Fast and sensitive alignment of large genomic sequences

Michael Brudno[1] and Burkhard Morgenstern[2]

[1]110 Gates Building, Computer Science Department, Stanford University, Stanford, CA 94305
[2]International Graduate School in Bioinformatics and Genome Research, Universität Bielefeld, Postfach 100131, 33501 Bielefeld, Germany.

Email: brudno@CS.Stanford.EDU, burkhard@TechFak.Uni-Bielefeld.DE

Abstract

Comparative analysis of syntenic genome sequences can be used to identify functional sites such as exons and regulatory elements. Here, the first step is to align two or several evolutionary related sequences and, in recent years, a number of computer programs have been developed for alignment of large genomic sequences. Some of these programs are extremely fast but often time-efficiency is achieved at the expense of sensitivity. One way of combining speed and sensitivity is to use an anchored-alignment approach. In a first step, a fast heuristic identifies a chain of strong sequence similarities that serve as anchor points. In a second step, regions between these anchor points are aligned using a slower but more sensitive method.

We present CHAOS, a novel algorithm for rapid identification of chains of local sequence similarities among large genomic sequences. Similarities identified by CHAOS are used as anchor points to improve the running time of the DIALIGN alignment program. Systematic test runs show that this method can reduce the running time of DIALIGN by more than 93% *while affecting the quality of the resulting alignments by only* 1% .

The source code for CHAOS is available at `http://www.stanford.edu/~brudno/chaos/`
An integrated program package containing CHAOS and DIALIGN is available at `http://bibiserv.techfak.uni-bielefeld.de/dialign/`.

1. Introduction

Cross-species sequence comparison is playing an increasingly important role in genome analysis and annotation. The functional parts of the genome are under selective pressure, and therefore evolve more slowly than non-functional parts, where random mutations can be tolerated without affecting the evolutionary fitnes of the organism. Consequently, islands of local sequence conservation often correspond to functional elements. Comparative sequence analysis has been used for a variety of purposes, e.g. for gene prediction [10, 3, 4, 20, 33, 30, 27] and identification of regulatory elements [15, 17, 23, 11, 12, 23, 13]. One major advantage of these approaches is that they are based on simple measurement of sequence similarity and require little additional information about the elements to be detected. While more traditional statistical methods need large sets of training data to construct species-specific models of genes or regulatory elements, the comparative methods essentially depend on the availability of syntenic sequences at an appropriate evolutionary distance, making them effective for analysis of the newly sequenced genomes, when little training data is available.

If syntenic sequences are to be compared, the most straight-forward approach is to compile a list of sequence similarities identified by a local alignment method such as BLAST [2]; these similarities can then be ranked according to some quality measure, e.g. by their statistical significance [18]. One problem with this method is that a stringent cut-off criterion needs to be applied in order to eliminate spurious similarities, but such a threshold would also exclude many weak but biologically important similarities.

It is well known that in eukaryotes large-scale genome rearrangements are relatively rare events during evolution; therefore the relative order of genes remains conserved across fairly large evolutionary distances. This fact can be used to reduce the noise of false positive similarities in comparative genome analysis. Rather than considering *all* local sequence simliarities above some (arbitrary) threshold value, one can search for *chains* of similarities, i.e. for sets of similarities that respect the relative order within the sequences under study. This combinatorial constraint can considerably reduce the noise of false positive similarities without preventing weak but biologically important similarities from being detected. Consequently, a new challenge in sequence analysis is construction of global alignments of large genomic sequences.

Recently a number of alignment algorithms have been proposed that combine local and global alignment features by returning ordered chains of local similarities. Since these algorithms must be able to deal with large sequences, a trade-off between sensitivity and speed is necessary. Some approaches for genomic alignment use suffix-tree or hashing algorithms to identify pairs of k-mers of a certain minimum length (and, possibly, a maximum number of mismatches) [8, 22, 21]. These approaches are extremely time-efficient but are most effective at aligning sequences from closely related genomes, e.g. from different strains of a bacterium [8]. Other approaches are more sensitive and have been successfully used to compare more distantly related organisms but these methods are far more time-consuming and are currently restricted to sequences of a few hundred kb in length [19, 25, 27]. A third approach is used in the PipMaker [31] set of tools, where a local alignment program implementing a gapped BLAST algorithm (BLASTZ) is used.

