

Sequence analysis

GR-Aligner: an algorithm for aligning pairwise genomic sequences containing rearrangement eventsTe-Chin Chu^{1,2}, Tsunglin Liu³, D. T. Lee^{1,4,5}, Greg C. Lee²
and Arthur Chun-Chieh Shih^{1,4,*}¹Institute of Information Science, Academia Sinica, ²Department of Computer Science and Information Engineering, National Taiwan Normal University, ³Biodiversity Research Center, ⁴Research Center for Information Technology Innovation, Academia Sinica and ⁵Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Received on January 12, 2009; revised on June 9, 2009; accepted on June 11, 2009

Advance Access publication June 19, 2009

Associate Editor: John Quakenbush

ABSTRACT

Motivation: Homologous genomic sequences between species usually contain different rearrangement events. Whether some specific patterns existed in the breakpoint regions that caused such events to occur is still unclear. To resolve this question, it is necessary to determine the location of breakpoints at the nucleotide level. The availability of sequences near breakpoints would further facilitate the related studies. We thus need a tool that can identify breakpoints and align the neighboring sequences. Although local alignment tools can detect rearrangement events, they only report a set of discontinuous alignments, where the detailed alignments in the breakpoint regions are usually missing. Global alignment tools are even less appropriate for these tasks since most of them are designed to align the conserved regions between sequences in a consistent order, i.e. they do not consider rearrangement events.

Results: We propose an effective and efficient pairwise sequence alignment algorithm, called GR-Aligner (Genomic Rearrangement Aligner), which can find breakpoints of rearrangement events by integrating the forward and reverse alignments of the breakpoint regions flanked by homologically rearranged sequences. In addition, GR-Aligner also provides an option to view the alignments of sequences extended to the breakpoints. These outputs provide materials for studying possible evolutionary mechanisms and biological functionalities of the rearrangement.

Availability: http://biocomp.iis.sinica.edu.tw/new/GR_Aligner.htm

Contact: arthur@iis.sinica.edu.tw

1 INTRODUCTION

Sequence alignment plays important roles in the fields of molecular biology and evolution, such as conserved region identification among species (Lyons and Freeling, 2008; Zimmermann *et al.*, 2008), protein structure prediction (Pei and Grishin, 2007; Tomomura *et al.*, 2007) and phylogeny analysis (Janecka *et al.*, 2007). By comparing cross-species genomic sequences, researchers have identified different types of genome-wide evolutionary events,

such as horizontal gene transfer (Wang *et al.*, 2007), gene duplication (Goyal *et al.*, 2006) and gene loss/gain (Town *et al.*, 2006). Various types of genomic rearrangements have also been observed. For example, conserved regions in one species might be shuffled to other positions (transposition) or reversed in orientation in contrast with the other species (inversion) (Chen *et al.*, 2007; Eichler and Sankoff, 2003; Kehrer-Sawatzki and Cooper, 2007).

As the length of sequences increases exponentially with time, early alignment methods based on dynamic programming (Needleman and Wunsch, 1970; Smith and Waterman, 1981) are inefficient. Moreover, a short highly conserved region flanked between relatively longer less conserved regions cannot be aligned well by these methods (Batzoglou *et al.*, 2000). In recent years, numerous methods for large-scale sequence alignment have been proposed. These methods include global alignment algorithms, such as MUMmer (Delcher *et al.*, 1999), Avid (Bray *et al.*, 2003), LAGAN (Brudno *et al.*, 2003a) and GS-Aligner (Shih and Li, 2003), as well as local alignment algorithms, such as BLAST (Altschul *et al.*, 1990), CHAOS (Brudno and Morgenstern, 2002) and BLASTZ (Schwartz *et al.*, 2003). These alignment tools have speeded up the progress in the fields of molecular biology and evolution significantly; however, many of them cannot deal with evolutionary events, e.g. rearrangements.

Most global alignment algorithms can only align conserved regions that have a consistent order, while leaving the regions containing transpositions or inversions unaligned with their respective homologous regions. In contrast, local alignment algorithms can identify highly conserved rearranged segments successfully. However, because the output of discontinuous alignments usually does not give detailed alignments within the breakpoint regions, getting a complete picture of complicated rearrangement events is not a trivial task. Information about such alignments is crucial for understanding the possible mechanisms involved in rearrangement events. Furthermore, understanding those mechanisms could provide information on speciation and gain or loss of biological functions. Thus, it is necessary to develop an alignment algorithm that can align breakpoint regions at the nucleotide level and combine the local alignments for a more global output.

*To whom the correspondence should be addressed.

