BIOINFORMATICS

Vol. 19 Suppl. 1 2003, pages i54–i62
DOI: 10.1093/bioinformatics/btg1005

Glocal alignment: finding rearrangements during alignment

Michael Brudno[1,†], *Sanket Malde*[1,†], *Alexander Poliakov*[2], *Chuong B. Do*[1], *Olivier Couronne*[2], *Inna Dubchak*[2] *and Serafim Batzoglou*[1,*]

[1]*Department of Computer Science, Stanford University, Stanford, CA 94305 and* [2]*Life Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*

Received on January 6, 2003; accepted on February 20, 2000

ABSTRACT

Motivation: To compare entire genomes from different species, biologists increasingly need alignment methods that are efficient enough to handle long sequences, and accurate enough to correctly align the conserved biological features between distant species. The two main classes of pairwise alignments are *global* alignment, where one string is transformed into the other, and *local* alignment, where all locations of similarity between the two strings are returned. Global alignments are less prone to demonstrating false homology as each letter of one sequence is constrained to being aligned to only one letter of the other. Local alignments, on the other hand, can cope with rearrangements between non-syntenic, orthologous sequences by identifying similar regions in sequences; this, however, comes at the expense of a higher false positive rate due to the inability of local aligners to take into account overall conservation maps.

Results: In this paper we introduce the notion of *glocal* alignment, a combination of global and local methods, where one creates a map that transforms one sequence into the other while allowing for rearrangement events. We present Shuffle-LAGAN, a glocal alignment algorithm that is based on the CHAOS local alignment algorithm and the LAGAN global aligner, and is able to align long genomic sequences. To test Shuffle-LAGAN we split the mouse genome into BAC-sized pieces, and aligned these pieces to the human genome. We demonstrate that Shuffle-LAGAN compares favorably in terms of sensitivity and specificity with standard local and global aligners. From the alignments we conclude that about 9% of human/mouse homology may be attributed to small rearrangements, 63% of which are duplications.

Availability: Our systems, supplemental information, and the alignment of the human and mouse genomes using Shuffle-LAGAN are available at http://lagan.stanford.edu/glocal.

Contact: serafim@cs.stanford.edu

1 INTRODUCTION

The availability of complete vertebrate genomes, such as the human, mouse, and fugu (Lander *et al.*, 2001; Venter *et al.*, 2001; Waterson *et al.*, 2002; Aparicio *et al.*, 2002) has pushed comparative genomics into a new era. Comparing genomic sequences from related species is a fruitful source of biological insight, as functional elements such as genes and regulatory sites tend to exhibit significant sequence similarity across related genomes, whereas regions that are not functional tend to be non-conserved (Dubchak *et al.*, 2000; Göttgens *et al.*, 2002). Performing alignment on a whole genome scale introduces new challenges, as it is no longer feasible to do manual post-processing of the alignments to ensure their correctness.

The availability of multiple genomic sequences also makes it possible to learn more about the mutations that DNA undergoes during evolution, and thus create better alignment programs. In particular, most current programs penalize insertions, deletions and substitutions of individual base pairs between two sequences (we call these events *simple edits*). It is known, however, that DNA also mutates by various *rearrangement* events, such as translocations (a subsegment is removed and inserted in a different location but the same orientation), inversions (a subsegment of DNA is removed from the sequence and then inserted back in the same location but opposite orientation), duplications (a copy of a subsegment is inserted into the sequence, the original subsegment is unchanged), or a combination of the above (See Fig. 1). In this paper we explore local rearrangement events, which we define to be longer than 100 base pairs (bp) and shorter than 100 thousand bp (Kbp).

Of the two main methods for pairwise alignment—

*To whom correspondence should be addressed.
† These authors contributed equally to the work.

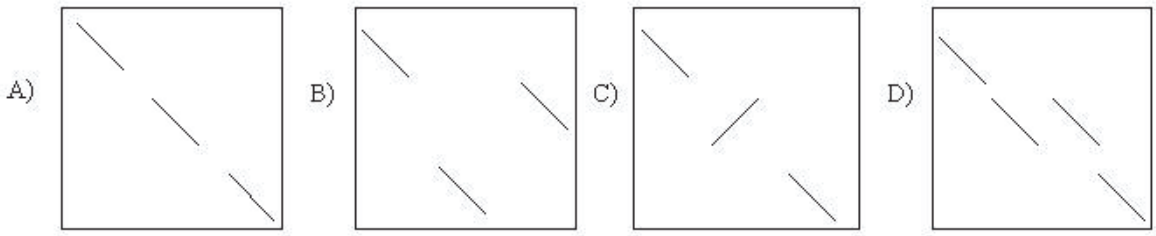

Fig. 1. Common rearrangement events. Top left corner corresponds start of both sequences, diagonal lines to areas of homology. (A) No rearrangements (*consistent homology*). (B) Translocation. (C) Inversion. (D) Duplication.

