Sequence analysis

GR-Aligner: an algorithm for aligning pairwise genomic sequences containing rearrangement events

Te-Chin Chu1,2, Tsunglin Liu3, D. T. Lee1,4,5, Greg C. Lee2 and Arthur Chun-Chieh Shih1,4,*

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 homologously 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; Tomonura et al., 2007) and phylogeny analysis (Janecka et al., 2000). 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 MUMiner (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 (Braudno 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 exploit the information 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.

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. $S_i[-j]$ denotes a subsequence of $S_i$ that starts from the $j$th position to the $i$th position in $S_i$. $S_i[j]$ represents the $j$th base of $S_i$. $S_i^-$ denotes the inversion of $S_i$ (e.g. $S_i = 5′$-AACT-3′ and $S_i^- = 5′$-TCAAT-3′) and $S_i^+$ denotes the complement of $S_i$ (e.g. $S_i = 5′$-AACT-3′ and $S_i^+ = 5′$-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 $M(i) \in Q$, $M_{start}(i)$, $M_{end}(i)$, $M_{score}(i)$ and $M_{seq}(i)$ represent the start and end positions of $M(i)$ in $S_1$ and $S_2$, respectively. $M_{seq}(i)$ represents the score of $M(i)$. Note that for an inverse match $M(i), M_{seq}(i) \rightarrow M_{start}(i)$. 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:

(a) $M(i)$ and $M(i+1)$ are ordered in $S_1$;
(b) $\text{INTER}_{start}(M(i), M(i+1)) < \min(\text{Length}(M(i)), \text{Length}(M(i+1))));$
(c) $M_{seq}(i) + \text{INTER}_{end}(M(i), M(i+1)) > \min(M_{seq}(i), M_{seq}(i+1));$

where $\text{INTER}_{start}(M(i), M(i+1)) = \max(\{M_{seq}(i) - M_{start}(i+1), M_{start}(i+1) - M_{seq}(i+1)\}$) and $\text{INTER}_{end}(M(i), M(i+1))$ 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)
The optimal solution can be inferred by examining all possible positions resulted from reversing and complementing the sequence between RB
2, GR-Aligner identifies two breakpoints, CB1 and CB2, in S1 and S2, respectively, and an insertion breakpoint IB in S1. Similar to the approach used for identifying breakpoints in an inversion, GR-Aligner first aligns Sn with S∗n, as well as and S∗n with dynamic programming and derives two optimal paths from the scoring matrices, D(S,S∗), D(S∗,S), and D(S∗,S∗), respectively. Because the orientations of Sn in the two alignments are opposite, the order of the optimal path in D(S,S∗) should be reversed before recording the scores into T(S,S∗). If a∗i + b∗i+1 = max(a∗i−d, a∗i−e, a∗i−f, a∗i−g), we call RB1 the optimal breakpoint in S1 (Fig. 2B). Once RB1 has been determined, the partial sequence of Sn aligned with S∗n can also be determined, and the remaining sequence of Sn is aligned with the reversed S∗n (Fig. 2B). The breakpoint in Sn, RB2, is identified as the position mapped to the last base of the remaining sequence of Sn aligned with S∗n. This procedure for finding breakpoints starts with RB1, and is repeated for RB2. Then, the case with the higher maximal summed score is selected as the breakpoints in Sn. 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 S2, GR-Aligner repeats the above procedure assuming there is an inversion in S2. 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

Assuming a transposition of the segment M(i+1) in S2, GR-Aligner identifies two breakpoints, CB1 and CB2, in S1 and S2, respectively, and an insertion breakpoint IB in S1. Similar to the approach used for identifying breakpoints in an inversion, GR-Aligner first aligns Sn with S∗n, as well as and S∗n with dynamic programming and derives two optimal paths from the scoring matrices, D(S,S∗), D(S∗,S), and D(S∗,S∗), respectively. Because the orientations of Sn in the two alignments are opposite, the order of the optimal path in D(S,S∗) should be reversed before recording the scores into T(S,S∗). If a∗i + b∗i+1 = max(a∗i−d, a∗i−e, a∗i−f, a∗i−g), we call RB1 the optimal breakpoint in S1 (Fig. 2B). Once RB1 has been determined, the partial sequence of Sn aligned with S∗n can also be determined, and the remaining sequence of Sn is aligned with the reversed S∗n (Fig. 2B). The breakpoint in Sn, RB2, is identified as the position mapped to the last base of the remaining sequence of Sn aligned with S∗n. This procedure for finding breakpoints starts with RB1, and is repeated for RB2. Then, the case with the higher maximal summed score is selected as the breakpoints in Sn. 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 S2, GR-Aligner repeats the above procedure assuming there is an inversion in S2. 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.

2.2 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 S1 as Sn and Sn, respectively; and those in S1 as Sn and Sn, respectively (Fig. 2A). Assuming the inversion event occurred in Sn, GR-Aligner identifies the left and right breakpoints, RB1 and RB2, as the position in Sn and Sn that optimize the total alignment score. The optimal solution can be inferred by examining all possible positions for RB1 and RB2 that contains Sn possibilities, where each alignment takes O(|Sn + Sn|) 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 Sn are orthologous to the left end of Sn and the inverse of the left end of Sn, respectively (Fig. 2A). Specifically, GR-Aligner aligns Sn with Sn and Sn with Sn with a dynamic programming approach, resulting in two scoring matrices, D(Sn,Sn) and D(Sn,Sn), respectively. The optimal paths in the scoring matrices are then taken as the optimal alignments. Similarly, Sn is aligned with Sn and Sn with Sn.

After obtaining the optimal alignments, we can reduce the complexity of exploring all the possible breakpoint positions. GR-Aligner finds the optimal alignment in Sn with the maximum score from the two optimal alignments. Let T(Sn,Sn) = {ai,a2,...,ai}, and T(Sn,Sn) = {bi,b1,b2,...,bi} be two 1D arrays that record the scores along the optimal paths to each position of Sn and Sn in D(Sn,Sn) and D(Sn,Sn), respectively. Because the orientations of Sn in the two alignments are opposite, the order of the optimal path in D(Sn,Sn) should be reversed before recording the scores into T(Sn,Sn). If ai+bj = max(ai−d,ai−e,ai−f,ai−g), we call RB1 the optimal breakpoint in Sn (Fig. 2B). Once RB1 has been determined, the partial sequence of Sn aligned with Sn can also be determined, and the remaining sequence of Sn is aligned with the reversed Sn (Fig. 2B). The breakpoint in Sn, RB2, is identified as the position mapped to the last base of the remaining sequence of Sn aligned with Sn. This procedure for finding breakpoints starts with RB1, and is repeated for RB2. Then, the case with the higher maximal summed score is selected as the breakpoints in Sn. 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 Sn, GR-Aligner repeats the above procedure assuming there is an inversion in Sn. 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.
The accuracy of the breakpoint identification was then evaluated. We evaluated the performance of GR-Aligner via simulations on MSeS in Se not align with in simple inversions and transpositions found in the real genomic sequences. In the second part, we applied GR-Aligner to identify breakpoints when we knew which sequence contained the simulated rearrangement, we introduced rearrangement events 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. The 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
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.

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. Thus we recommend using our tool when the two species are not as divergent as mouse-rat, especially when identifying good breakpoints is not the main goal. For other complex rearrangement events, we may extend our algorithm to accommodate other recombination hotspots. Numerous insertion and deletions are inferred from mammalian genome sequences. Genome Res., 17, 16–22.


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