A number of alignment algorithms have been proposed for aligning sequences that contain rearrangement events, e.g. Shuffle-LAGAN (Brudno *et al.*, 2003b), Mauve (Darling *et al.*, 2004) and Mercator (Dewey, 2007). Shuffle-LAGAN is a pairwise alignment algorithm that combines local and global alignments. Given two genomic sequences, Shuffle-LAGAN generates local alignments between the two sequences and connects the local alignments in a so-called ‘1-monotonic map’. The connections in this map are required to be ordered in only one sequence without putting any restrictions on the other sequence. Then, the maximal consistent sub-segments are derived from the map and aligned by a global alignment algorithm. Shuffle-LAGAN allows the sequence to contain duplications, inversions and transpositions by relaxing the order requirement in the other sequence. However, it does not explore the alignment within breakpoint regions, even though the idea is mentioned by the authors. Mauve is a multiple sequence alignment algorithm that identifies and aligns homologous regions in the presence of rearrangement events by identifying ‘locally collinear blocks’, which are the regions shared by two or more species. Like Shuffle-LAGAN, Mauve does not output detailed alignments near breakpoints. Mercator, on the other hand, is not designed for global alignment. Its goal is to construct orthology maps between multiple whole genomes. We mentioned Mercator because it provides a breakpoint finding procedure as an option to refine the boundaries of the orthology maps. However, the performance of the procedure has not been reported.

In this article, we present a pairwise global sequence alignment algorithm called GR-Aligner (Genomic Rearrangement Aligner), which aligns two genomic sequences containing rearrangement events. GR-Aligner estimates the locations of possible breakpoints that render the alignment of sequences extended to the breakpoints. Regions with complicated rearrangement events are usually too diverse to be aligned well; hence GR-Aligner only treats simple inversions and simple transpositions (See Section 2) instead of all types of rearrangement events owing to an accuracy requirement. The rationale behind breakpoint identification in GR-Aligner is to integrate the forward and reverse alignments of the breakpoint regions and determine the breakpoints by the alignment scores. The resulting sequence alignment adjacent to the breakpoints can be further used to infer the evolutionary mechanisms and biological functionalities of the rearrangements.

2 METHODS

Implementing GR-Aligner involves three steps: (i) identifying and merging highly conserved sequence pairs; (ii) identifying breakpoints in inversions; and (iii) identifying breakpoints in transpositions. First, GR-Aligner searches for all highly conserved sequence pairs, called matches, between two input sequences by using *Bl2seq* (Tatusova and Madden, 1999). For all matches, the first sequences are in the plus strand by default. Here, we define a match as direct (respectively inverse) when the second sequence is in the plus (respectively minus) strand. Two direct or inverse matches that are adjacent in both sequences are merged together repeatedly if the score of the merged match is higher than a default threshold. Second, all inverse matches flanked by two direct matches (called simple inversions) are identified (Fig. 1A) and the breakpoints are then determined (Fig. 1B). In the beginning of the third step, GR-Aligner identifies all simple transpositions, i.e. direct matches that cross only one other direct match (Fig. 1A). All the breakpoints of each transposition are then determined (Fig. 1C). For each simple rearrangement event, GR-Aligner outputs the coordinates of the corresponding breakpoints.

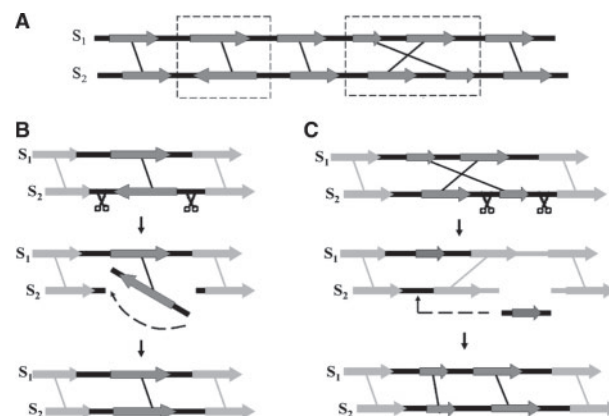


Fig. 1. A graphical interpretation of GR-Aligner’s flowchart: (A) Finding highly conserved sequence pairs, which includes a simple inversion event (the left dashed-line box), and a simple transposition event (the right dashed-line box). (B) Unshuffling a simple inversion (only one possibility is shown). (C) Unshuffling a simple transposition (only one possibility is shown).

We provide an option to view the alignment of sequences that extend to the breakpoints after the possible rearrangements are unshuffled. In the following, we describe each step of the algorithm in detail.

First, we define some notations used in the text. Let S_1 and S_2 be two input genomic sequences whose lengths are denoted by $|S_1|$ and $|S_2|$, respectively. $S_k[i \dots j]$ denotes a subsequence of S_k that starts from the i -th position to the j -th position in S_k . $S_k[i]$ represents the i -th base of S_k . $\overline{S_k}$ denotes the inversion of S_k (e.g. $S_k = 5'$ -AACT-3' and $\overline{S_k} = 5'$ -TCAA-3') and S_k^* denotes the complement of S_k (e.g. $S_k = 5'$ -AACT-3' and $S_k^* = 5'$ -TTGA-3').