global alignment, which shows how one sequence can be transformed into another using a combination of the simple edits, and *local* alignment, which identifies local similarities between regions of sequences—neither handles rearrangement events satisfactorily. Global alignment algorithms such as Needleman and Wunsch (1970), Dialign (Morgenstern, 1999), MUMmer (Delcher *et al.*, 1999, 2002), Avid (Bray *et al.*, 2003), and LAGAN (Brudno *et al.*, 2003) do not handle these events at all: the map between the two sequences that a global alignment algorithm creates must be monotonically increasing in both sequences. While local alignment methods such as Smith and Waterman (1981), BLAST (Altschul *et al.*, 1990, 1997), CHAOS (Brudno and Morgenstern, 2002) and BLASTZ (Schwartz *et al.*, 2003) are able to identify homology in the presence of rearrangements between two sequences, they do not suggest how the two sequences could have evolved from their common ancestor. Local alignment algorithms also depend on a cutoff score: only those alignments that score above a threshold are accepted. Deciding on this cutoff can be difficult: if it is set too high, some significant hits will be missed, and if it is set too low there will be too many false positives. Also, in the case where both sequences have n paralogs (copies) of a particular gene or feature, local aligners return n^2 local alignments between all of the pairs, whereas a simple global alignment may more clearly reflect the evolutionary process.

Most work on detecting large scale rearrangements for whole genomes has been done at the level of genes and chromosomes, not genomic sequences. Here each gene is represented by a unique letter, and a string of these letters is a 'chromosome.' Hannenhalli and Pevzner (1995) were the first to present an algorithm to find the minimal number of inversions required to transform one chromosome into another. This method has been extended to other operations, such as translocations, block interchanges and transpositions (Hannenhalli, 1996; Christie, 1996; Bafna and Pevzner, 1998). More recently Pevzner and Tesler (2003) have investigated the rearrangements that have occurred between the human and mouse genome, and found that significant stretches of human/mouse homology are due to micro-rearrangements, of size less than 1 million base pairs (Mbp).

Varré *et al.* (1999) proposed a distance metric between two DNA sequences that models various rearrangement events. Their algorithm, called Tnt1, builds a second sequence from an initially empty string using insertions and copying of blocks from the first sequence. The distance between the two strings is defined to be the Kolmogorov complexity of the program that builds the second sequence. This algorithm has several shortcomings, the most notable being its inability to handle simple edit operations.

An alignment tool for genomic sequences that models rearrangements must meet several criteria, the most important of which is speed: an aligner for genomic sequences should be able to rapidly align sequences of length 1 Mbp or more. Likewise, it is important to use an objective function that correctly penalizes various rearrangement events and simple edits. In this work we introduce the problem of *glocal* alignment, a hybrid of global and local alignments, and specify what features a glocal aligner should have.

We also present Shuffle-LAGAN (SLAGAN), a glocal aligner capable of quickly aligning long genomic sequences. SLAGAN is based on the CHAOS and LAGAN aligners (Brudno and Morgenstern, 2002; Brudno *et al.*, 2003), and has been used to align the mouse genome, split into BAC-sized pieces, to the human genome. We show that SLAGAN, when compared with the regular LAGAN global aligner, has better sensitivity and similar specificity. When compared to the BLASTZ local aligner, SLAGAN shows better specificity at a modest cost in sensitivity. Using the results produced by SLAGAN, we discuss the characteristics of the local rearrangement events that occured since the divergence of the human and mouse genomes.

2 ALGORITHMS

In this section we present a definition of glocal alignment, and present SLAGAN, a glocal alignment algorithm that we have implemented. SLAGAN is more sensitive than regular global aligners, yet it has better specificity than local aligners. Finally SLAGAN is able to rapidly align long genomic sequences of length greater than 1 Mbp. The total time to align the human and mouse genomes using SLAGAN and the technique of Couronne *et al.* (2003) on a modern cluster was about 12 hours of wall clock time (25 CPU-days).

2.1 Glocal alignment

A glocal alignment between two sequences is a series of operations that transform one sequence into the other. We believe the necessary set of operations includes insertions, deletions, point mutations, inversions, translocations and duplications. Each operation incurs a penalty, and the total edit distance between the two sequences is the sum of the penalties. It may also be advantageous to penalize not only individual operations, but also combinations; for instance an inverted translocation should possibly have a smaller penalty than the sum of an inversion penalty and a translocation penalty. Finally, a glocal alignment algorithm should be symmetric in the sequence order: the resulting alignment should not depend on the order in which the sequences are given. One of the main shortcomings of the SLAGAN algorithm presented below and the aforementioned Tnt1 algorithm is that neither is symmetric in the sequence order. This is especially noticeable when one aligns duplicated regions, as both SLAGAN and Tnt1 will currently report duplications in only one of the sequences.