Another possible way of combining speed and sensitivity for genomic alignment is to (i) use a fast heuristics to identify a chain of high-scoring sequence similarities that can be used as anchor points for the final alignment, and then (ii) use a more sensitive method to align the regions that are left over between these anchor points. Such an approach has been proposed, for example, by Batzoglou *et al.* [4]. These authors developed a computer program called *GLASS* that aligns genomic sequences based on matching k-mers. A recursive procedure is used where the minimum length for matching k-mers is decreased at every level of the recursion.

Generally, if sequences of length L_1 and L_2, respectively, are to be aligned, a first heuristic procedure would define anchor points $x_0 = 1, x_1, \ldots, x_N = L_1$ in the first sequence and $y_0 = 1, y_1, \ldots, y_N = L_2$ in the second sequence. This way, the search space for the final alignment procedure is reduced to $\sum_{i=1}^{N}(x_i - x_{i-1}) \times (y_i - y_{i-1})$ compared to $L_1 \times L_2$ for the non-anchored procedure. The idea is that strong local sequence similarities identified by fast heuristics can be expected to be part of an optimal alignment for any reasonable objective function. Thus, if these similarities are used as anchor points for the final alignment procedure the resulting alignment would still be optimal or near-optimal with respect to the objective function employed in the second step. This approach is related to divide-and-conquer strategies for sequence alignment [16, 32].

Obviously, the more dense a chain of anchor points $(x_0, y_0), \ldots, (x_N, y_N)$ is, the higher is the reduction of the search space and gain in speed for the final procedure – on the other hand, too many anchor points could overly restrict the search space and result in decreased alignment quality. The main challenge in the anchored-alignment approach is therefore to find a trade-off between speed and alignment quality – to locate anchor points that are as dense as possible while still leading to optimal or near-optimal alignments.

In this paper, we present a fast local alignment tool called *CHAOS* (CHAins Of Scores). For a pair of input sequences, CHAOS returns a chain of lo-

cal sequence alignments that can be used as anchor points to reduce the search space and improve the running time of any sensitive global alignment procedure. Herein, we show how these anchor points can be used to speed up the DIALIGN alignment program [24]. Systematic test runs demonstrate that this way the running time of DIALIGN can be reduced by 1 - 2 orders of magnitude while the quality of the resulting alignments is only minimally affected. Moreover, the relative improvement in speed increases with the length of the input sequences, making our approach particularly effective for alignment of large genomic sequences.

2. Algorithm

The CHAOS algorithm works by chaining together pairs of similar regions, one region from each of the two input DNA sequences; we call such pairs of regions *seeds*. More precisely, a seed is a pair of words of length k with at least n identical base pairs (*bp*). A seed $s^{(1)}$ can be chained to another seed $s^{(2)}$ whenever (i) the indces of $s^{(1)}$ in both sequences are higher than the indices of $s^{(2)}$, and (ii) $s^{(1)}$ and $s^{(2)}$ are "near" each other, with "near" defined by both a distance and a gap criteria as illustrated in Figure 1. The final score of a chain is the total number of matching *bp* in it. The default parameters used by CHAOS are words of legth 7, with no degeneracy, a distance and gap criteria of 20 and 5 bp respectively, and a score cutoff of 25.

The seeds are located using a simplified version of the Aho-Corasick [1] algorithm. A variation on the *trie* data structure [9] which we call a *threaded trie* (T-trie) is used to store the k-mers of one sequence. A trie is a tree for storing strings in which there is one node for every common prefix. A node which corresponds to the word $w_1...w_p$ would have as its parent a node that corresponds to $w_1...w_{p-1}$. A trie that contains all of the k-mers of some string each leaf is at height k, and in each leaf we store all of the locations where this k-mer occurs in the indexed sequence.

A T-trie differs from a regular trie in that a node that corresponds to the string $w_1...w_p$ will also have a *back pointer* to the node which corresponds to $w_2...w_p$. We start by inserting into the T-trie all of the k-mers of one of the sequences, which we will call the *database*. Then we do a "walk" using the other, *query* sequence, where we start by making the root of the T-trie our current node, and for every letter of the query:

1. if the *current* node has a child corresponding to this letter we make this child our current node, and return any seeds stored in it.

2. otherwise make the node pointed to by our *back pointer* our *current* node, and return to step 1.

