Algorithms for genome analysis

(Detailed program)

The recent acceleration in genome sequencing projects raised several problems starting an important area of investigation for computer scientists: computational biology. This area presents two main aspects. One is obtaining a formalization of the biological issues at stake. This formalization should enable the issues to be treated by computer programs in a way that is both rigorous and biologically pertinent. The second aspect is developing algorithms that will efficiently solve the problems as formalized, and then encapsulating these algorithms into programs that can be of real use to the computational biology community.

Our goal is to contribute to both the formal and practical parts of this investigative domain. Since the list of problems related to this area is very extensive, in this project we decided initially to concentrate on the following two topics: genome sequencing and analysis. Next, we present some of the problems we intend to investigate.

1 DNA sequencing: fragment assembly

The problem of fragment assembly arises when a DNA molecule is broken into small fragments for sequencing. Once these fragments have been "read", they must be assembled, i.e. put back together, to reconstitute the original molecule. This is an error-prone process at the stages of both sequencing (there may be up to 5% error in base reading with current methods) and assembling.

In essence, the procedure of automated sequencing involves measuring the migration rates of DNA fragments through an electrophoretic gel, where fragments ending in each of the four bases A, C, G and T are labeled with colored dyes so that they can be distinguished [31]. The signal intensities of the migrating fragments are sampled at fixed time intervals as they cross a detector. The result, after signal processing to deconvolve the dyes' signals and remove background noise, is a colored chromatogram, or trace, comprising four superimposed curves, one for each base. The heights of the traces at a given time sample point represent the corresponding signal intensities recorded by the detector. The sampling rate yields typically 10 to 20 points between successive peaks, so although the traces are in fact discrete, they appear continuous to the eye (they therefore look like waveforms). The process of determining the underlying DNA base sequence, essentially from the order of peaks in the trace, is termed base-calling and the sequence obtained is called a read.

Once the fragments have been sequenced, they are assembled. It is this process we are interested in.

At the conceptual level, the problem of assembling DNA sequence fragments naturally divides into three phases [27, 32]. In the overlap phase, each fragment is compared against every other fragment to see if they share a common subsequence, implying that they were potentially sampled from overlapping stretches of the original strand [13, 14, 15]. At this stage, clipping may occur to remove parts of the sequence coming, not from the data one is trying to assemble, but from the vector used for cloning and sequencing the fragmented DNA. One must also consider the presence of errors in the readings, and the fact that the orientation of the fragments relative to the sequence one is trying to reconstruct is unknown. The result of this first phase may be thought of as an overlap graph.

The second, layout phase, takes the overlap graph as input and generates a series of alternate assemblies, or layouts, of the fragments based on the pairwise overlaps therein. A layout specifies the relative locations and orientations of the fragments with respect to each other and is typically visualized as an arrangement of overlapping directed lines, one for each fragment.

In the final, multi-alignment phase, the sequences of all fragments in a layout are simultaneously aligned, giving a consensus sequence as the desired reconstruction of the original DNA strand.

In this project, we wish to address problems coming before the sequencing step itself, as well as between levels two and three of an assembly, or arising after level three, that is, once such an assembly has been performed. In this last case, we shall also use the information given by traces.

All work in this area will be done under the orientation of Gene Myers and therefore rely on his extensive experience in this field of inquiry (he has, among other things concerning fragment assembly, built a prototype software called FAKtory for dealing with this problem: http://www.cs.arizona.edu/faktory/).

Incremental assembly The aim here is to produce tools that can help in assessing the sequencing effort in a genome sequencing project as it is under way. Such a tool could help indicate when sequencing from a particular clone library should be stopped, because new sequences are highly redundant with old ones. The tool should be able to deal with fragments of various sizes and qualities, and flag regions most in need of supporting sequences (weak joins, for example). It should take into account and possibly extend the well-known statistical models developed by Lander and Waterman for sequencing/mapping projects.

Comparing two assemblies The comparison of two assemblies from either the same or from different assembly programs may be an important tool for evaluating and refining the quality of the final sequence. It may be performed at two levels.

At the top level, one should compare the layouts for the two assemblies. One should be able to identify the components of each assembly that are the same, and analyze the difference in the way these common pieces were put together. Some complicating factors may intervene in this analysis, such as the fact that the two assemblers, or two assemblies from a same program, may have clipped the fragments differently. The program should then be also capable of conveying these clipping differences.