This list of allowed operations is not meant to be complete: as we learn more about the evolution of DNA sequences, additional operations may become necessary. It is also difficult to find good parameters for the various penalties, as currently there is no sound mathematical basis for glocal alignments. Both these areas are promising topics for future work.

2.2 The Shuffle-LAGAN algorithm

The SLAGAN algorithm consists of three distinct stages. During the first stage the local alignments between the two sequences are found using the CHAOS tool (Brudno and Morgenstern, 2002; Brudno *et al.*, 2003). Second, the maximal scoring subset of the local alignments under certain gap penalties is picked to form a *1-monotonic conservation map*. It is the structure of this map that makes SLAGAN different from standard anchored global aligners. Finally, the local alignments in the conservation map that can be part of a common global alignment are joined into *maximal consistent subsegments*, which are aligned using the LAGAN global aligner (Brudno *et al.*, 2003). See Figure 2 for a graphical overview of the algorithm. The exact parameters used in the algorithm are specified in Appendix 1.

2.2.1 Generation of local alignments. To generate local alignments between the two sequences, SLAGAN uses CHAOS, a method that finds small matching words with degeneracy, and chains them into local alignments. Here, we only summarize the CHAOS algorithm. (See Brudno and Morgenstern, 2002; Brudno *et al.*, 2003).

CHAOS works by chaining short words, the *seeds*, which match between the two sequences. Given a word length k, and a degeneracy c, a (k, c)-*seed* is a pair of k-long words ($k-$mers) that match with at most c differences between the two sequences. Given a maximum distance d, and maximum shift s, two seeds that are x- and y-letters apart in the first and second sequences, respectively, can be *chained* together if $x \leqslant d$, $y \leqslant d$, and $|x - y| \leqslant s$. A seed is chained to the single previous seed that creates the highest scoring chain among all chains that end with this seed. CHAOS also supports a translation option, in which both nucleic sequences are translated in all 6 coding frames (3 forward and 3 reverse), and all combinations of frames are compared in turn. Amino acids are grouped (Stanfel, 1996), and all amino acids in the same group are considered equal. Finally, the chains are extended using the standard ungapped BLAST (Altschul *et al.*, 1990) extension, until the score drops below a certain threshold.

After computing the chains, CHAOS scores each chain by a rapid scoring mechanism: ungapped extensions are performed in both directions from each seed, and the optimal location to insert a gap of length exactly $|x - y|$ is found. The resulting alignment is scored using a standard Needleman–Wunsch edit metric with penalties described in Appendix 1. If the alignment is done on amino acid sequences, the rescoring is done using a BLOSUM62 matrix (Henikoff and Henikoff, 1992).

2.2.2 Building the 1-monotonic conservation map. Most tools for rapid global alignment start with a set of local alignments, which they resolve into a 'rough global map' (Delcher *et al.*, 1999; Batzoglou *et al.*, 2000; Bray *et al.*, 2003; Brudno *et al.*, 2003). The rough global map must be non-decreasing in both sequences. In order to allow our alignment algorithm to catch rearrangements, we relax this assumption to allow the map to be non-decreasing in only one sequence, without putting any restrictions on the second sequence. We call this a *1-monotonic conservation map*. Here we show how to create a 1-monotonic conservation map under the affine gap model.

For this portion of the algorithm we represent each local alignment L generated in the previous section as $L = (start_1, end_1, start_2, end_2, score, strand)$, a vector with six fields: the start and end positions of the local

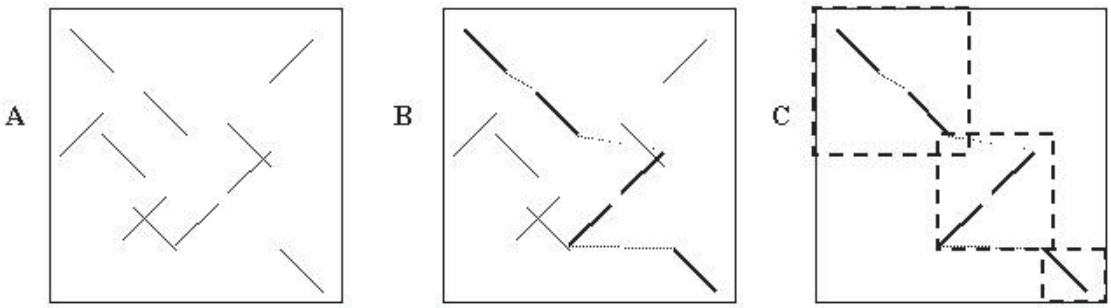

Fig. 2. Overview of the SLAGAN Algorithm. (A) The local alignment between the two sequences are generated using CHAOS. (B) The highest scoring 1-monotonic map (indicated in bold) is found. (C) The maximal consistent subsegments of the 1-monotonic map (dashed boxes) are aligned using LAGAN.

alignment in the two sequences, the score of the alignment, and the strand on which the alignment occurs in the second sequence. We define local alignments on the positive strand to have both start positions less than their respective end positions; on the negative strand, the end position in sequence 2 is smaller than the start position.

