optimized implementation)

**input:** A collection of blocks  $b_1(s_{11}, s_{12}), \dots, b_n(s_{n1}, s_{n2})$ , a distance limit  $t$ , and a similarity threshold  $k$ .

**output:** A set of contigs  $C$ .

```

 $S \leftarrow \{\{s_{11}\}, \{s_{12}\}, \dots, \{s_{n1}\}, \{s_{n2}\}\}$ 
 $\forall P, Q \in S, \text{JoinCandidate}(P, Q) \leftarrow \text{true}$ 
repeat
  find distinct  $P, Q \in S$  such that  $\text{JoinCandidate}(P, Q)$  and such that  $|P \cap Q|$  is
    maximal and at least  $k$ .
  if no such  $P, Q$  available break
   $P' \leftarrow P \cup Q$ 
   $m \leftarrow \text{ParBPMedian}(\text{compress}(P'))$ 
  if  $m \neq \epsilon$  then
     $S \leftarrow S \cup \{P'\} - \{P, Q\}$ 
     $\forall W \in S, \text{JoinCandidate}(P', W) \leftarrow \text{JoinCandidate}(P, W) \wedge \text{JoinCandidate}(Q, W)$ 
  else
     $\text{JoinCandidate}(P, Q) \leftarrow \text{false}$ 
end repeat
 $C = \{\text{ParBPMedian}(P) \mid P \in S\}$ 

```

## 4 Results

We simulated genomes evolving over time and analyzed the final products using  $\text{eAssembler}$  in order to assess the quality of the reconstructions obtained. Our model of evolution in a multiple chromosome genome (see Section 2.1) combines global duplication and single-marker deletion with three types of operations acting on gene order alone (inversion, transposition, and reciprocal translocation). In our simulations, the initial genome consisted of a single chromosome containing four hundred markers. The genome then experienced two rounds of global duplication. After each duplication, the genome was subject to deletion, inversion, transposition and reciprocal translocation according to specified probabilities. The end result, in each simulation, was a genome of four chromosomes with variable marker content. The resultant genomes were then preprocessed, using the pairwise genome alignment algorithm FISH [4] with default parameters, to provide sets of duplicated segments which, together with the identities of all duplicated genes, served as input for

Table 1: Parameters used in simulation Experiments 1-5. Twelve combinations of  $k$  and  $t$  were evaluated in each experiment:  $k$  varied from 2-4 and  $t$  from 4-7. For each experiment, the proportion of rearrangement events is given as deletions (D) : inversions (I) : transpositions (Tp) : translocations (T). Ten replicates were evaluated for each combination of  $k$ ,  $t$  and I:Tp:Tl:D. Inversion and transposition lengths were sampled from a normal distribution with mean= 6 and standard deviation= 2.

experiment	name	events/marker	D:I:Tp:Tl
1	base	0.2	12:6:1:1
2	high deletion	0.26	18:6:1:1
3	high inversion	0.23	12:9:1:1
4	high transposition	0.22	12:6:3:1
5	high translocation	0.22	12:6:1:3

eAssembler. This preprocessing step is a hidden factor affecting the quality of the ancestral reconstructions that are obtained by eAssembler, but we do not study it further here.

We performed two sets of simulation experiments. In the first set of five experiments, we analyzed the accuracy of reconstructions for a roughly constant number of rearrangement events, varying both the relative frequency of the different types of events and also the assembly parameters  $k$  and  $t$  (Table 1). For each parameter setting, ten different genomes were simulated and analyzed using FISH and eAssembler. This allowed us to assess the sensitivity of the reconstruction to each type of rearrangement event singly (deletion, inversion, transposition and translocation), relative to a base condition. It also allowed us to understand how  $k$  and  $t$  interacted to affect the quality of the reconstruction obtained.

In the second set of experiments, we examined, for three fixed ratios of the rearrangement events, how the reconstruction quality varied with the total number of events (from 0.04/marker to 0.32/marker). Ten genomes were simulated and analyzed by FISH and eAssembler (with  $k = 6$  and  $t = 3$ ) for each parameter setting.