At the lower level, one should compare the consensus sequences produced for the common layout components and give some indication of why it is different. One could consider, for instance, simply displaying the offending column of the multi-alignments and its immediate context.

In both cases, it would be convenient therefore if such a comparison were given in some graphical form.

Post-assembly analysis The idea here would be to build tools that look at a given assembly once it has been obtained and summarize it in a number of ways. The obvious, and something already commonly done, is to summarize coverage in both directions (fragments may come from both strands and appear oriented in either direction). Even more interesting would be to do a post-analysis of the occurrence of errors in the fragments with respect to the consensus. This information could then be used to further refine the multi-alignment and to flag suspicious joins. One could also attempt to detect repetitive sequences in the DNA, and especially examine the nature of the solution involving such substrings for suspicious assembly.

Automatic assembly editing based on waveform data analysis Typically, once the data has been assembled, a human being reviews the multi-alignment of the assembly and confirms the consensus in any regions where this alignment looks suspicious or where the coverage is low. This review consists of the human being simultaneously viewing the waveform trace information that lead to the base readings at the start of the sequencing process. To a human being the correct decision is obvious 95% of the time. The question we wish to address here is whether we can build software that is capable of recognizing the "obvious" cases, thus economizing on the time of the user. This problem would require one to become familiar with the waveform data, and to do feature recognition on it. The real challenge however is to correctly define/capture/discover what features a human user is utilizing. This problem is a specialization of the more general one of comparing traces to each other directly [31].

2 Genome analysis

2.1 Gene prediction and gene function

Gene prediction Many algorithms and software for gene prediction have been elaborated, particularly as sequencing projects for whole genomes have proliferated in the last few years. One may consult the web page whose URL is http://linkage.rockefeller.edu/wli/gene/right.html for pointers to an extensive bibliography and references on the subject.

Success for such programs has however been mixed, in the case of eukaryotes much more so than for prokaryotes since the gene-finding problem in prokaryotes is less complex, and therefore much easier to solve. In eukaryotes, the problem is compounded by introns that interrupt the coding regions of genes, so one has to distinguish between the introns and the exons in a predicted gene. This problem is thus generally divided into two tasks: finding candidate exons in a long DNA sequence and then selecting a subset of these candidates to form the predicted gene. This second task is called exon assembly [19]. Methods to identify candidate exons on the other hand usually produce a large number of overlapping candidates, but also provide a score that reflects the "likelihood" that each identified candidate is a true exon. Exon-finding methods often have the most difficulty identifying short exons as well as precisely locating exon-intron boundaries (called splice sites).

It has been estimated that, in the case of eukaryotes, current programs are able to predict genes correctly 80 to 90% of the time only when given a clean set of data to analyze, and just 60% of the time in general. Moreover, the quality of the model obtained (the exons assembled into a gene) is poor. In particular, nearly 100% of the predicted gene structures have incorrect boundaries (whether exon-intron or of 5' and 3' transcribed but untranslated gene flanking regions) [6]. None of the existing programs is capable of dealing with alternative splicing (except GenScan [5] in some limited way).

For prokaryotes, programs for predicting gene have a better accuracy, but are unable to deal with the case of overlapping genes.

In both cases, no algorithm so far has been really good at detecting signals (promoters, splicing sites, enhancers) except by analogy to what is already known (through laboratory experiments) of the signals present in other organisms. In general, all the developed methods rely too much on what they can "learn" from examples which do not allow to discover new things.

There are therefore many problems in this area that remain to be solved, of which the following three would be interesting to explore:

- deal with overlapping genes and/or alternative splicing;
- improve the resolution of the boundaries of the different regions in a genome, when these may be biologically defined;
- improve signal recognition.

In a certain sense, the first two problems may depend on a better understanding and resolution of the third. Although the real biological problem is probably quite complex, Guigó [6] suggested exploring the possibilities of predicting gene structure in genomic DNA sequences by relying only on the identification of sequence signals. His suggestion could be adopted by improving algorithms for pattern inference (see corresponding section below). This would require establishing some model — if only rudimentary so long as it is precise and amenable to a combinatorial algorithmic treatment — of how such signals are recognized and processed.

Gene function The exon assembly problem mentioned above could be solved by finding a set of non-overlapping candidate exons of highest score, since each individual candidate is, in general, given one. It has been reported however that this method does not work well, and more sophisticated approaches to the problem have been tried. Many of these make use, among other tools, of the power of sequence comparison and database search [16, 6]. This implies determining the right exon assembly by comparing all possible ones to genes previously indexed in a database (or translated exon assemblies to all known proteins).