Consider two local alignments, L_1 and L_2. We call L_1 and L_2 *1-monotonic* if $L_2.start_1 > L_1.end_1$. We call these alignments *consistent* if (1) they are 1-monotonic, (2) they are both on the same strand, and (3) $L_2.start_2 > L_1.end_2$ for alignments on the positive strand or $L_2.start_2 < L_1.end_2$ for alignments on the negative strand. An ordered list of local alignments $[L_1 \ldots L_k]$ is 1-monotonic or consistent if for any pair of local alignments L_i, L_j if $i < j$ then L_i and L_j are 1-monotonic or consistent, respectively. Intuitively, a list of local alignments is 1-monotonic if it is strictly increasing in the first sequence. The list is consistent if the alignments in it are strictly increasing in both sequences and all on the positive strand, or strictly increasing in the first sequence and strictly decreasing in the second and all are on the negative strand. Equivalently, a list of local alignments is consistent if all of them can be chained into a single global alignment. A consistent, ordered list of local alignments will consequently be referred to as a *consistent chain*, or a *consistent subchain*.

The problem of finding the highest scoring *consistent* subset of local alignments can be solved using standard dynamic programming in time $O(n^2)$, by chaining each alignment L_j to the highest scoring consistent subchain ending in some alignment L_i such that L_i and L_j are consistent. Eppstein and colleagues (Eppstein *et al.*, 1992) have shown how to use the sparse dynamic programming technique to speedup the algorithm to run in time $O(n \log n)$. The key observation is that at any point in the dynamic programming, the set of optimal consistent subchains considered so far partitions the two-dimensional space of ($start_1$, $start_2$) coordinates into 'influence areas' such that any local alignment in a particular influence area must be chained to a particular previous optimal consistent subchain. By considering local alignments in order of increasing $start_1$, we need only perform a one-dimensional search to find the appropriate influence area for chaining each local alignment; more specifically, this may be accomplished in $O(\log n)$ time by storing the active influence areas sorted by diagonal number ($start_1$–$start_2$) in a balanced binary tree (see Eppstein *et al.*, 1992 for details). We generalize this result to finding the highest scoring *1-monotonic* subset of alignments, also under the affine gap model. Here we present an overview of the algorithm (a more detailed description is in our online supplement, http://lagan.stanford.edu/glocal/).

Within the new model, the assumptions that all local alignments in a chain lie on the same strand and must be consistent are relaxed. In particular, when chaining local alignment L_2 to a subchain ending in L_1, L_1 may be on either the positive or negative strand $(+/-)$, L_1 may come either before ($L_1.end_2 < L_2.start_2$) or after ($L_1.end_2 > L_2.start_2$) L_2 in sequence 2 $(+/-)$, and L_2 may be on either the positive or negative strand $(+/-)$. We represent the eight possible combinations of $L_1.strand$, *direction*, and $L_2.strand$ by 3 bits, in that order.

The original Eppstein $O(n \log n)$ algorithm dealt with the special case where an alignment on the positive strand is always joined to a preceding subchain on the same strand $(+ + +)$. In the generalized model, however, an alignment need not have the same strand or direction as the subchain to which it is joined; more precisely, for

each local alignment L_2 on a given strand $L_2.strand$, we must consider joining it to consistent subchains that are either on the positive or negative strand ($L_1.strand$) and that either precede or follow it in sequence 2 (*direction*). Finding the optimal consistent subchain with which to join L_2 in each of the four possible combinations of $L_1.strand$ and *direction* may be done in O(log n) time using the balanced binary trees as in the original Eppstein algorithm. Taking the maximum over the four cases then gives the optimal chaining for L_2. Since the gap penalties used and hence the influence regions depend on $L_2.strand$ as well, a total of eight balanced binary trees (for each combination of $L_1.strand$, *direction*, and $L_2.strand$) must be maintained, of which only the four consistent with each $L_2.strand$ are used in determining the proper chaining for L_2. In a sense, we may regard this as running eight parallel versions of Eppstein's algorithm at once, building a single common chain.