2.1 Match identification and merging

In the first step, we use *Bl2seq* to find all highly conserved regions, called matches, in two input sequences S_1 and S_2 . *Bl2seq* is a BLAST-based local alignment tool that utilizes the BLAST algorithm to compare two amino acid or nucleotide sequences (Altschul *et al.*, 1990, 1997). Matches with a $P < 0.001$ and the alignment score higher than a given threshold are selected. To avoid ambiguous matches, only non-overlapping matches are selected and collected as elements for a candidate set Q . For each match $M(i) \in Q$, $M_{\text{start1}}(i)$, $M_{\text{start2}}(i)$, $M_{\text{end1}}(i)$ and $M_{\text{end2}}(i)$ represent the start and end positions of $M(i)$ in S_1 and S_2 , respectively. $M_{\text{score}}(i)$ represents the score of $M(i)$. Note that for an inverse match $M(i)$, $M_{\text{start2}}(i) > M_{\text{end2}}(i)$. GR-Aligner also accepts well-aligned local alignments as input, e.g. pairwise genome alignments from UCSC Genome Bioinformatics Site (<http://genome.ucsc.edu/>).

After match identification, GR-Aligner sorts all the matches by their start1 values, i.e. matches are ranked in increasing order in S_1 . Then, two adjacent direct or inverse matches, $M(i)$ and $M(i+1)$, in Q are merged if the following conditions hold:

- $M(i)$ and $M(i+1)$ are ordered in S_2 ;
- $\text{INTER}_{\text{length}}(M(i), M(i+1)) < \min(\text{Length}(M(i)), \text{Length}(M(i+1)))$;
- $M_{\text{score}}(i) + M_{\text{score}}(i+1) + \text{INTER}_{\text{score}}(M(i), M(i+1)) > \min(M_{\text{score}}(i), M_{\text{score}}(i+1))$;

where $\text{INTER}_{\text{length}}(M(i), M(i+1)) = \max\{|M_{\text{end1}}(i) - M_{\text{start1}}(i+1)|, |M_{\text{end2}}(i) - M_{\text{start2}}(i+1)|\}$, and $\text{INTER}_{\text{score}}(M(i), M(i+1))$ is the alignment score of the sequences between $M(i)$ and $M(i+1)$ aligned by Needleman-Wunsch algorithm. Condition (b) requires that the distance between $M(i)$ and $M(i+1)$ is shorter than the length of $M(i)$ and $M(i+1)$. Condition (c)

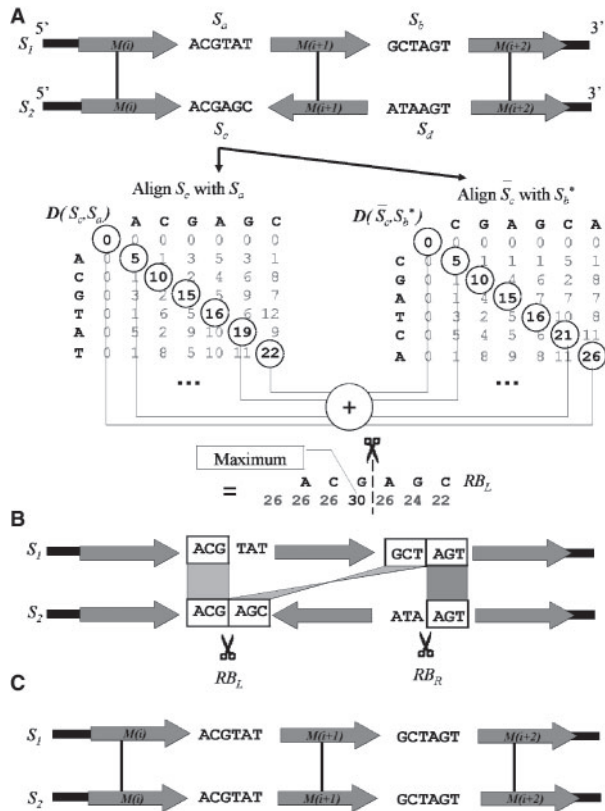


Fig. 2. An example of breakpoint identification in a simple inversion. (A) S_c and \bar{S}_c are aligned with S_a and S_b^* , respectively, by dynamic programming and the alignment scores are recorded in arrays $D(S_c, S_a)$ and $D(\bar{S}_c, S_b^*)$, respectively. The left-hand inversion breakpoint, RB_L , is the position at which the sum of the two alignment scores is maximized. (B) GR-Aligner aligns \bar{S}_b with \bar{S}_d to derive the right inversion breakpoint, RB_R . (C) After reversing and complementing the sequence between RB_L and RB_R in S_2 , inversion unshuffling is implemented for $M(i+1)$.

ensures that, after merging $M(i)$ and $M(i+1)$, the alignment score of the resulting match is not lower than the alignment score of $M(i)$ and $M(i+1)$.

2.2 Inversion and the breakpoint identification

From all the matches sorted in order, it is straightforward to identify a simple inversion event, i.e. an inverse match flanked by two direct matches, and verify that there are no other matches located in or between the three matches (Fig. 1A).