Database search is also an important mean of trying to determine the function of new genes, from eukaryotes or prokaryotes, once these have been, hopefully correctly, predicted. For this, one usually uses programs such as BLAST [1] or FASTA [34] that are efficient and sensitive. In all genome sequencing projects however, a certain number of predicted proteins (up to 30% in Yeast for instance) remains "orphans", that is, no similar homolog is found for them in the databases, and so their function cannot be inferred, except through laboratory experiments that are often costly.

One of the big problems in genome analysis concerns therefore predicted proteins (or translated candidate exon assemblies) for which no homolog is found in the databases. In some cases, this may be because there truly are no such homologs that were found before. It may often be the case however that there exist proteins in the databases having common ancestry/functionality with the newly obtained one but that are evolutionarily too distant for the similarity to be detectable by current database scanning programs such as BLAST and FASTA, or even by the more sensitive, and more time and space consuming, ones using the Smith-Waterman dynamic programming algorithm for optimal local alignment of two strings. One of the reasons for this may be that all these methods perform pairwise comparisons only. A multiple comparison would certainly be more sensitive to faint similarities, but it is very costly (the problem is known to be NP-complete). Yet in the case of a set of sequences (as opposed to a whole database), there have been formulations of the problem [36], closely related to the one given for the search for a Steiner consensus string [19], that have led to reasonably efficient algorithms while remaining biologically pertinent.

The idea here would then be to explore the possibility of extending such formulations, or elaborate new ones, to arrive at a local, multiple comparison approach for database scanning.

2.2 Pattern inference

The general problem of pattern inference may be succinctly described as follows. We are given as input a set of N strings over a certain finite alphabet Σ (for instance, Σ may be the alphabet of nucleotides or that of amino acids) together with a definition $\mathcal{D}$ of similarity (for instance, an edit distance) between a factor in one of the strings and an object external to the set of strings (called a pattern or model). This object can be either a word defined over Σ, or a limited regular expression defined over $\mathcal{P}(\Sigma)$, that is, over the set of parts of Σ. The output is all patterns P that have at least a factor F similar to P (according to the definition $\mathcal{D}$) in at least $Q \leq N$ strings of the set (such a factor F is called an occurrence of P). The length of P may be fixed or may vary between a min and a max value.

Typically, in the case where the strings are segments of DNA or protein sequences, the factor may correspond to a site or signal, i.e. an object having a precise biological function. In general, it will represent the location where the molecule may interact with another molecule. Because this function is important for the organism, and because the recognition by the other molecule of the site where the interaction must happen is at least in part a physico-chemical one, the symbols in the site are better conserved than other positions in the molecule, that is, they usually mutate less. In terms of the factors representing these sites, they correspond therefore to a set of similar words under some definition of similarity [36].

There are many known solutions for this problem [11, 19, 36, 38, 39], but many subproblems remain open. We shall be particularly interested in the following six.

The partitioning problem There is often presence of noise in the set of strings, i.e. some strings do not contain any signal at all. However, we do not know how many there are in such case, so we do not know how to set parameter Q in the search except by trial and error. Incidentally, we do not know even the length of the signal.

Equally important, there are cases where the strings contain not one signal but a collection of different ones and it is not only difficult to find all of them, but sometimes impossible to identify even the strongest family of signals (in terms of frequency of appearance and degree of conservation) among all those present in the sequences.

One of the problems related to pattern inference we wish to solve is therefore that of finding a partition of the input sequences so that each class possesses only one family of signal, or none. In terms of both algorithmic efficiency and sensitivity of the result, we wish also to explore whether it is best to effect the partition after all similar sets of factors have been found by some algorithm (knowing most of these factors will correspond to patterns that are just "junk") or to do both simultaneously.

The multiple signal problem Often there is not one signal, that is one site, but a series of p signals in each sequence and the "real" signal is in fact such series of p signals, for $p \geq 2$.