In practice we use only four different gap penalties, as we eliminate the symmetric cases, e.g. if both strands are positives and chained in the positive direction ($+++$), they should be penalized the same way as negative strands chained in the negative direction ($---$). One can think of the four gap penalties as corresponding to the regular gap penalties ($+++$ and $---$), inversions ($++-$ and $--+$), translocated inversions ($+--$ and $-++$) and translocations ($+-+$ and $-+-$). The gap penalty charged for all transitions consists of three parts: (1) the gap open penalty is charged for the *consistent* cases ($+++$ and $---$) if the two segments are on different diagonals ($L_1.end_1$–$L_1.end_2 \neq L_2.start_1$–$L_2.start_2$), and is always charged for the other cases (in these cases it is the inversion, translocation or the inverted translocation penalty); (2) the gap continue penalty is equal to $|(L_1.end_1–L_1.end_2)–(L_2.start_1–L_2.start_2)| \times$ *constant*; and (3) the *distance* between two alignments is defined to be $min(|L_1.end_1–L_2.start_1|, |L_1.end_2–L_2.start_2|)$, and it is also penalized as (*distance* $\times$ *constant*). Finally, in order to not pay both an inversion and a translocated inversion penalty for a simple inversion, we add a 'momentum heuristic': if a majority of the last five elements in the chain have a different strand than the current local alignment, and the case we are considering is a translocated inversion ($+--$ or $-++$), the regular gap penalty is charged instead. This is done in order to charge inversions a smaller penalty than translocated inversions (see Appendix 1 for the exact penalties used).

2.2.3 Aligning consistent subsegments. Recall that two local alignments are considered to be consistent if they can both be a part of a global alignment. Once we have a 1-monotonic conservation map it is straight-forward to generate the maximal consistent subsegments of the map by simply sorting all of the local alignments in the 1-monotonic map by their $start_1$ coordinates, taking the first alignment to be the start of a consistent subsegment, and adding additional local alignments while they are all consistent. As soon as an alignment is found to be inconsistent with the current subsegment, we start a new subsegment.

Because the edges of the local alignment may not be reliable borders of the end of homology (there may be additional conservation which was not strong enough to meet the local alignment criterion) the start and end positions of consistent subsegments are expanded in the first sequence to the borders of the previous and next consistent subsegments, respectively. The borders are also expanded in the second subsequence, in proportion to how much they were expanded in the first sequence. The expanded consistent subsegments can now be aligned using any global alignment algorithm.

Note that because of the expansion, the consistent subsegments will now overlap in the first sequence. In order to get a true glocal alignment between the two sequences, it is now necessary to clip the overlapping alignments so that they do not overlap in the first sequence. This can be done using two linear passes over the alignments, one forward and one backward, to find the optimal point at which to end one alignment and start the other.

3 RESULTS

We tested SLAGAN by aligning the human and mouse genomes using the whole genome alignment technique used in the Berkeley Genome Pipeline (Couronne *et al.*, 2003). In this technique the mouse genome is split up into contigs of 250 Kbp. The potential human orthologs for each contig are found using the BLAT aligner (Kent, 2002). The human sequence is then extended around the BLAT anchor, and aligned to the mouse contig using the tested aligner. If the aligner being used is a global aligner and the BLAT hits fall on both strands, then the aligner is called both with the original mouse contig and a reverse-complemented copy, making it possible to catch inversions. However, a global aligner in this context would not be able to deal with small scale translocations or duplications on the same strand. When using SLAGAN about 40% of all local alignments created by CHAOS are eliminated while creating the 1-monotonic conservation map, indicating that the homology map is 'cleaned up' compared to just having all local alignments. This allows for more sensitive settings when generating the local alignments.

3.1 Quality of SLAGAN alignments

In order to evaluate the quality of SLAGAN alignments we have used the metrics developed to evaluate the various alignment programs for the mouse genome paper

Table 1. Sensitivity and specificity of Shuffle-LAGAN compared to a local aligner (BLASTZ) and a global aligner (LAGAN)

Dataset	BLASTZ	LAGAN	Shuffle-LAGAN
all human v. all mouse	39.6	36.5	38.2
human chr 20 v. all mouse	40.5	41.6	42.5
human chr 20 v. mouse chr 2	37.2	41.4	42.4
% non-orthologous	8.1%	0.5%	0.2%

The first three rows are the total coverage of the human dataset by alignments with the mouse dataset. The last line is the percentage of non-onhologous alignments on human chromosome 20

(Waterson *et al.*, 2002). In particular, we consider the sensitivity of an alignment program to be the percentage of base pairs in the alignment it produces that meet a particular Smith–Waterman scoring threshold. The exact matrices we use ('total' and 'tight') are identical to those used to evaluate the mouse genome aligners (see Appendix 1). To test the specificity of the aligners we applied the technique used to evaluate the specificity of BLASTZ: the coverage of human chromosome 20.