We used two assessments, coverage and normalized breakpoint distance, to measure the quality of the reconstructions. We calculate coverage as the ratio of the distinct symbols of all contigs to the number of genes in the original genome. Since each block has two copies, only contigs assembled from at least three segments are counted. For a contig, the normalized breakpoint distance (hereafter referred to simply as the *BP distance*) is defined as the ratio of its induced breakpoint distance to its length in markers.

We also applied eAssembler to a published dataset of 103 duplicated blocks and 18,569 pairs of duplicated protein coding genes in *Arabidopsis thaliana* [27]. These data are available from [http://www.bio.unc.edu/faculty/vision/lab/arab/science\\_supplement](http://www.bio.unc.edu/faculty/vision/lab/arab/science_supplement). In this case, since the ancestral genome is not known, we cannot calculate coverage and mean BP distance. Instead, we measure the number of segments in each contig in order to quantify the increase in information obtained using eAssembler over that obtained by pairwise genomic alignment alone.

#### 4.1 Sensitivity test of the eAssembler

The distributions of contig length and BP distance before and after the assembly process are shown for the ten combined replicates of Experiment 1 (see Table1) in Figure 2. Here,  $k = 6$  and  $t = 2$ . In this example, we obtain a 88% increase in mean contig length (from 10.6 to 19.9) with only a 21% increase in mean BP distance (from 0.260 to 0.316).

To examine the sensitivity of ancestral reconstruction quality to different rearrangement regimes, we carried out four additional experiments in which we varied the relative proportions of deletion, inversion, transposition and translocation (Figure 3C-I).

Figures 3A and 3B show the mean distance and coverage as a function of  $k$  and  $t$  for the basal conditions

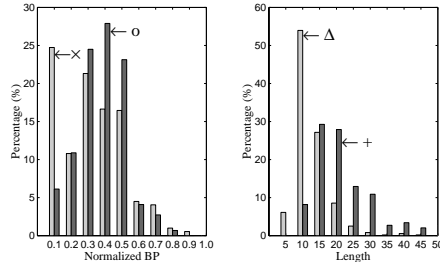


Figure 2: Comparison of output from pairwise genome alignment (FISH), and after further analysis by eAssembler. Left: length distribution of blocks/contigs, in number of markers. Right: distribution of BP distances. Light gray bars show FISH output, dark gray bars show eAssembler output.  $\Delta$  and +: mean lengths of blocks and contigs, respectively.  $\times$  and 0: mean BP distances of blocks and contigs, respectively.

(Experiment 1, Table 1). As one might expect, both coverage and BP distance are increased by low  $k$  and high  $t$ . Within the parameters ranges studied, coverage is more sensitive to a unit change in  $k$  while distance is more sensitive to a unit change in  $t$ . Depending on the exact values of  $k$  and  $t$ , coverage is 46-78% of the ancestral genome with the mean BP distance among contigs ranging from 0.3-0.4 per marker.

In Experiment 2, the level of deletion was increased by 50% while holding the frequencies of other rearrangements at the basal values (Figures 3C and 3D). Coverage in the eAssembler output was reduced by 19-50%, the most severe reductions occurring at high  $k$  and low  $t$ . BP distance increased less dramatically (0-15%). The increases were proportionally the greatest at low  $k$  and high  $t$ . There was essentially no difference in BP distance at  $k = 7$ ,  $t = 2$ , for which parameters coverage was most reduced.

In Experiment 3, inversion frequency was increased by 50% over the basal value (Figures 3E and 3F). The decline in coverage with increasing  $k$  was considerably steeper in the presence of a high inversion frequency than for any other regime. Also, the sensitivity of BP distance to unit changes in  $t$  was much reduced. Apart from these trends, the results were similar to those obtained under the high deletion regime.

Figures 3G and 3H show the results of Experiment 4, in which the level of transposition was increased 3-fold while the rest of the conditions hold the same. The coverage of the reconstruction degraded substantially relative to the basal conditions, with coverage ranging from 0.19-0.41. Normalized BP distance decreased slightly, from 10

Lastly, a high translocation rate makes little difference to the BP distance relative to basal conditions (Figure 3I). Curiously, though, as shown in Figure 3J, coverage actually increased somewhat at some points (around 10% at  $k = 4$  and  $t = 2$ ).