As an illustration of why this method works well in practice, assume that all of the possible k-mers are present in the database (which is most likely the case). Then, finding the k-mers that correspond to the next letter of the query requires only two pointer operations: the first is to follow a back pointer from the k level node which is our *current* node, the second to follow a down pointer from the resulting node to the appropriate child. To allow degeneracy we permit multiple current nodes, which correspond to the possible degenerate words.

Let D be the maximum distance between two adjacent seeds. The seeds generated while examining the last D base pairs of the query sequence (Figure 1) are stored in a skip list, a probabilistic data structure that allows for fast searches and easy in-order traversal of its elements [29]. The seeds are ordered by the difference of its indices in the two sequences (*diagonal number*). For each seed s found at the *current location* do a search in the skip list for previously stored seeds which have diagonal numbers within the permitted gap criterion of the diagonal number of s. We thus find the possible previous seeds with which s can be chained. The highest scoring chain is picked, and this chain can be further extended by future seeds. In order to enforce the distance criterion we then remove from the skip list all seeds which were generated D base pairs from the position at which the new seeds are, and insert the new seeds into the skip list.

CHAOS can be used as a stand-alone program for local sequence alignment or as a pre-processing step to find anchor points for any global alignment proce-

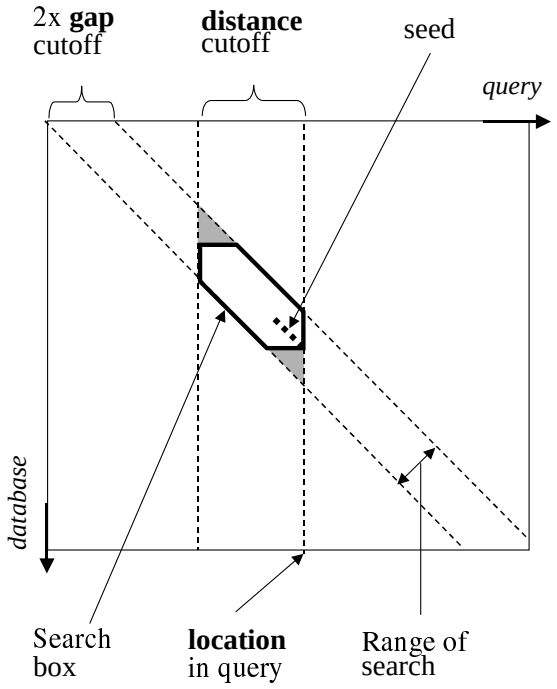

Figure 1: The figure shows a matrix representation of sequence alignment. The seed shown can be chained to any seed which lies inside the *search box*. All seeds located less then *distance bp* from the *current location* are stored in a skip-list, in which we do a range query for seeds located within a *gap* cutoff from the diagonal on which the current seed is located. The seeds located in the grey areas cannot be chained to in order to make the algorithm independent of sequence order.

dure. In this case, once all of the local alignments are identified, we use an algorithm based on the longest increasing subsequence problem [14] to find the highest scoring montonically increasing subsequence of alignments. In the present study we used such chains of local alignments identified by CHAOS as anchor points for DIALIGN.

3. Results

To test our method, we used a set of 42 sequence pairs form human and mouse as compiled by Jareborg *et al.* [17], they vary in length between less than 6 *kb* and more than 227 *kb*, with an average length of 38 *kb*. These sequences were used in a recent paper for a systematic comparison of five different software tools for genomic alignment [27]; it was shown that DIALIGN was the most *specific* of these methods though it was considerably slower than the other methods. In the present paper, we do not repeat the comparison of these different alignment methods. Instead, we investigate how the running time of DIALIGN is improved by the above described anchoring procedure and how this affects the quality of the output alignments. First, we applied CHAOS to our data in order to obtain chains of anchor points. Next, we aligned the sequence pairs with DIALIGN, first without anchoring and then using the anchor points identified by CHAOS. We measured the running time for aligning our test sequences with and without anchoring and compared the quality of the resulting alignments. DIALIGN was run with the *translation* option where local similarity among DNA sequences is compared at the peptide level, see [26].