Because human chromosome 20 is considered to be completely ortholgous to mouse chromosome 2, very little difference should be seen between the coverage of human chromosome 20 by mouse chromosome 2 and by the whole mouse genome. The results are summarized in Table 1. The overall conclusion is that while SLAGAN may not be quite as sensitive as BLASTZ on the whole genome scale, it is able to align a larger percentage of the orthologous regions of human chromosome 20 and has higher specificity as shown by the lower percentage of non-orthologous alignments. As compared to regular LAGAN, SLAGAN is more sensitive, especially for aligning genes and nearby areas (see Table 2), which are the areas of the genome that are more likely to undergo duplications in order to evolve new function. SLAGAN appears to be slightly more specific than LAGAN, though the difference may not be statistically significant.

3.2 Analysis of rearrangements

After aligning the whole human and mouse genomes with a glocal aligner, it becomes possible to characterize the extent to which BAC-sized chunks in the human genome have maintained a linear order, and by this whether global alignments are an appropriate method to analyze BAC-sized chunks of the genome. Our results (Table 2) indicate that overall, the homologies found on the whole genome level by global and glocal alignments are very similar. There is, however, a noticeable difference when one considers the alignment of genes and gene-related elements, such as coding regions, UTRs and elements close to genes, where one sees a significant improvement of about 2% in total coverage. This result suggests that as much as 2% of the gene coding regions in the human genome may have evolved by local translocation or duplication since the human/mouse divergence. The evolution of genes by local duplication is supported by anecdotal evidence of co-location of homologous genes such as the globin cluster (Flint *et al.*, 2001).

Once the 1-monotonic map is found, it is possible to classify the various types of rearrangements on it. We generate the highest scoring *consistent* map of all the local alignments above a threshold CHAOS score in order to determine whether the main conservation is on the positive or negative strand. Using this map, we classify any local alignment that lies on the opposite strand as an inversion. We label as a translocation any local alignment that is on the main strand, but not part of the consistent chain, as well as any alignment on the weak strand that could not be part of the main consistent chain had it been on the main strand. Finally, if a particular alignment covers at least 70% of another alignment in the mouse sequence, the lower scoring of the two local alignments is labeled a duplication. Because SLAGAN is not symmetric only the duplications in mouse are found. Thus, the number of duplications may be underestimated by as much as a factor of two. We would also not be able to locate translocations and duplications that have been split between two contigs. Table 3 summarizes the different rearrangement events, their proportion in the human genome, and the level to which they are conserved, both overall and per base pair. It is notable that duplications as a whole score lower per base pair (42.7) than sequences that have undergone other rearrangements (54.4). This could be the result of duplicated sequences being free to mutate to evolve new function, while sequences rearranged but not duplicated being more constrained to their previous function. Of the non-duplicated sequences, simple inversions tend to be the shortest (average length of 122 bp), and have the highest score per base pair (55.3), followed by translocations (188 bp, 46.8).

4 DISCUSSION

Sequence alignment is one of the oldest and most successful applications of computer science to biology. Despite the considerable advances achieved after several decades of research in this area, many important challenges remain. One of these challenges is the development of systems for aligning sequences in the presence of rearrangements. This is becoming increasingly important when automatically comparing whole genomes of related species.

We developed SLAGAN, a system suitable for high-throughput reliable alignment of genomic sequences in the presence of rearrangements. SLAGAN is based on LAGAN, a pairwise aligner that is designed for rapid and reliable alignment of a pair of genomic sequences that may be very similar (e.g. human and chimpanzee), or very

Table 2. Coverage of various features of the human genome by BLASTZ, LAGAN, and Shuffle-LAGAN. The tight matrix is the one from Schwartz *et al.* (2003). Coding, UTR and Upstream500 refer to the corresponding areas of Refseq genes

	BLASTZ		LAGAN		Shuffle-LAGAN	
Coverage	Total	Tight[1]	Total	Tight	Total	Tight
Overall	39.5	5.6	36.49	5.24	38.17	5.45
Coding (CDS)	98.2	92.5	93.18	88.31	95.46	90.34
UTRs	86.1/85.9[2]	39.6/26.0[2]	82.46	28.78	84.35	29.16
Upstream200	85.2	28.3	77.71	26.56	80.48	27.1

[1] These results use chromosome 20 rather than the whole genome as the human sequence, as numbers for the latter were not published. Whole genome numbers tend to be slightly lower. [2] BLASTZ authors did not publish the overall coverage of UTRs, the 5′ UTR/3′UTR coverage is given instead.

Table 3. Classification of local rearrangements found between the human and mouse genomes by Shuffle-LAGAN and LAGAN.