To summarize the results of Experiments 1 through 5, we conclude that deletion and inversion have the most severe effects on both coverage and BP distance, while these quantities are relatively insensitive to the frequency of translocations and transpositions. In addition, some unusual patterns of sensitivity to  $k$  and  $t$  have been observed that are not well understood. Table 2 shows that, for BP distance, the experiments significantly differ from one another, as do the results for different values of  $k$  and  $t$ . However, there are no significant interactions among these factors. By contrast, when the results for coverage are analyzed in this way, the significant effects of experiment,  $k$  and  $t$  are complicated by two different pairwise interactions:  $\text{experiment} \times k$  and  $\text{experiment} \times t$ .

It is difficult to know whether the parameter values we have used in these simulations are realistic ones. We used estimates from a variety of eukaryotic genomes to inform our choices. Genome comparisons between the nematodes *Caenorhabditis elegans* and *C. briggsae* have led to an estimate for the ratio of inversions : transpositions : translocations of  $\sim 1:2:1$  [5]. The higher frequency of small inversions used in our simulations is supported by a comparison of *Saccharomyces cerevisiae* and *Candida albicans* and other studies reviewed in [21]. The relatively high rate of marker loss after duplication in these simulations is designed to mimic what is observed in higher plant genomes, where most differences in gene order between homologous segments appear to be due to gene deletion rather than inversion, though transposition may also

Table 2: Analysis of variance for data from Experiments 1 through 5. expt: experiment; d.f.: degrees of freedom.

factor	d.f.	BP distance		coverage	
		$F$	$p$	$F$	$p$
expt	4	64.12	< 0.0001	287.82	< 0.0001
$k$	3	28.91	< 0.0001	163.49	< 0.0001
$t$	2	33.07	< 0.0001	33.03	< 0.0001
expt $\times k$	12	0.77	ns	2.84	0.0008
expt $\times t$	8	1.46	ns	2.06	0.0380
$k \times t$	6	0.91	ns	1.73	ns
expt $\times k \times t$	24	1.26	ns	1.20	ns

Table 3: Results from assembly of the Arabidopsis dataset. Given below is the number of clusters containing the number of segments given above.

2	3	4	5	6	7	8	total
24	18	13	3	1	0	1	60

be playing a role [10, 11].

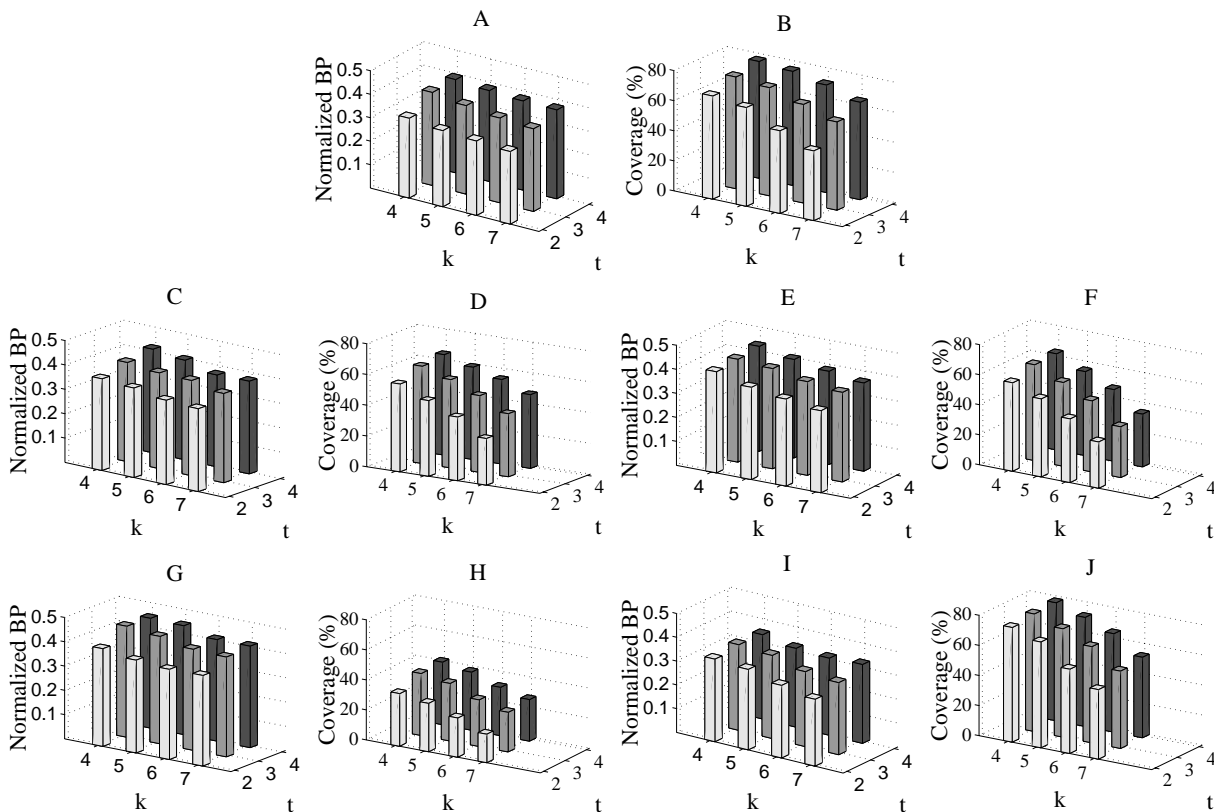
In addition, we explored the effects of fixing the relative rates of the different rearrangement types while varying the total number of rearrangements that occur (Figure 4). Coverage decreased nearly linearly with the number of rearrangement events. BP distance increased in a similarly linear fashion. Both measures are bounded, however, and so must necessarily plateau beyond the range examined here. It was also found, not surprisingly, that the variance among replicates in both coverage and BP distance increased as the events per marker increased.