Let us call each signal in a series a subsignal. Furthermore, it has been observed that the distance between two consecutive subsignals may vary only within a restricted interval of values $[x - \delta, x + \delta]$, where δ is small. This distance constraint comes, for instance, from the fact that the subsignals in a signal are recognized simultaneously by the same molecule. Steric reasons force therefore the corresponding sites to be rather precisely located one relatively to the other. Trying to identify all subsignals simultaneously while taking this distance constraint into account may greatly increase the sensitivity of the search. Of course, in algorithmic terms this also increases the complexity of the problem. Indeed, even supposing there are only two classes of sequences: one having the multiple signal and the other that is just noise, this problem presents now various unknowns:

- Q (the number of strings that have a signal) and the lengths of each subsignal as before;
- the number p of subsignals in a signal of interest;
- the interval of values inside which the distance between each pair of consecutive subsignals of a same signal may vary.

Given these unknowns — possibly fixing some of them — the problem we would like to solve is therefore to identify, in a most efficient and sensitive way, all subsets of p patterns $P_1, \ldots, P_p$ (in that order) such that $P_1, \ldots, P_p$ have occurrences (also in that order) in at least Q strings and that, in each string the distance between the occurrences of successives patterns always varies inside the interval of values $[x - \delta, x + \delta]$. As before, we may ask ourselves whether it would be better in terms of complexity to try to solve this problem after we have identified a list of "solitary" patterns P, or as we are searching for such patterns.

The problem of a "ringed Steiner string" There are examples of signals that may seem to be best modeled not by a motif of the "Steiner string" [19] or "Center of gravity" [39] sort, but by a doughnut, i.e. by a Steiner string with a hole in the middle. It has been suggested for instance that the site for the binding of a protein called CRP, that has a role to play in the expression of a certain number of genes, notably in *Escherichia Coli*, is conserved in terms of its sequence of nucleotides, but must not be too conserved otherwise the protein would have no difficulty in always binding to the site (i.e. the recognition would always happen) [4, 7]. This would make it impossible for such protein to play a regulatory role modulated by the varying needs of the organism. In terms of patterns and factors, it means the patterns P of interest will be those which have at least Q factors F such that P and F are similar, but neither identical, nor even very closely related. For instance, if the definition of similarity used were based on an edit distance, this would imply the distance between P and F should be less than e but more than e', where $0 < e' \leq e$. The question is how best to exploit this constraint to make the algorithmic search as efficient as possible.

The problem of introducing a phylogenetic metrics into pattern inference It is our experience that the similarity metrics usually employed for extracting conserved factors from a set of sequences (or from a single one) may not be the most adequate "likeness" measure, in the same way that the "Sum of Pairs" way of calculating the score of a global alignment is in general inferior to the score given by a tree. We wish therefore to explore the idea of a measure of similarity of a pattern P to the set of factors F that are its occurrences. This measure is based on the underlying, and unknown, phylogenetical tree for the sequences from which the patterns are inferred. We shall start by considering a measure based on some compatibility criteria of corresponding columns in a set of similar factors. We will base ourselves for this on work done mainly by Warnow [24]. We wish then to prospect other phylogenetic reconstruction approaches.

The problem of identifying objects resulting from genome rearrangements Sequence comparison and pattern matching or inference have relied on similarity measures that model the most common types of mutations occurring at the level of a nucleic acid sequence or protein: substitutions, insertions or deletions.

Recently, attention has been given to mutations that occur on a scale much larger than a single DNA/RNA or protein sequence. These mutations occur at the chromosome or genome level and are central in the evolution of the genome. With more genome-level molecular data becoming available, larger scale string comparisons may indeed give insights into evolution that are not seen at the single sequence level.

Genome-level mutations of importance include [19]:

Duplications where parts of a genome are duplicated, possibly very far away from the original site, and then modified;

Inversions where a segment of DNA is reversed;

Transpositions where two adjacent segments of DNA exchange places;

Translocations where two chromosomes exchange genetic material at their ends;

Recombinations where a new sequence is obtained by the, in general multiple, crossing of two parent sequences.

We wish in a first phase to be able to identify all objects resulting from such rearrangement events in a genome. This has two aims. One is to make an inventory of such objects, the second is to use them to build phylogenies (see next section).

The problem of sequence/structure relation Structural problems or sequence/structure relations in proteins represent a vast area of research still relatively unexplored and often ill-defined. We wish at first to consider its impact on genome analysis and the problems mentioned above, in particular to start defining the ways in which structural information, directly obtained or inferred, may be exploited to improve sequence comparison and pattern inference.

Subsidiarily, we hope thereby also to get a clearer idea of what sorts of molecular 3D problems may be amenable to an algorithmic, combinatorial treatment that uses the specific skills of both teams.