	Main Strand			Inverted			Total		
	% of Length	Avg Length	Score per bp	% of Length	Avg Length	Score per bp	% of Length	Avg Length	Score per bp
Consistent	90.8	143	59.9	1.1	133	55.9	91.8	143	59.9
Translocated only	1.0	150	53.6	1.5	159	54.0	2.5	155	53.8
Duplicated only	2.3	274	44.8	0.2	249	39.4	2.3	272	44.2
Transloc. & Dupl.	1.5	183	44.8	1.9	267	39.2	3.5	222	41.7
Total	95.4	145	59.3	4.7	185	47.6	263Mbp	133	58.1

% **of Length** is the percentage of the total length (in the human sequence) of all local alignments (263 Mbp). **Avg Length** is the length of the average local alignment. **Score per bp** is the CHAOS score divided by length

distant (e.g. human and zebrafish). SLAGAN source code is available at http://lagan.stanford.edu/glocal. The source code for LAGAN is also available from the authors.

The introduction of the concept of glocal alignment suggests several new research directions, of which multiple glocal alignment is perhaps the most natural. The scoring scheme that we use in the SLAGAN algorithm is simplistic, and it is likely that a more sophisticated scheme would result in better alignments. One of the major drawbacks of the SLAGAN algorithm is that it is not symmetric in the sequence order, and thus misses duplications in the sequence that is constrained to be *1-monotonic*. We suspect that chaining with rearrangements may be an NP-complete problem if one does not require the chain to be monotonic in one of the two sequences. It should also be possible to create glocal alignment algorithms using completely different edit models. This work represents a 'first stab' at an open problem, and that more work in this area is necessary.

ACKNOWLEDGEMENTS

M.B. was supported by the NSF Graduate Fellowship. The authors would like to thank Arend Sidow, Gregory Cooper, and Eugene Davydov for useful conversations, and Eric Green for providing the CFTR sequence data that we used to train our parameters.

REFERENCES

Aparicio,S. *et al.* (2002) Whole genome shotgun assembly and analysis of the genome of Fugu rubripes. *Science*, **297**, 1301–1310.

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Altschul,S.F., Madden,T.L., Schoffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Bafna,V. and Pevzner,P. (1998) Sorting by transpositions. *SIAM J. Discrete Math.*, **11**, 224–240.

Batzoglou,S., Pachter,L., Mesirov,J., Berger,B. and Lander,E.S. (2000) Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Res.*, **10**, 950–958.

Bray,N., Dubchak,I. and Pachter,L. (2003) AVID: a global alignment program. *Genome Res.*, **13**, 97–102.

Brudno,M., Do,C.B., Cooper,G.M., Kim,M.F. and Davydov,E. (2003) NISC comparative Sequencing Program, Green,E.D., Sidow,A. and Batzoglou,S. LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, **13**, 721–731.

Brudno,M. and Morgenstern,B. (2002) Fast and sensitive alignment of large genomic sequences. *Proceeding of the IEEE Computer Society Bioinformatics Conference (CSB)*.

Chiaramonte,F., Yap,V.B. and Miller,W. (2002) Scoring pairwise

genomic sequence alignments. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*.

Christie,D.A. (1996) Sorting permutations by block interchanges. *Inf. Process. Lett.*, **60**, 165–169.

Couronne,O., Poliakov,A., Bray,N., Ishkhanov,T., Ryaboy,D., Rubin,E., Pachter,L. and Dubchak,I. (2003) Strategies and tools for whole-genome alignments. *Genome Res.*, **13**, 73–80.

Delcher,A.L., Kasif,S., Fleischman,R., Peterson,J., White,O. and Salzberg,S.L. (1999) Alignment of whole genomes. *Nucleic Acids Res.*, **27**, 2369–2376.

Delcher,A.L., Phillippy,A., Carlton,J. and Salzberg,S.L. (2002) Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.*, **30**, 2478–2483.

Dubchak,I., Brudno,M., Loots,G.G., Pachter,L., Mayor,C., Rubin,E.M. and Frazer,K.A. (2000) Active conservation of noncoding sequences revealed by three–way species comparisons. *Genome Res.*, **10**, 1304–1306.

Eppstein,D., Galil,Z., Giancarlo,R. and Italiano,G.F. (1992) Sparse dynamic programming I: linear cost functions. *JACM*, **39**, 546–567.

Flint,J., Tufarelli,C., Peden,J., Clark,K., Daniels,R.J., Hardison,R., Miller,W., Philipsen,S., Tan-Un,K.C., McMorrow,T. *et al.* (2001) Comparative genome analysis delimits a chromosomal domain and identifies key regulatory elements in the alpha globin cluster. *Hum. Mol. Genet.*, **10**, 371–382.

