FOREST, a Browser for Huge DNA Sequences

R. Gras gras@irisa.fr
J. Nicolas jnicolas@irisa.fr

IRISA
Campus de Beaulieu 35042 Rennes cedex, France

Abstract

We present a new tool, FOREST, aiming at representing the content of a large nucleic acid sequence (e.g. >100KB) in a suitable form for the biologist. More precisely, FOREST builds all subsequences repeated in a sequence or a set of sequences. It allows not only to look for the location of the various occurrences of a given subsequence but points also to interesting subsequences with respect to a given criterion. This tool is based on two key ideas. The first idea consists to build a suffix-tree representation of a sequence and to associate to each node of this tree a set of synthesized attributes, computed on the set of subsequences under this node. This allows the biologist to "browse" in the sequence with a constant abstract view of what he may expect to find in the section of the tree he is currently investigating. The second idea consists to summarize the distribution of the information with boolean vectors associated to the sequence. These vectors may be easily displayed in form of a linear map of events, as it is done in genetic mapping. Both representations allow various efficient operations on the sequence. They provide a powerful filtering capacity of the data, while reducing the set of elementary filtering operations to a minimum of conceptual operations. This allows the biologist to easily investigate the most prominent features of the lexical structure of its sequences.

1 Introduction: the need for a lexical analyzer

It is well known, at least for the public of this conference, that the amount of available information on the macromolecules is growing at a fast rate. The last release of EMBL (RL44) provides more than 500Kb nucleic sequences, with a total of more than 360 megabases residues sequenced.

However, these numbers do not reflect the new challenge emerging since a couple of years, due to the production of very large sequences of contigs. A web server [1] gives access to public

[1] http ://golgi.harvard.edu/100kb, managed by Keith Robison, from Church and Gilbert laboratory, Harvard University

sequences of length greater than 100k bases. At present, it contains a list of 75 sequences. In the near future, complete chromosomes or complete genomes of simple organisms, as it is already the case for *Haemophilus influenzae* and *Saccharomyces cerevisiae*, are expected. It means that strings of several megabases have to be evaluated.

A lot of useful tools have been developed for the analysis of sequences. Very few of them are able to consider a whole genome and, when it is the case, are computing only an approximated view of the corresponding sequences. We address in this paper the issue of providing an exact "first level" analysis of these very large sequences.

The analysis of large biological sequences requires to manage a difficult tradeoff between the quantitative and qualitative aspects of the corresponding information. On one side, one is faced with a huge amount of information and algorithms working on complete sequences must be practically linear in space and time complexity. On the other side one must take into account the complexity and richness of the content of the biological sequences.

2 The detection of repeats in a large sequence

2.1 Main tools available for the biologist

The primary work of the biologist to study repeated patterns in a sequence, with respect to itself or another one, is to build a dot matrix, i.e. an array indexed by the positions in the sequences, with a dot in each cell where corresponding letters in the sequences are equal or highly similar. Repeats appear as diagonals in the matrix. This method is still used particularly in some standard algorithms of comparison of sequences like FASTA or BLAST [SA90]. Unfortunately, if no heuristics are used, i.e. one looks for exact results, the algorithm is quadratic with respect to the length of the sequence. Furthermore, the resulting patterns are sometimes hard to interpret [KMGL87]. It would be unrealistic (in term of computing time and in term of representation) to expect to analyse 2Mb sequences by this means.

2.2 Algorithms for finding repeats in strings

Finding all repeats in a sequence is a classical problem in combinatorial studies on strings. It may be solved in linear space and time [McC76], [CS85], [Apo85].
The principle to solve the problem in linear space is to build a subword tree, providing an index of all subwords present in the string. In such a tree, each node is associated to a particular repeated substring. Each parent node is a prefix of its sibling nodes and the leaves of the tree are all suffixes of the string (see section 3.1 figure 1). Furthermore, there are no nodes of degree one (with one sibling node). This ensures the tree to contain at most 2n nodes, if n is the length of the sequence.
The first algorithm building a suffix tree has been proposed by Weiner [Wei73]. In order to achieve linear time computation, efficient algorithms are reading the string character by character, adding at each step the subword built at the previous step plus or minus the current character (depending on the direction of reading) in the current tree. In order to be able to find the place of a new subword in the tree without an expensive search in this tree, the algorithm

must maintain an additional structure, a set of *shortcut links* at each node.

Other representations of the set of repeats have been proposed in the literature, in order to further compress the number of nodes and edges necessary to code this set. The main idea is to define an equivalence relation $\mathcal{R}$ on the set of nodes of the suffix tree and to build a deterministic finite automaton with states corresponding to the equivalence classes of the relation and transitions corresponding to edges in the suffix tree. Blumer et al. [AB85] have proposed an algorithm building a directed acyclic word graph (DAWG), where $\mathcal{R}$ is simply the suffix equivalence relation. Lefèvre and Ikeda have proposed [LI93] to build instead a position end-set tree (PESTry).

2.3 Peculiarities of the treatment of biological sequences

Biological sequences have a few characteristics that entail the necessity to devise new variants of the basic string algorithms. First of all, the alphabet is very small for DNA or RNA sequences (size 4). Sequences are also generally splitted into a great number of contigs with gaps of unknown regions between these contigs. Another complexity level is due to the fact that observed repeats may be approximative due to independent mutations on the several copies of a same fragment. Finally, sequences are oriented and both directions are potentially meaningful, so we must take in account reverse repeats.