Let $M(i+1)$ be an inverse match between two direct matches $M(i)$ and $M(i+2)$ (Fig. 2A) and the breakpoints reside within the regions (called breakpoint regions) between the inverse and the direct match. We denote the left and right breakpoint regions in S_1 as S_a and S_b , respectively; and those in S_2 as S_c and S_d , respectively (Fig. 2A). Assuming the inversion event occurred in S_2 , GR-Aligner identifies the left and right breakpoints, RB_L and RB_R , as the position in S_c and S_d that optimize the total alignment score. The optimal solution can be inferred by examining all possible positions for RB_L and RB_R that contains $|S_c| \cdot |S_d|$ possibilities, where each alignment takes $O(|S_a + S_b| \cdot |S_c + S_d|)$ time. However, such an exhaustive search is impractical since the run-time is proportional to the fourth power of the length of the breakpoint regions. Thus, we adopt a heuristic approach whose run-time is quadratic to the length of the breakpoint regions.

Our heuristic approach only aligns the sequences between homologous segments once. Then, based on the alignment results, we search for possible breakpoint positions. Homologous segments are the part of sequences extended from one end of a match to a breakpoint. For example, the left and right ends of S_c are orthologous to the left end of S_a and the inverse of the left end of S_b , respectively (Fig. 2A). Specifically, GR-Aligner aligns S_c with S_a and \bar{S}_c with S_b^* with a dynamic programming approach, resulting in two scoring matrices, $D(S_c, S_a)$ and $D(\bar{S}_c, S_b^*)$, respectively. The optimal paths in the scoring matrices are then taken as the optimal alignments. Similarly, S_d is aligned with S_b and \bar{S}_d with S_a^* .

After obtaining the optimal alignments, we can reduce the complexity of exploring all the possible breakpoint positions. GR-Aligner finds the optimal breakpoint in S_c with the maximum score from the two optimal alignments. Let $T_{(S_c, S_a) \rightarrow S_c} = \{a_0, a_1, a_2, \dots, a_{|S_c|}\}$ and $T_{(\bar{S}_c, S_b^*) \rightarrow \bar{S}_c} = \{b_0, b_1, b_2, \dots, b_{|S_c|}\}$ be two 1D arrays that record the scores along the optimal paths to each position of S_c and \bar{S}_c in $D(S_c, S_a)$ and $D(\bar{S}_c, S_b^*)$, respectively. Because the orientations of S_c in the two alignments are opposite, the order of the optimal path in $D(\bar{S}_c, S_b^*)$ should be reversed before recording the scores into $T_{(\bar{S}_c, S_b^*) \rightarrow \bar{S}_c}$. If $ar_{BL} + b_{|S_c| - RB_L} = \max_{0 \leq i \leq |S_c|} (a_i + b_{|S_c| - i})$, we call RB_L the optimal breakpoint in S_c (Fig. 2B). Once RB_L has been determined, the partial sequence of S_b aligned with \bar{S}_c can also be determined; and the remaining sequence of S_b is aligned with the reversed S_d (Fig. 2B). The breakpoint in S_d , RB_R , is identified as the position mapped to the last base of the remaining sequence of S_b aligned with S_d . This procedure for finding breakpoints starts with RB_L and is repeated for RB_R . Then, the case with the higher maximal summed score is selected as the breakpoints in S_2 . With the two identified breakpoints, GR-Aligner reverses and complements the inverted segment to generate an end-to-end global alignment. Since inversion could also occur in S_1 , GR-Aligner repeats the above procedure assuming there is an inversion in S_1 . By default, GR-Aligner outputs the coordinates of breakpoints with the best score and also provides an option to view the results of the other possible cases with lower scores, as well as the alignment of sequences after the inversions are unshuffled.

2.3 Transposition and the breakpoint identification

GR-Aligner only treats simple transpositions, i.e. where a direct match crosses just one other direct match, and the crossed matches are flanked by two direct matches that do not cross any other matches (Fig. 3A). If we use only two sequences without an extra outlier sequence for comparison, we may not be able to identify the sequence in which the transposition had occurred and which of the two segments in a sequence had been moved. Thus, GR-Aligner calculates all the four possible unshuffled alignment results and outputs the breakpoint coordinates of the best score among the four scenarios. Here, we describe the procedure for breakpoint identification where the short segment in sequence 2 was transposed (Fig. 3A). The other three results can be obtained by the same procedure.

Assuming a transposition of the segment $M(i+1)$ in S_2 , GR-Aligner identifies two breakpoints, CB_L and CB_R , in S_e and S_f , respectively, and an insertion breakpoint IB in S_d . Similar to the approach used for identifying breakpoints in an inversion, GR-Aligner first aligns S_e with S_c , as well as \bar{S}_e with \bar{S}_a by dynamic programming and derives two optimal paths from the scoring matrices, $D(S_e, S_c)$ and $D(\bar{S}_e, \bar{S}_a)$. CB_L is the position that maximizes the sum of two alignment scores (Fig. 3A). After obtaining CB_L , GR-Aligner aligns \bar{S}_c with \bar{S}_f as well as S_a with S_d to determine CB_R and IB , respectively (Fig. 3B). This procedure for finding breakpoints starts with CB_L and is repeated for CB_R . Then, the case with the higher alignment score is selected as the breakpoints in S_2 . GR-Aligner cuts the segment $M(i+1)$ between the two identified breakpoints and inserts it into the predicted insertion breakpoint IB (Fig. 3C). The algorithm uses the same procedure to deal with the other three cases of possible transpositions, and then outputs the breakpoint coordinates of the best score among the four scenarios. GR-Aligner also provides, as an option, the alignment of sequences after the transpositions are unshuffled.