2.3 Phylogeny: recombination and genome rearrangements

Recombination Almost all phylogenetic programs available at the current time build strictly bifurcating trees where the sequences used as input to the program are placed at the leaves of the tree. This is a completely inadequate approach when dealing with rapidly evolving sequences (such as retroviruses) where the ancestor to the sequences is often still present among the population (i.e. the set) and may give rise to multiple descendants. Furthermore, such populations often undergo recombination which is an additional reason why the assumption of a strictly bifurcating tree topology is likely to be violated.

Indeed, as mentioned, a recombination event produces a new sequence by crossing two parent sequences. The new sequence is formed by starting at the left end of one parent, copying a substring, crossing over to some location in the other parent, copying a substring, crossing back to the first parent, and so on, always moving to the right within a sequence and stopping only when at the right end of one of the parents [26]. This mechanism of crossover leads to many of the more complex events considered by sequence algorithms, including block insertions and deletions, as well as tandem repeats.

A more accurate algorithm for inferring phylogenies in such cases should therefore be able to place sequences at the nodes of the tree and not just the leaves, and should allow for trees that are not strictly binary. An additional problem when trying to infer phylogenies for populations of rapidly evolving sequences is that the data used to draw such inferences present typically a lower level of variation, thereby resulting in fewer characters for the phylogenetic analyses. Although a few algorithms, both distance-based and character-based, have been devised for dealing with such populations [9, 10, 22, 23], none presents completely satisfactory results.

Genome rearrangements Following the identification of objects involved in the rearrangement of genomes, we wish to study the phylogenetic history of such genomes by addressing four of the five sorts of rearrangement events mentioned in the previous section: reversals (i.e. inversion events) and transpositions between genomes, as well as chromosomal translocations and recombinations.

Reversals The problem of the reversal distance between two genomes may be modeled in the following way. Given a permutation π of integers between 1 and n that represent n genes of a genome G, we define the reversal distance between G and another genome G' to be the minimum number of reversals (i, j) (a reversal (i, j) consists in the inversion of elements $i, i+1, \ldots, j$ of a permutation) which transform π into the identity permutation, this last representing the order of genes in G'.

There exist two versions of the problem: a signed and an unsigned one. The signed version considers the orientation of the genes in a genome, the unsigned one does not.

For the signed version, there exist polynomial-time algorithms that determine the reversal distance between two genomes. The best one is from Kaplan, Shamir and Tarjan [25]. It finds such distance in time $O(n^2)$. It would be interesting to find a faster algorithm for such problem.

The unsigned version has been shown to be NP-hard. In this case, it would be interesting to try to obtain faster algorithms than the one elaborated by Hannenhalli and Pevzner [21] which is in $O(2^k n^3 + n^4)$, where k denotes the number of strips of the permutation containing a single element (a strip of a permutation π is an interval $[i, j]$ such that $(i-1, i)$ and $(j, j+1)$ are breakpoints — that is, $|\pi_{i-1} - \pi_i| \neq 1$ and the same is true for π_j and π_{j+1} — and no breakpoint lies between them).

Transpositions For a permutation π, a transposition $\rho(i, j, k)$ (defined for all $1 \leq i \leq j \leq n+1$ and all $1 \leq k \leq n+1$ such that $k \notin [i, j]$) "inserts" an interval $[i, j-1]$ of π between π_{k-1} and π_k, i.e. $\rho(i, j, k)$ corresponds to a permutation:

$$\begin{pmatrix} 1 & \cdots & i-1 & i & i+1 & \cdots & & \cdots & \cdots & j-2 & j-1 & j & \cdots & k-1 & k & \cdots & n \\ 1 & \cdots & i-1 & j & \cdots & k-1 & i & i+1 & \cdots & \cdots & \cdots & j-2 & j-1 & k & \cdots & n \end{pmatrix}.$$

Given permutations π and σ, the transposition distance problem [35] is to find a series of transformations $\rho_1, \rho_2, \ldots, \rho_t$ such that $\pi \cdot \rho_1 \cdot \rho_2 \cdots \rho_t = \sigma$ and t is minimum. We call t the transposition distance between π and σ. Sorting π by transpositions is the problem of finding the transposition distance $d(\pi)$ between π and the identity permutation.