When CHAOS is run with default parameters the density of the returned anchor points was, on average, 2.1 anchor points per *kb*; the total CPU time for running CHAOS on all the 42 sequence sets was 94 *s* on a SUN Ultra Enterprise 420 with a 450MHz CPU. The total CPU time for aligning the 42 sequences pairs with the standard version of DIALIGN – i.e. without anchor points – was 179,001 *s*, i.e. more than two days. By contrast, running DIALIGN using the anchor points found by CHAOS with the default cut-off value took only 11,391 *s* of CPU time. Thus,

the combined running time for the anchored alignment procedure – i.e. the time necessary for finding the anchor points plus the running time of the alignment final procedure using these anchor points – was only 6.4 % of the original running time of DIALIGN. If CHAOS is run with decreased cut-off parameters, larger chains of anchor points are found. Thereby the search space for DIALIGN is further decreased and the running time is improved accordingly. On the other hand, this option leads to slightly decreased quality of the final alignments as shown in Table 1.

To compare the quality of the anchored alignments to the non-anchored ones, we used two different measures of alignment quality. First, we considered the *numerical* alignment score that is employed by the DIALIGN program. DIALIGN construcs alignments by assembling pairs of un-gapped segments, so-called *fragments*. Each possible fragment is given a quality score and, for pairwise alignment, the program identifies a chain of fragments with maximum total score; the scores of the fragments are defined based on probabilistic considerations as explained in [24]. We considered the total sum of scores of the alignments produced with and without anchoring option, respectively. For the non-anchored DIALIGN alignments, the sum of scores was 54,214. If CHAOS was applied with the default cut-off in order to anchor the DIALIGN alignment, the total sum of scores was 53,658, i.e. the numerical alignment score was reduced by only 1.02% while the running time was reduced by more than 93%.

Next, we tried to compare the *biological* quality of the returned alignments. The 42 sequence pairs contain a total of 77 known gene pairs and we investigated to what extent the alignments with and without anchoring option were able to identify protein-coding regions. We compared the different alignments at the *nucleotide level*, i.e. a nucleotide that is part of a selected fragment is considered a *true positive* (TP) if it is also part an annotated exon and as *false positives* (FP) if it is not; true and false negatives (TN and FN) are defined accordingly. We used the usual measures for prediction accuracy, namely *sensitivity* $= TP/(TP + FN)$, *specificity* $= TP/(TP + FP)$, and *approximate correlation* $= 0.5((TP/(TP + FN) + (TP/(TP + FP) + (TN/(TN + FP) + (TN/(TN + FN)) - 1$.

The sensitivity of DIALIGN without anchoring option was 83.68%, the specificity was 40.06%, and the approximate correlation was 57.31%. Note that DIALIGN is a general-purpose alignment program and does not attempt to find exact exon boundaries. Therefore, these results roughly reflect the ability of DIALIGN to identify biologically functional regions but they cannot be compared with specialized gene-finding programs. With the new anchoring option, the sensitivity of DIALIGN was 83.44%, the specificity was 40.53% while the approximate correlation was 57.51%. Here, the results for the anchored alignments were even slightly better than for the non-anchored ones though the difference in quality was minimal. By comparison, for CHAOS alone (i.e. without using DIALIGN in the second step), sensitivity was 50%, specificity was 45% and approximate correlation was only 44%.

Finally, we wanted to know how the relative improvement in program running time that we achieved by our anchoring method depends on the length of the input sequences. The main benefit of reduced running time of DIALIGN is that this way the program becomes applicable to genomic sequences that are currently beyond its scope, so we wanted to estimate the behavior of the running time for very long sequences. Let d be the average distance between two anchor points and let L be the maximum sequence length. If the anchors are evenly spaced over the length of the sequences, one has approximately L/d anchor points while the search space between any two adjacent anchor points is d^2. Thus, the the search space for the anchored alignment procedure would be $L/d \times d^2 = L \times d$ compared to L^2 for the non-anchored algorithm.

The running time for our algorithm would therefore grow linearly rather than quadratically with the sequence length, so the relative improvement in running time that is achieved by using anchor points grows with the sequence length. In reality, it is difficult to estimate the distribution of distances between anchor points since this depends, of course, on the sequences being compared. Nevertheless, for our data we could confirm that the improvement in running time was larger for longer sequences (Figure 2).