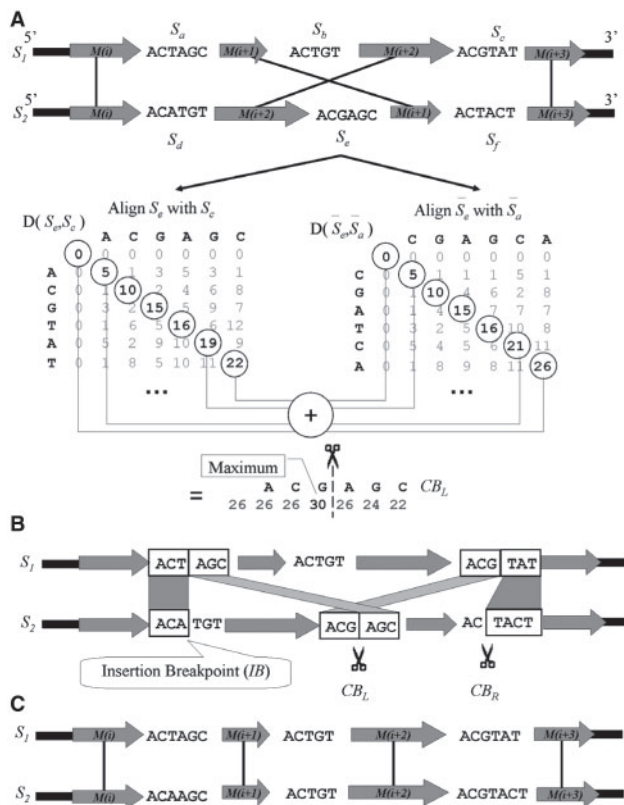


Fig. 3. An example of breakpoint identification in a simple transposition. It is assumed that $M(i+1)$ in sequence S_2 is the transposed match. (A) We align S_e with S_c and align S_d with S_a by dynamic programming and record the alignment scores in arrays $D(S_e, S_c)$ and $D(S_d, S_a)$, respectively. The left transposition breakpoint, CB_L , is the position at which the sum of two alignment scores representing concatenation of matches is maximized. (B) GR-Aligner aligns S_e with S_f to derive the right-hand breakpoint, CB_R , which is identified as the position mapped to the last base of S_c that does not align with S_e . Then, GR-Aligner aligns S_a with S_d to derive the insertion breakpoint, IB, which is identified as the position that aligns to the last base of S_a that does not align with S_e . The transposition is unshuffled by cutting the sequence between CB_L and CB_R and inserting it into the position IB in S_2 . (C) After the transposition unshuffling, the matches, $M(i)$, $M(i+1)$, $M(i+2)$ and $M(i+3)$, are consistent and can be merged.

3 RESULTS

We evaluated the performance of GR-Aligner via simulations on sequence divergences of several species pairs. It was necessary to conduct simulations because we could not test the real-world performance due to the lack of ancestral genomes and original breakpoint positions. In the first part of the results, we used real genomic sequences, which contained small-scale mutations such as insertions, deletions and substitutions. Then, simple rearrangement events were introduced into one of the real sequences, and the rearranged sequences were input to GR-Aligner. Since we knew which sequence contained the simulated rearrangement, we constrained our algorithm to output the result in the correct sequence. The accuracy of the breakpoint identification was then evaluated. In the second part, we applied GR-Aligner to identify breakpoints in simple inversions and transpositions found in the real genomic

sequences of humans and chimpanzees. We compared some of our results with those generated by Shuffle-LAGAN.

3.1 Simulations

Given two genomic sequences, we took the real sequences for the breakpoint simulations. Since our objective was breakpoint identification, we started with well-aligned sequences and then simulated two simple rearrangement events in the sequence pairs. Specifically, we downloaded the pairwise genome alignment files from UCSC Genome Bioinformatics Site (e.g. hg18.panTro2.net). Each file contains the local alignments of two species, and each alignment can be considered as a match in our method. For the simulations, we selected local alignments whose lengths were between 1 kb and 6 kb. Given a local alignment, we introduced a simple inversion or a simple transposition into one of the two sequences. The introduced rearrangement event broke a single match into multiple matches.

Breakpoints were simulated to occur at the base with a lower sequence identity, $R_{\text{breakpoint}}$, than the neighboring regions. For a pair of aligned sequences, the sequence identity at a base was defined as the percentage of identical bases within a window of length 101 bp centered at the base. This definition treats indels and substitutions in the same way. By sliding the window over a local alignment, we obtained the sequence identity profile, from which breakpoints could be set easily. The segment between the two breakpoints was then reversed and complemented in an inversion event or transposed into an insertion breakpoint in the transposition event. After simulations, we applied GA-Aligner to the rearranged sequences.