Sorting by transpositions is a somehow harder combinatorial problem than sorting by reversals. In particular, the transposition diameter $D_t(n)$ of the symmetric group is still unknown. Bafna and Pevzner [2] devised a 1.5 performance guarantee algorithm for sorting by transpositions and demonstrated that:

$$\frac{n}{2} \leq D_t(n) \leq \frac{3n}{4}.$$

The questions to be addressed are whether the exact value of $D_t(n)$ may be calculated, and whether an improved performance guarantee algorithm for sorting by transpositions may be elaborated.

Translocations Recent advances in large-scale comparative genetic mapping offer prospects for understanding mammalian genome evolution. The large number of conserved segments in the maps of man and mouse, for instance, suggest that multiple chromosomal rearrangements have occurred since the divergence of lineages leading to humans and mice. Nadeau and Taylor [33] estimated in 1984 that just 179 + 39 chromosomal rearrangements have occurred since this divergence. This estimate has been revised in 1993 on the basis of the latest data [8] but has changed little compared to the one given by Nadeau and Taylor. Their analysis is however non constructive and does not lead to a scenario of chromosomal rearrangements.

The idea here would therefore be to devise an algorithm to analyze chromosomal rearrangements. Recent research in this area, most notably by Hannenhalli [20] and Kececioglu [28], has left many questions open.

Recombinations The problem here is that of finding a series of recombinations, called spanning history [26], that generate precisely the sequences in a given set, starting from two of these sequences, called a protopair. In [26], Kececioglu and Gusfield considered increasingly more general forms of this problem: the case of a single crossover, equal or unequal, pure or with point mutation; the case of a multiple crossover, pure or with point mutation. In all situations, they upper-bound the cost of every recombination.

The questions we wish to address are the open ones they suggest at the conclusion of their paper. These concern mainly whether there exist algorithms that have better bounds for the problem as they formulated it. They make the observation also that their approach is as costly for finding one pair of protopair than for finding all of them. Is there another approach for which it would be faster either to obtain a single such pair than all, or vice-versa?

Finally, they suggest there may be other forms of formulation for the recombination problem and we wish to explore that also.

2.4 RNA structure search and prediction

RNA structure search Few attempts have been made to write general purpose RNA structure searching programs that take into account both sequence and structure information. Anrep

[30] considers a combination of sequence motifs only. Among those that intertwine such motifs with spatial structure, the two most successful ones are either incapable of dealing with complicated correlations between sequence and/or structural elements [17] or may be quite slow [3]. We aim at building a RNA structure search program that is both rich in its descriptive capabilities, flexible and fast.

RNA structure prediction Many RNA secondary structure prediction algorithms have traditionally been based on free-energy minimization of structures. These, however, consider a single sequence and are often not reliable enough. It is generally agreed that comparative methods, by making use of the information provided by multiple sequences, are the most reliable for determining the structure common to a set of related RNA sequences. Most such comparative methods require though that the alignment be known in advance. On the other hand, algorithms that compare sequence and structure simultaneously in order to infer the structure common to a set of (unaligned) sequences either are heuristic [18, 12], or assume that at least one structure from the set is known [29], or are too time-consuming [37]. Most (Gorodkin is the sole exception) do global alignment although only a portion of the structure each molecule adopts may be common to all. The idea here would therefore be to find a model that would allow to develop an efficient algorithm for inferring the substructure(s) common to a set of unaligned sequences.

References

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

[2] V. Bafna and P. Pevzner. Genome rearrangements and sorting by reversals. *SIAM J. of Computing*, 25:272–289, 1995.

[3] B. Billoud, M. Kontic, and A. Viari. Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence databases. *Nucleic Acids Res.*, 24:1395–1403, 1996.

[4] J. L. Botsford and J. G. Harman. Cyclic AMP in prokaryotes. *Microbiological Rev.*, 56:100–122, 1992.

[5] C. Burge. *Identification of genes in human genomic DNA*. PhD thesis, 1997. Ph. D thesis, Stanford University, Stanford, CA.

[6] M. Burset and R. Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34:353–367, 1996.

[7] B. Combrugghe, S. Busby, and H. Buc. Cyclic AMP receptor protein: role in transcription activation. *Science*, 224:831–838, 1984.

[8] N. G. Copeland, N. A. Jenkins, D. J. Gilbert, J. T. Eppig, L. J. Maltals, J. C. Miller, W. F. Dietrich, A. Weaver, S. E. Lincoln R. G. Steen, L. D. Steen, J. H. Nadeau, and E. S. Lander. A genetic linkage map of the mouse: Current applications and future prospects. *Science*, 262:57–65, 1993.