To test the eAssembler algorithm on real data, we applied it to a dataset of duplicated blocks in the genome of *Arabidopsis thaliana* [27]. The results of this analysis will be reported in more detail elsewhere. Here we simply note that, even with for very strict parameters ( $k = 10$ ,  $t = 5$ ) over half of the contigs contain more than two segments (Table 3). Interestingly, not all contigs contain even numbers of segments. The longest contig contains eight segments and over 500 distinct markers. The longest of the component segments is only 134 markers long, which demonstrates a substantial expansion of contig size through the use of eAssembler. With data from additional species, the number of segments per contig, and the length of the contigs, would be expected to increase yet further.

## 5 Discussion

Two other recent studies have explored the use of a reconstructed ancestral genome in the case of the model plant *Arabidopsis thaliana* [1, 3]. *Arabidopsis* is of interest because, having been sequenced, it is the logical hub for comparative mapping to many agriculturally important plant species. However, it is known to have undergone multiple large-scale duplication events [27]. In both studies, the authors first identified duplicated blocks in the *Arabidopsis* genome, using methods analogous to FISH, then generated a pseudo-ancestral genome by replacing the two segments in each duplicated block with a single segment having the union of the gene content in both members. By reapplying the search for ancient duplicated regions within the pseudo-ancestral genome, Blanc and colleagues were able to identify approximately twice as many duplicated blocks (from yet older duplication events) as in the original analysis. Both papers report a substantially improved ability to detect related chromosome segments in other species when species divergence predates the age of the duplications within *Arabidopsis*. Our approach in this paper may be seen as a

Figure 3: BP distance (left) and coverage (right) as a function of the overlap threshold,  $k$ , and distance bound,  $t$ , for experiments 1-5. Parameter settings unique to each experiment are given in Table 1. A, B: Base conditions; C, D: High deletion; E, F: High inversion; G, H: High transposition; I, J: high translocation.

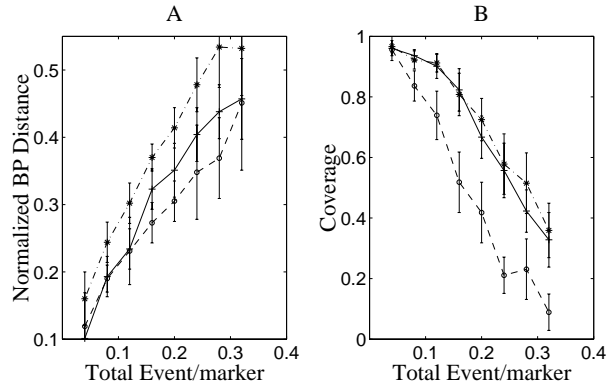


generalization that does not limit the merge operation to pairs of non-overlapping blocks.