Göttgens,B., Barton,L.M., Chapman,M.A., Sinclair,A.M., Knudsen,B., Grafham,D., Gilbert,J.G.R., Rogers,J., Bentley,D.R. and Green,A.R. (2002) Transcriptional regulation of the stem cell leukemia gene (SCL)—comparative analysis of five vertebrate SCL loci. *Genome Res.*, **12**, 749–759.

Hannenhalli,S. (1996) Polynomial Algorithm for Computing Translocation Distance between Genomes. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*.

Hannenhalli,S. and Pevzner,P.A. (1995) Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Proceedings of the 27th Annual Symposium on the Theory of Computing (STOC 95)*. Las Vegas, Nevada, pp. 178–189.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA*, **89**, 10 915–10 919.

Kent,W.J. (2002) BLAT–the BLAST-Like Alignment Tool. *Genome Res.*, **12**, 656–664.

Lander,E.S. *et al.* (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

Morgenstern,B. (1999) DIALIGN 2: improvement of the segment–to–segment approach to multiple sequence alignment. *Bioinformatics*, **15**, 211–218.

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.

Pevzner,P. and Tesler,G. (2003) Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. *Genome Res.*, **13**, 37–45.

Schwartz,S., Kent,W.J., Smit,A., Zhang,Z., Baertsch,R., Hardison,R.C., Haussler,D. and Miller,W. (2003) Human-mouse alignments with BLASTZ. *Genome Res.*, **13**, 103–107.

Simon,A., Stone,E.A. and Sidow,A. (2002) Inference of functional regions in proteins by quantification of evolutionary constraints. *PNAS*, **99**, 2912–2917.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Stanfel,L.E. (1996) A new approach to clustering the amino acids. *J. Theor. Biol.*, **183**, 195–205.

Varré,J.S., Delahaye,J.P. and Rivals,E (1999) Transformation distances: a family of dissimilarity measures based on movements of segments. *Bioinformatics*, **15**, 194–202.

Venter,J.C. *et al.* (2001) The sequence of the human genome. *Science*, **291**, 1304–1351.

Waterson,G.A. *et al.* (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature*, **420**, 520–562.

APPENDIX 1: DEFAULT PARAMETERS

The initial local alignments are generated using CHAOS using seeds of size 11 with one degeneracy. The chains are then rescored with matches and mismatches scored using the match matrices from Chiaramonte *et al.* (2002), a gap open penalty of –50 and a gap continuation penalty of –25. All local alignments scoring above 2000 are returned, and are used to create the 1-monotonic conservation map.

When creating the 1-monotonic conservation map, we penalize all inversions with a cost of—1000— $1 \times$ *distance*, and translocations with—2000—$2.5 \times$ *distance*, where the *distance* is the offset between the two alignments, as defined in section 2.2.2. In order to prevent false chaining of two segments that are *consistent* we use a small gap continuation penalty of -0.5, with no gap opening penalties, as they are not meaningful when one is dealing with gaps between alignments, rather than individual base pairs.

Finally for reasons of efficency we have arbitrarily limited the expansion of consistent subsegments to only 25 kilobase pairs at each end. A whole genome alignment pipeline often tries to align non-homologous regions, and because LAGAN runs slowly in such cases it is necessary to limit the input sequence lengths. LAGAN is run with its default parameters. We do not currently perform the final merging of the overlapping alignments. In order to build the 1-monotonic map we use repeat masked (Repeatmasker, Smit and Green, unpublished) sequences, but we use the unmasked sequences for the final LAGAN alignment, allowing us to align conserved repeats without being misled by non-homologous repeats when building the conservation map.

All of the parameters were hand trained on the Cystic Fibrosis Transmembrane Conductance Regulator region (CFTR), representing $\sim$1.8 megabases of human chromosome 7 and its ortholog in mouse (Thomas *et al.* manuscript in preparation, 2003).

The whole genome alignments were run on a 24 node farm of two processor 2.3 GHz Pentium 4 machines. For evaluation we used human genome June 2002 freeze

Total:
Gap open 500, Gap extend 25, threshold 2500

	A	C	G	T
A	90	−100	−50	−100
C	−100	110	−100	−50
G	−50	−100	110	−100
T	−100	−50	−100	90

Tight:
Gap open 2000, Gap extend 50, threshold 3400

	A	C	G	T
A	100	−200	−100	−200
C	−200	100	−200	−100
G	−100	−200	100	−200
T	−200	−100	−200	100

Fig. 3. Matrices and cutoffs.

(http://genome.ucsc.edu/) and mouse genome MGSC v3 freeze (http://www.ncbi.nlm.nih.gov/genome/guide/mouse/index.html). The matrices and cutoffs used to compute coverage are given in Figure 3.