We tested GR-Aligner with several species' pairs: human–chimpanzee, human–orangutan, mouse–rat and two fly species *Drosophila melanogaster* versus *D.simulans*. In the simulations, we introduced rearrangement events in a fixed species and constrained the algorithm to find breakpoints for that species. After GR-Aligner found the breakpoints, we calculated the successful breakpoint identification rate. A breakpoint prediction is deemed successful when the distance to the answer is $<5\%$ of the length of the breakpoint region. Since there is more than one breakpoint in a simulation, we took the one with the largest relative distance to the answer to represent the prediction. We grouped simulations with similar $R_{\text{breakpoint}}$ scores together in order to calculate success rate, referred to as the accuracy of breakpoint identification.

It is clear from Figure 4 that the accuracy of breakpoint identification declines as the $R_{\text{breakpoint}}$ becomes lower. In general, this trend applies across species pairs. The sequence identity $R_{\text{breakpoint}}$ is thus the prime factor that determines the accuracy of breakpoint identification. We note that, with the same sequence identity, the accuracy for the mouse–rat pair is slightly lower than those for the other pairs. This indicates the involvement of factors other than $R_{\text{breakpoint}}$ in accuracy determination. In addition, the accuracy for transposition is often a little lower than that for inversion.

3.2 Human and chimpanzee genomic sequences

We used GR-Aligner to identify breakpoints in simple inversions and transpositions found in human–chimpanzee alignments. Human and chimpanzee are very close species and their genetic diversity was estimated to be $\approx 1.23\%$ [Mikkelsen *et al.*, 2005 (The Chimpanzee Sequencing and Analysis Consortium)]. Many rearrangement events

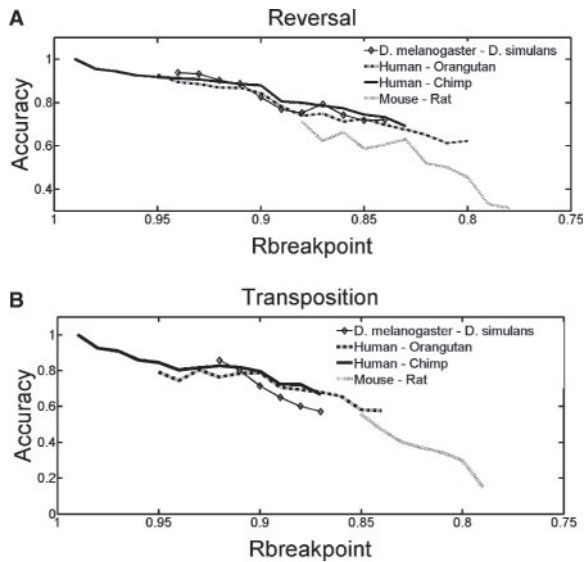


Fig. 4. The accuracy of breakpoint identification versus the sequence identity near the breakpoints, $R_{\text{breakpoint}}$, for four pairs of species.

have been found in the genomes of both species (Chen *et al.*, 2007). The input used for GR-Aligner was the well-aligned pairwise local alignment dataset compiled by UCSC (<http://hgdownload.cse.ucsc.edu/goldenPath/hg18/vsPanTro2/hg18.panTro2.net>). We identified 130 simple inversion events and 846 simple transposition events. The total execution time on an IBM PC with a 2.8 GHz processor and 2 GB RAM was <40 min. The lengths of the breakpoint regions ranged from 0 to 35 kb.

To compare GR-Aligner with Shuffle-LAGAN, we downloaded two homologous regions of human and chimpanzee alignments from UCSC. The first region (Human chr6:93494274-93559185 and Chimp chr6:94060000-94125100) and second region (Human chrX:2563601-2577546 and Chimp chrX:2563040-2577265) contain an inversion and transposition event, respectively (Fig. 5).

The plots in the bottom row of Figure 5A and B are the alignment results derived by Shuffle-LAGAN. Since Shuffle-LAGAN generates different results depending on the order of the two input sequences, we picked one based on the human sequence for output. Each line in the alignment dot-plots represents a set of 'expanded consistent sub-segments' defined by Shuffle-LAGAN. More importantly, one shall only read the end coordinates of the 'expanded consistent sub-segments' in the plots. The lines are not parallel because the sub-segments are usually separated by gaps of different lengths in the two sequences. From the plots, it is clear that, unlike GR-Aligner, Shuffle-LAGAN does not unshuffle inversions. Thus, it is unfair to compare the results of inversions derived by GR-Aligner and Shuffle-LAGAN in the dot-plots. Even so, Shuffle-LAGAN still lumps the transposed element with the adjacent sub-segment in the human-chimpanzee alignment (Fig. 5B). Thus, identifying a transposition in the plots of Shuffle-LAGAN is not straightforward. More importantly, Shuffle-LAGAN does not incorporate a breakpoint identification procedure; hence comparison of the accuracy of breakpoint identification is not possible.