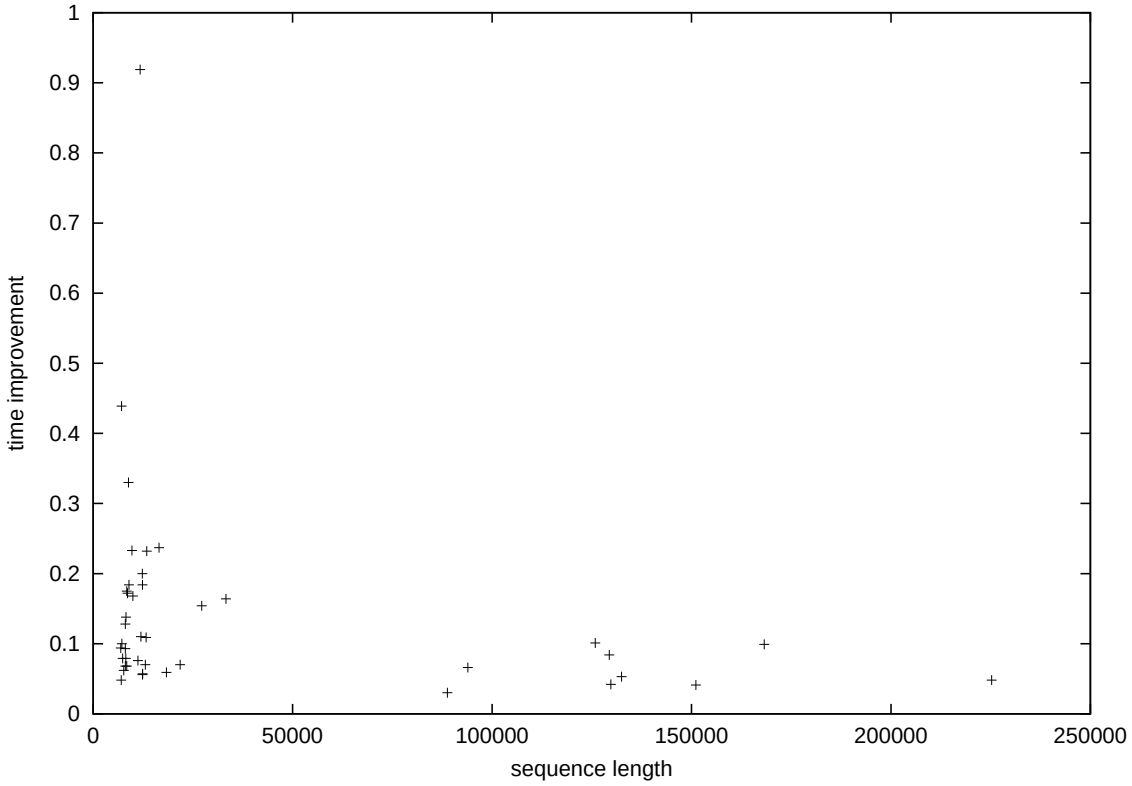

Figure 2: Relative improvement in program running time for 42 pairs of genomic sequences form human and mouse of different length. Each point represents one sequence pair. The x-axis is the medium sequence length of sequence pairs while the y-axis is the relative running time of the anchored alignment procedure compared to the non-anchored procedure.

CHAOS-DIALIGN results

program	cut-off	anc./*kb*	CPU	% CPU	score	% score	Sn	Sp	AC
D			179,001	100.0	54,214	100.0	83	40	57
C+D	35	1.4	14,334	8.0	53,839	99.3	83	40	57
C+D	30	1.7	11,717	6.5	53,820	99.2	83	40	57
C+D	25	2.1	11,485	6.4	53,654	98.9	83	40	57
C+D	20	2.8	8,964	5.0	53,642	98.9	83	40	57
C+D	15	4.2	7,404	4.1	53,208	98.1	82	41	57
C+D	10	6.5	6,696	3.7	52,684	97.1	82	41	57
C	35		92				43	48	42
C	30		92				46	47	43
C	25		94				50	45	44
C	20		93				53	43	44
C	15		93				57	39	43
C	10		94				60	34	42

Table 1: Total CPU time and alignment quality for DIALIGN (D), DIALIGN anchored with CHAOS (C+D) and CHAOS (C) applied to a set of 42 pairs of genomic sequences from human and mouse [17]. CHAOS was run with varying cut-off parameters. Lower cut-off values for CHAOS produced higher numbers of anchor points resulting in a decreased search space for the final DIALIGN alignment procedure thus leading to improved running time but slightly decreased alignment quality. The average number of anchor points per kilo base is shown (anc./*kb*). *Score* is the total *numerical* score of all produced DIALIGN alignments, i.e. the sum of the scores of the segment pairs the alignments consists of. As a rough measure of the *biological* quality of the produced alignments, we compared local sequence similarities identified by DIALIGN and CHAOS to known protein-coding regions. Here, Sn, Sp and AC are *sensitivity*, *specificity* and *approximate correlation*, respectively. For the D and C+D results, DIALIGN was evaluated by comparing *all* segment pairs contained in the alignment to annotated exons.