[9] K. A. Crandall. Intraspecific phylogenetics: support for dental transmission of human immunodeficiency virus. *J. of Virology*, 69:2351–2356, 1995.

[10] K. A. Crandall. Multiple interspecies transmissions of human and simian T-cell leukemia/lymphoma virus type I sequences. *Mol. Biol. Evol.*, 13:115–131, 1996.

[11] M. Crochemore and W. Rytter. *Text algorithms*. Oxford University Press, 1994.

[12] S. Eddy. RNA sequence analysis using covariance models. *Nucleics Acids Research*, 22:2079–2088, 1994.

[13] C. E. Ferreira, C. C. Souza, and Y. Wakabayashi. Reconstruction of DNA fragments: a graph model. In *Anais do XVIII Congresso de Matemtica Aplicada e Computacional*, pages 90–94. 1995.

[14] C. E. Ferreira, C. C. Souza, and Y. Wakabayashi. Rearrangement of DNA fragments: a branch-and-cut algorithm. In *Proceedings of the 2nd Workshop on Solving Practical Combinatorial Optimization Problems*. Valparaíso, Chile, November 1996. Submitted.

[15] C. E. Ferreira, C. C. Souza, and Y. Wakabayashi. A polyhedral approach for DNA fragments arrangement. In *Proceedings of the XVI International Symposium on Mathematical Programming*. Lausanne, Suiça, August 1997. Abstract.

[16] J. W. Ficket. Finding genes by computer: the state of the art. *Trends in Genetics*, 12:316–320, 1996.

[17] D. Gautheret, F. Major, and R. J. Cedergren. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. *Comput. Appl. Biosci.*, 6:325–331, 1990.

[18] J. Gorodkin, L. J. Heyer, and G. D. Stormo. Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleics Acids Research*, 25:3724–3732, 1997.

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

[20] S. Hannenhalli and P. Pevzner. Transforming mice into men (polynomial algorithm for genomic distance problem). In *Proc. of the 36 Annual Symposium on Foundations of Computer Science*, pages 581–592. 1995.

[21] S. Hannenhalli and P. Pevzner. To cut... or not to cut: applications of comparative physical maps in molecular evolution. In *Proc. of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 304–313. 1996.

[22] J. Hein. Reconstructing the evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98:185–200, 1990.

[23] J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *J. of Mol. Evol.*, 36:396–405, 1993.

[24] S. K. Kannan and T. Warnow. Inferring evolutionary history from DNA sequences. *SIAM J. of Computing*, 23:713–737, 1994.

[25] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 344–351. ACM Press, 1997.

[26] J. Kececioglu and D. Gusfield. Reconstructing a history of recombinations from a set of sequences. In *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 471–480. 1994.

[27] J. Kececioglu and E. W. Myers. Combinatorial algorithms for dna sequence assembly. *Algorithmica*, 13:7–51, 1995.

[28] J. Kececioglu and R. Ravi. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 604–613. 1995.

[29] H. Lenhof, K. Reinert, and M. Vingron. A polyhedral approach to RNA sequence structure alignment. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB'98)*, pages 234–242. ACM Press, 1998.

[30] G. Mehldau and E. W. Myers. A system for pattern matching applications on biosequences. *Comput. Applic. Biosc.*, 9:299–314, 1993.

[31] R. F. Mott. *BioInformatics*, 1998.

[32] E. W. Myers. Toward simplifying and accurately formulating fragment assembly. *J. of Comput. Biol.*, 2:275–290, 1995.

[33] J. H. Nadeau and B. A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proc. Natl. Acad. Sci.*, 81:814–818, 1984.

[34] W. R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Meth. Enzymol.*, 183:63–98, 1990.

[35] P. A. Pevzner and M. S. Waterman. Open problems in computational molecular biology. 1995. 3rd Israel Symposium on Theory of Computing and Systems Tel Aviv, January, 1995 (invited presentation.

[36] M.-F. Sagot. *Ressemblance lexicale et structurale entre macromolécules - Formalisation et approches combinatoires.* 1996. Thèse de doctorat - Université de Marne-la-Vallée, Noisy le Grand, France.

[37] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45:810–825, 1985.

[38] J. C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology.* PWS Publisher, 1996.

[39] M. S. Waterman. *Introduction to Computational Biology. Maps, Sequences and Genomes.* Chapman and Hall, 1995.