Until recently, approaches for reconstructing ancestral gene orders have assumed that each block of duplication brackets a unique pair of genomic regions (*i.e.* there is no spatial overlap among blocks) (*e.g.* [8, 18]). However, in organisms that have undergone multiple genome duplications (*e.g.* *Arabidopsis* [27]), and in comparisons involving more than two genomes, many blocks are expected to overlap. Van de Peer and colleagues [22, 26] identify ghost blocks by performing transitive closure on sets of overlapping blocks. In a similar manner, eAssembler may join segments into clusters that do not share a sufficient number of anchors when considered in isolation. eAssembler also takes advantage of the overlap among segments to infer the gene order in longer ancestral segments than are represented by any single block, as shown by the analysis of the *Arabidopsis* data (Table 3). This latter feature will allow more extensive segments of homology to be detected via comparative mapping, as homologous segments are not necessarily limited in size by the boundaries of rearrangement breakpoints unique to one lineage or the other.

Several refinements to the algorithm are currently being explored. One we wish to direct the clustering process to assemble blocks at particular levels of phylogenetic relationship so that reconstructions can be estimated for ancestors that existed at varying times in the past. Two, improved reconstructions may be possible if we allow the clustering parameters to vary for clusters at different levels (*e.g.* allow  $k$  to decrease and  $t$  to increase as clustering progresses), or to scale for different sized input segments. Ideally, we would wish to statistically guide the clustering process [6]. Three, while our choice of median is a sensible one, typically there are a large number of other medians with equivalent, or nearly equivalent, cost and variance. Evaluating this population of medians probabilistically could allow us, among other things, to test hypotheses about the gene arrangement in ancestral genomes. Four, it will be necessary in dealing with more complex

Figure 4: Normalized BP distance and coverage under varying numbers of rearrangement events. The three lines show results for different fixed proportions of deletion : inversion : transposition : translocation. 12:6:1:1(solid), 11:3:3:3 (dashed), 9:9:1:1 (dot-dashed) Ten genomes were simulated and assembled for each point. Assembly parameters were  $k = 6, t = 3$ . Vertical bars show one standard deviation.



genomes to accommodate multiple occurrences of the same marker within a segment, which has proved to be a difficult problem in similar contexts [23, 19].

From our initial results, we conclude that eAssembler and its derivatives should substantially augment our ability to identify segments of homology among highly divergent genomes. Thus, we aim to incorporate this software into an analysis pipeline for large-scale comparative mapping among plants as part of a project to develop a regularly-updated internet-accessible plant comparative genomics database.

## References

- [1] G. Blanc, K. Hokamp, and K. H. Wolfe. A recent polyploidy superimposed on older large-scale duplications in the arabidopsis genome. *Genome Research*, 13:137–44, 2003.
- [2] G. Bourque and P. A. Pevzner. Genome-scale evolution: Reconstructing gene order in the ancestral species. *Genome Research*, 12:26–32, 2002.
- [3] J. E. Bowers, B. A. Chapman, J. Rong, and A. H. Paterson. Unraveling angiosperm genome evolution by phylogenetic analysis of chromosomal duplication events. *Nature*, 422:433–438, 2003.
- [4] P. P. Calabrese, S. Chakravarty, and T. J. Vision. Fast identification and statistical evaluation of segmental homology in comparative maps. *Bioinformatics*, in press, 2003.
- [5] A. Coghlan and K. Wolfe. Fourfold faster rate of genome rearrangement in nematodes than in drosophila. *Genome Research*, 16:857–867, 2002.
- [6] D. Durand and D. Sankoff. Tests for gene clustering. *Journal of Computational Biology* in press, 2002.
- [7] El-Mabrouk. Genome rearrangement by reversals and insertions/deletions of contiguous segments. *Combinatorial Pattern Matching, 11th annual symposium, Lecture Notes in Computer Science*, 1848:222–34, 2000.
- [8] N. El-Mabrouk, D. Bryant, and D. Sankoff. Reconstructing the pre-doubling genome. *RECOMB*, 3:154–63, 1999.
- [9] X. Huang. A contig assembly program based on sensitive detection of fragment overlaps. *Genomics*, 14:18–25, 1992.
- [10] H.-M. Ku, T. Vision, J. Liu, and S. D. Tanksley. Comparing sequenced segments of the tomato and arabidopsis genomes: Large-scale duplication followed by selective gene loss creates a network of synteny. *Proc. of the National Academy of Sciences USA*, 97:9121–9126, 2000.