4. Discussion

CHAOS is a novel heuristic local alignment program for large genomic sequences. Most of the methods for heuristic local alignment, such as BLAST [2] and FASTA [28] were developed when the bulk of available sequence where proteins. It has been shown that such algorithms are not as efficient in aligning non-coding sequences [5]. With the new availability of genomic sequences it is appropriate to refine the algorithms used for local alignment so that they more closely reflect the fashion in which the genomic sequences are conserved. Unlike other fast algorithms for genomic alignment, CHAOS does not depend on long exact matches, does not require extensive ungapped homology, and allows mismatches in seeds, all of which are important when comparing distantly related organisms or non-coding regions, where conservation is generally much poorer than in coding areas.

Some previous algorithms for anchored global alignment have worked by first identifying very strong local similarities among the input sequences and adding weaker similarities later. The problem with this approach is that one high-scoring spurious match can lead to a wrong output alignment where many weaker but biologically important homologies may be missed. By contrast, CHAOS searches for the *highest scoring* chain of local alignments. This way, a numerically high-scoring but biologically wrong local alignment can be conterbalanced by several weaker local alignments if the total score of these alignments exceeds the score of the one wrong alignment.

We demonstrate that the chains of local alignments returned by CHAOS can be used to anchor the DIALIGN alignment procedure, significantly improving the alignment speed, without affecting the quality of the resulting alignments. To compare the quality of the anchored and non-anchored alignments, we compared the *numerical* scores of the resulting alignments as well as their *biological* quality. The numerical scores that we used are the scores for alignment quality as employed by DIALIGN. That is, we measured the sum of the scores of the segment pairs an alignment is composed of. The biological alignment quality was measured by comparing (local) alignments to existing protein-coding exons. This is, admittedly, a very rough measure of alignment quality as non-coding functional elements are completely ignored. However, in the absence of reliable benchmark data for genomic sequence alignment, comparing identified sequence similarities to known exons gives us an effective approximation for assessing the performance of different alignment methods.

Several studies have shown that DIALIGN is highly efficient in identifying coding regions and regulatory elements in genomic sequences [11, 13, 27, 6, 7] and a new gene-prediction program called AGenDA (**A**lignment-based **Gen**e **D**etection **A**lgorithm) has recently been developed that takes DIALIGN alignment as input information [30]. However, DIALIGN used to be far slower than other programs for genomic alignment so its applicability was limited to sequences of a few hundred kilo bases. With the new anchoring option and anchor points created by CHAOS, DIALIGN as well as other slow but sensitive alignment programs can be applied to a much larger range of sequences.

Acknowledgements

M.B. would like to thank Serafim Batzoglou, Inna Dubchak, and Lior Pachter for valuable conversations during this study. The authors also gratefully acknowledge Serafim Batzoglou's comments on the manuscript.

References

[1] A. Aho and M. Corasick. Efficient string matching: an aid to bibliographic search. *Comm. ACM*, 18:333–340, 1975.

[2] S. F. Altschul, W. Gish, W. Miller, E. M. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[3] V. Bafna and D. H. Huson. The conserved exon method for gene finding. In R. Altmann, T. Bailey, P. Bourne, M. Gribskov, T. Lengauer,

I. Shindyalov, L. T. Eyck, and H. Weissig, editors, *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, Menlo Parc, CA, 2000. AAAI Press.

[4] S. Batzoglou, L. Pachter, J. P. Mesirov, B. Berger, and E. S. Lander. Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Research*, 10(7):950–958, 2000.

[5] C. M. Bergman and M. Kreitman. Analysis of conserved noncoding dna in drosophila reveals similar constraints in intergenic and intronic sequences. *Genome Research*, 11:1335–1345, 2001.

[6] M. Blanchette, B. Schwikowski, and M. Tompa. Algorithms for phylogenetic footprinting. *Journal of Computational Biology*, 9(2)211-23, 2002.

[7] M. Blanchette and M. Tompa. Discovery of reguatory elements by a computational method for phylogenetic footprinting. Algorithms for phylogenetic footprinting. *Genome Research*, 12(5):739-48, 2002.

[8] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. *Nucleic Acids Res*, 27(11):2369–2376, 1999.