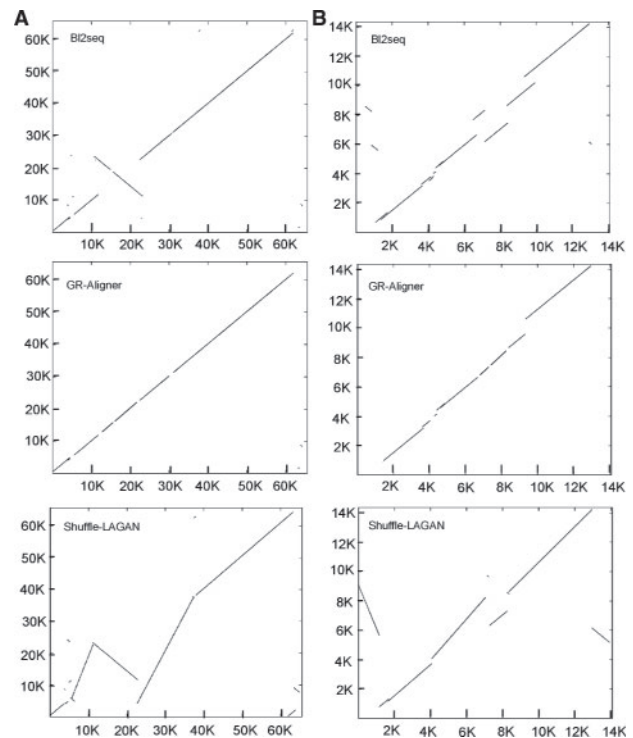


Fig. 5. Comparison of the alignment results derived by GR-Aligner and Shuffle-LAGAN for (A) an inversion event and (B) a transposition event. The top, middle and bottom plots are the dot-plots of the local alignment results by *BL2seq*, GR-Aligner and Shuffle-LAGAN, respectively.

4 DISCUSSION

The goal of GR-Aligner is to identify breakpoints in the rearranged sequences at the nucleotide level. In the present version, GR-Aligner only considers two types of genomic rearrangements, simple inversion and simple transposition. Thus, it cannot treat all types of rearrangement events, such as segmental duplications, inversed transpositions or entangled rearrangements. These tasks, however, are so complicated that none of the algorithms proposed thus far can solve them.

Shuffle-LAGAN, a related tool, tries to map one genome onto the other so that the mapping is optimal. Although the authors consider inversions, transpositions and duplications, their goal differs substantially from our focus. The goal of Mercator, another related program, is to select a subset from the regular homology mappings of multiple genomes of which components have one-to-one sub-relationship of orthology. Although Mercator implements a breakpoint finding procedure, its performance in terms of breakpoint identification has not been reported. Clearly, GR-Aligner, Shuffle-LAGAN and Mercator, have their own advantages for different tasks.

It is possible to extend the scope of our application. For example, an inverted transposition is complicated because we do not know the order in which the event occurred or which homologous segment was moved. We could assume that both scenarios occurred and output all the possible cases. However, multiple rearrangement events probably result in a low accuracy of breakpoint identification

because the sequence identities of the breakpoint regions after unshuffled are usually low.

Using real sequences in our simulations closely reflects the various authentic small-scale mutations during the evolution. In addition, the way we simulate the position of breakpoints is consistent with the observation that the homologous sequences near breakpoints are often lower identical after rearrangements have been unshuffled. Here, we would like to point out an assumption behind our simulations: large-scale genomic rearrangements are independent of small-scale mutations.

Figure 4 shows that the accuracy of breakpoint identification declines more rapidly when the sequence identity is <0.85 . We thus recommend using our tool when the two species are not as divergent as mouse–rat, especially when identifying good breakpoints is essential. In fact, we have tested GR-Aligner for human and mouse alignment, but the accuracy is quite low ($\sim 0.1\%$) due to the low sequence identity between the two sequences (ranging from ~ 0.15 to ~ 0.85). Nevertheless, there are still some successful cases, where the sequence identity is usually high (>0.85). On the other hand, if we simply want a rough idea about the breakpoint positions, we can neglect the accuracy and our algorithm can always output a result. In summary, the performance of our algorithm is determined primarily by the sequence identity near the breakpoints. In addition to the distance between two species, many factors can alter the sequence identity. For example, the nucleotide substitution rate is positively correlated with the recombination rate (Berglund *et al.*, 2009; Duret and Arndt, 2008). Thus, we expect lower accuracy in breakpoint identification near recombination hotspots.

Because it is often unknown in which sequences rearrangement occurred, GR-Aligner outputs the results with the best score. However, we still keep the option of viewing all possible cases. This will be useful when additional information is available, e.g. from a third sequence as an out-group, our aligned sequences can be taken immediately for further exploration.