- [11] K. Mayer, G. Murphy, R. Tarchini, R. Wambutt, G. Volckaert, T. Pohl, A. Dsterhft, W. Stiekema, K.-D. Entian, N. Terryn, K. Lemcke, D. Haase, C. R. Hall, A.-M. van Dodeweerd, S. V. Tingey, H.-W. Mewes, M. W. Bevan, and I. Bancroft. Conservation of microstructure between a sequenced region of the genome of rice and multiple segments of the genome of *Arabidopsis thaliana*. *Genome Research*, 11:1167–74, 2001.
- [12] A. McLysaght, K. Hokamp, and K. H. Wolfe. Extensive genomic duplication during early chordate evolution. *Nature Genetics*, 31:200–4, 2002.
- [13] B. Moret, L. Wang, T. Warnow, and S. K. Wyman. New approaches for reconstructing phylogenies from gene order data. *ISMB*, 9:165–173, 2001.
- [14] J. H. Nadeau and D. Sankoff. Counting on comparative maps. *Trends in Genetics*, 14:495–501, 1998.
- [15] J. H. Nadeau and B. A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proc. Natl. Acad. Sci. USA*, 81:814–818, 1984.
- [16] S. J. O’Brien, M. Menotti-Raymond, W. J. Murphy, W. G. Nash, J. Wienberg, R. Stanyon, N. G. Copeland, N. A. Jenkins, J. E. Womack, and J. A. Marshall Graves. The promise of comparative genomics in mammals. *Science*, 286:458–462, 1999.
- [17] I. Pe’er and R. Shamir. The median problems for breakpoints are NP-complete. *Elec. Colloq. on Comput. Complexity*, 71, 1998.
- [18] P. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution: Lessons from human and mouse genomes. *Genome Research*, 13:37–45, 2003.
- [19] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15:909–917, 1999.
- [20] D. Sankoff, D. Bryant, M. Deneault, B. F. Lang, and G. Burger. Early eukaryote evolution based on mitochondrial gene order breakpoints. *Journal of Computational Biology*, 7:521–536, 2000.
- [21] C. Seoighe, N. Federspiel, T. Jones, N. Hansen, V. Bivolarovic, R. Surzycki, R. Tamse, C. Komp, L. Huizar, R. W. Davis, S. Scherer, E. Tait, D. J. Shaw, D. Harris, L. Murphy, K. Oliver, K. Taylor, M. A. Rajandream, B. G. Barrell, and K. H. Wolfe. Prevalence of small inversions in yeast gene order evolution. *Proc of the National Academy of Sciences USA*, 97:14433–14437, 2000.
- [22] C. Simillion, K. Vandepoele, M. C. Van Montagu, M. Zabeau, and Y. Van de Peer. The hidden duplication past of *arabidopsis thaliana*. *Proc Natl Acad Sci USA*, 99:13627–32, 2002.
- [23] J. Tang and B. Moret. Phylogenetic reconstruction from gene rearrangement data with unequal gene content. *U. New Mexico TR-CS-2003-9*, 2003.
- [24] J. S. Taylor, I. Braasch, T. Frickey, A. Meyer, and Y. Van de Peer. Genome duplication, a trait shared by 22,000 species of ray-finned fish. *Genome Research*, 13:382–90, 2003.
- [25] G. Tesler. Efficient algorithm for multichromosomal genome rearrangements. *Journal of Computer and System Sciences*, 65:587–609, 2002.
- [26] K. Vandepoele, C. Simillion, and Y. Van de Peer. Detecting the undetectable: Uncovering duplicated segments in arabidopsis through rice. *Trends in Genetics*, 18:606–608, 2002.
- [27] T. Vision, D. Brown, and S. Tanksley. The origins of genomic duplications in arabidopsis. *Science*, 290:2114–2117, 2000.
- [28] L. S. Wang. Genome rearrangement phylogeny using weighbor. *WABI*, 2:112–25, 2002.
- [29] K. H. Wolfe and D. C. Shields. Ancient duplication of the entire yeast genome. *Nature*, 387:708–13, 1997.