[9] E. Fredkin. Trie memory. *Comm. ACM*, 3:490–500, 1960.

[10] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition via spliced sequence alignment. *Proc. Natl. Acad. Sci. USA*, 93(17):9061–9066, 1996.

[11] B. Göttgens, L. Barton, J. Gilbert, A. Bench, M. Sanchez, S. Bahn, S. Mistry, D. Grafham, A. McMurray, M. Vaudin, E. Amaya, D. Bentley, and A. Green. Analysis of vertebrate SCL loci identifies conserved enhancers. *Nature Biotechnology*, 18:181–186, 2000.

[12] B. Göttgens, L. Barton, M. Chapman, A. Sinclair, B. Knudsen, D. Grafham, J. Gilbert, J. Rogers, D. Bentley, and A. Green. Transcriptional regulation of the Stem Cell Leukemia gene (SCL) Comparative analysis of five vertebrate SCL loci. *Genome Research*, 12:749-759, 2002.

[13] B. Göttgens, J. Gilbert, L. Barton, D. Grafham, J. Rogers, D. Bentley, and A. Green. Long-range comparison of human and mouse SCL loci: localized regions of sensitivity to restriction endonucleases correspond precisely with peaks of conserved noncoding sequences. *Genome Res.*, 11:87–97, 2001.

[14] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, Cambridge, UK, 1997.

[15] R. Hardison, J. L. Slightom, D. L. Gumucio, M. Goodman, N. Stojanovic, and W. Miller. Locus control regions of mammalian β-globin gene clusters: combining phylogenetic analyses and experimental results to gain functional insights. *Gene*, 205:73–94, 1998.

[16] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18:314–343, 1975.

[17] N. Jareborg, E. Birney, and R. Durbin. Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs. *Genome Research*, 9:815–824, 1999.

[18] S. Karlin and S. F. Altschul. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc. Natl. Acad. Sci. USA*, 90:5873–5877, 1993.

[19] W. J. Kent and A. M. Zahler. Conservation, regulation, synteny, and introns in a large-scale *C. briggsae-C. elegans* genomic alignment. *Genome Research*, 10(8):1115–1125, 2000.

[20] I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, (17):S140–S148, 2001.

[21] S. Kurtz, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. Computation and visualization of degenerate repeats in complete genomes. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, pages 228–238, Menlo Parc, CA, 2000. AAAI Press.

[22] S. Kurtz and C. Schleiermacher. *REPuter*: Fast computation of maximal repeats in complete genomes. *Bioinformatics*, 15(5):426–427, 1999.

[23] G. G. Loots, R. M. Locksley, C. M. Blankespoor, Z. E. Wang, W. Miller, E. M. Rubin, and K. A. Frazer. Identification of a coordinate regulator of interleukins 4, 13, and 5 by cross-species sequence comparisons. *Science*, 288(5463):136–140, 2000.

[24] B. Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15:211–218, 1999.

[25] B. Morgenstern. A simple and space-efficient fragment-chaining algorithm for alignment of DNA and protein sequences. *Applied Mathematics Letters*, 15:11–16, 2002.

[26] B. Morgenstern, A. W. M. Dress, and T. Werner. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. USA*, 93:12098–12103, 1996.

[27] B. Morgenstern, O. Rinner, S. Abdeddaïm, D. Haase, K. Mayer, A. Dress, and H.-W. Mewes. Exon discovery by genomic sequence alignment. *Bioinformatics*, in press.

[28] W. R. Pearson and D. J.Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.

[29] W. Pugh. Skip lists: Skip lists: A probabilistic alternative to balanced trees. *Comm ACM*, 33:668–676, 1990.

[30] O. Rinner and B. Morgenstern. AGenDA: Gene prediction by comparative sequence analysis. *In Silico Biology*, 2:0018, 2002

[31] S. Schwartz, Z. Zhang, K. A. Frazer, A. Smit, C. Riemer, J. Bouck, R. Gibbs, R. Hardison, and W. Miller. PipMaker–a web server for aligning two genomic DNA sequences. *Genome Research*, 10:577–586, 2000.

[32] J. Stoye. Multiple sequence alignment with the divide-and-conquer method. *Gene*, 211:GC45–GC56, 1998.

[33] T. Wiehe, S. Gebauer-Jung, T. Mitchell-Olds, and R. Guigó. SGP-1: Prediction and validation of homologous genes based on sequence alignments. *Genome Research*, 11:1574–1583, 2001.