5 CONCLUSIONS

GR-Aligner is an alignment algorithm that can identify breakpoints in homologous sequences with rearrangement events at the sequence level. In this article, we only deal with some simple rearrangement events. For other complex rearrangement events, we may extend our current algorithm to take account of all possible rearrangements. However, the sequences that are adjacent to these regions under multiple rearrangement events are usually too diverse to align well. Therefore, the calculated breakpoints may be far from the real biological ones. In the future, when more genomes of other close species have been completely sequenced, they can be used as a third party reference to determine which homologous regions have been rearranged compared with an ancestral one. The source code, all sequence data, breakpoint positions and alignment results can be found in http://biocomp.iis.sinica.edu.tw/new/GR_Aligner.htm.

ACKNOWLEDGEMENTS

We wish to thank Dr Feng-Chin Chen and Miss Tiffany Chang for valuable suggestions, as well as the anonymous reviewers for their comments and suggestions, which helped us to improve the quality of our article.

Funding: Academia Sinica (in parts); National Science Council of Taiwan (grant No. 96-2221-E-001-032, in parts).

Conflict of Interest: none declared.

REFERENCES

- Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
- Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
- Batzoglou,S. *et al.* (2000) Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Res.*, **10**, 950–958.
- Berglund,J. *et al.* (2009) Hotspots of biased substitutions in human genes. *PLoS Biol.*, **7**, e26.
- Bray,N. *et al.* (2003) AVID: a global alignment program. *Genome Res.*, **13**, 97–102.
- Budno,M. and Morgenstern,B. (2002) Fast and sensitive alignment of large genomic sequences. *Proc. IEEE Comput. Soc. Bioinform. Conf.*, **1**, 138–147.
- Budno,M. *et al.* (2003a) LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, **13**, 721–731.
- Budno,M. *et al.* (2003b) Global alignment: finding rearrangements during alignment. *Bioinformatics*, **19**(Suppl. 1), i54–i62.
- Chen,F.C. *et al.* (2007) Human-specific insertions and deletions inferred from mammalian genome sequences. *Genome Res.*, **17**, 16–22.
- Darling,A.C. *et al.* (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Res.*, **14**, 1394–1403.
- Delcher,A.L. *et al.* (1999) Alignment of whole genomes. *Nucleic Acids Res.*, **27**, 2369–2376.
- Dewey,C. (2007) Aligning multiple whole genomes with Mercator and MAVID. In Bergman,N. (ed) *Comparative Genomics*, Vol. 395 of *Methods in Molecular Biology*. Humana Press, Totowa, NJ.
- Duret,L. and Arndt,P.F. (2008) The impact of recombination on nucleotide substitutions in the human genome. *PLoS Genet.*, **4**, e1000071.
- Eichler,E.E. and Sankoff,D. (2003) Structural dynamics of eukaryotic chromosome evolution. *Science*, **301**, 793–797.
- Goyal,K. *et al.* (2006) Multiple gene duplication and rapid evolution in the groEL gene: functional implications. *J. Mol. Evol.*, **63**, 781–787.
- Janecka,J.E. *et al.* (2007) Molecular and genomic data identify the closest living relative of primates. *Science*, **318**, 792–794.
- Kehrer-Sawatzki,H. and Cooper,D.N. (2007) Structural divergence between the human and chimpanzee genomes. *Hum. Genet.*, **120**, 759–778.
- Lyons,E. and Freeling,M. (2008) How to usefully compare homologous plant genes and chromosomes as DNA sequences. *Plant J.*, **53**, 661–673.
- Mikkelsen,T.S. *et al.* (2005) Initial sequence of the chimpanzee genome and comparison with the human genome. *Nature*. **437**, 69–87.
- Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.
- Pei,J. and Grishin,N.V. (2007) PROMALS: towards accurate multiple sequence alignments of distantly related proteins. *Bioinformatics*, **23**, 802–808.
- Schwartz,S. *et al.* (2003) Human–mouse alignments with BLASTZ. *Genome Res.*, **13**, 103–107.
- Shih,A.C. and Li,W.H. (2003) GS-Aligner: a novel tool for aligning genomic sequences using bit-level operations. *Mol. Biol. Evol.*, **20**, 1299–1309.
- Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.
- Tatusova,T.A. and Madden,T.L. (1999) BLAST 2 Sequences, a new tool for comparing protein and nucleotide sequences. *FEMS Microbiol. Lett.*, **174**, 247–250.
- Tomomura,M. *et al.* (2007) Structural and functional analysis of the apoptosis-associated tyrosine kinase (AATYK) family. *Neuroscience*, **148**, 510–521.
- Town,C.D. *et al.* (2006) Comparative genomics of Brassica oleracea and Arabidopsis thaliana reveal gene loss, fragmentation, and dispersal after polyploidy. *Plant Cell*, **18**, 1348–1359.
- Wang,Y. *et al.* (2007) Horizontal transfer of genetic determinants for degradation of phenol between the bacteria living in plant and its rhizosphere. *Appl. Microbiol. Biotechnol.*, **77**, 733–739.
- Zimmermann,T. *et al.* (2008) Cloning and characterization of the promoter of Hvgl-2, the human homologue of Drosophila lethal giant larvae (lgl) polarity gene. *Biochem. Biophys. Res. Commun.*, **366**, 1067–1073.