3 A browser of sequences

We introduce in this section the tool we have developed for the lexical analysis of large sequences. FOREST [2] [JN96] is funded on the search for repeats, i.e. the search for multiple occurrences of subsequences in a given sequence. It produces two types of structures: dictionaries and maps. FOREST is written in C++.

3.1 The dictionary tree

3.1.1 Representation

First FOREST builds suffix trees. Figure 1 shows such a tree for the sequence aagag ($ is a special character delimiting the end of the sequence). The root of the tree represents the whole sequence. Leaves of the tree are each position in the sequence. Each intermediate node represents a set of positions in the sequence, the set of all positions such that all subsequences (words) starting at these positions have and are the only ones sharing a common beginning (prefix). All repeats present in the sequences are present in the tree, with a node for each.

Trees are not a communly used structure in the analysis of biological sequences. So what are the adventages of building a seemingly complicated result?

First of all, it provides *a hierarchical representation of a set of knowledge*. The level zero of this hierarchy (leaves) describes the individual positions in the sequence. Higher levels describe

[2] FOREST is a French acronym for FOuineur de RÉpétitions dans les Séquences Titanesques (browser of repeats in huge sequences)

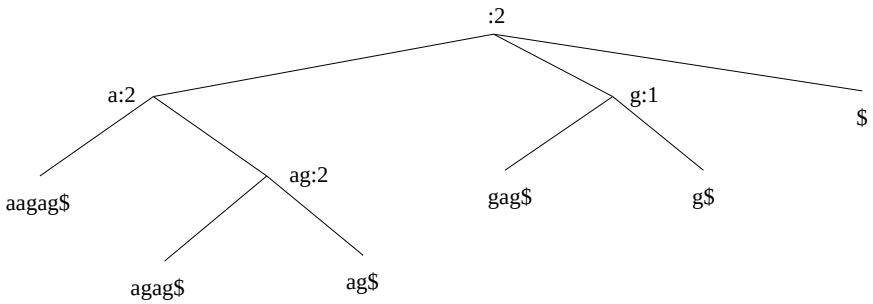

Figure 1: suffix tree for the word aagag$

sets of positions. Each of those sets may be summarized with a property checked on all its members or some statistics on its elements.
For instance, the interest of the biologist may focus on the length of the longest repetition in a given sequence or set of sequences. For the purpose, a "length" attribute is associated to each (interior) node in the tree. The length of a level one node (just above leaves) is the length of the common prefix of its leaves. Attributes are then computed (synthetised) from the leaves to the root of the tree. The value of greater level nodes is the maximum of the lengthes of its sibling nodes. The root gives the length of the longest repeat in the whole sequence.
Thus, higher levels in the hierarchy describe higher more abstract views of the sequence. This way, the biologist may explore the whole tree, starting from the root node or choose the node of a given repeat, keeping at each node the knowledge of what he may expect to find in the rest of the tree.

The second interest of the tree is that it allows a *contextual access to the information*. Each parent node in a tree summarizes a context common to all nodes under it. This context may be generally derived from synthesized attributes. In the opposite direction, it offers a structured filtering capacity, the information appearing at a node being cleared from information already known in parent nodes. This is particularly useful to reveal a fine phenomena that would be otherwise hidden by an already known strongest phenomena or, on the contrary, to help to throw away simple artefacts due to major evidences at a higher level. The context is represented in the tree with herited attributes. We have used them to filter interesting nodes, based on a criterion of "significativity" of the number of occurrences of a word in a sequence.

What is the cost of building such a tree? Our program is funded on the Chen and Sefeiras algorithm [CS85]. Even if the size of the tree is linear with respect to the length of the sequence, a particular attention on its internal representation is necessary. For sequences of N megabases, a tree has an average number of 1.7 10^6 N nodes. If each requires α words to be coded, this leads to an amount of 1.7α N megabytes of memory.

Contigs need a special treatment. Repeats may be localized on different contigs but a given repeat cannot span over two contigs. We have chosen to build a single sequence made of all contigs separated with the special delimiter character $. The construction algorithm is adapted

to handle the $ character. It remains linear in size of the whole sequence (for each new analysed position, one builds at least a new leaf if it corresponds to a repeat never seen before and at most a node and a leaf if it is a new occurence of an old repeat).

3.1.2 Illustration on *E. coli* genome

We have applied FOREST to the study of contigs of *E. coli* and *B. subtilis* [KVD95], [MVHD93], [MGD95]. For E. coli, we have analysed a sequence of more than 3.2 megabases, built up from the concatenation of all available sequences. A biological meaningful analysis requires a great number of exchanges between the computer scientist and the biologist and is not the subject of this paper. Our aim here is just to demonstrate that our tool may be effectively applied to huge sequences.
First, we give a partial view in figure 2 of a graphical representation accessible in FOREST for the whole set of contigs of *Escherichia coli*. Colors are not reproduced in the document. A scale of colors is associated to the scale of values of an attribute (e.g. the number of occurrences) and this allows the reader to quickly "apprehend" the distribution of values in the tree.

Figure 3 shows the view proposed by FOREST after clicking on node AGGAGG in a previous tree : AGGAGG appears at the root of the tree and a (limited in depth) view of the (sub)tree of all repeats in *E. coli* starting with AGGAGG is displayed. It is clear that this representation is not well adapted for a paper output. On a screen, the user may "browse" interactively, that is quickly explore the whole tree and choose at any time a particular node, in order to deepen its knowledge on a specific aspect of the tree. For this purpose he may either request complementary information or choose to further develop this node.

3.2 Biological and logical maps

3.2.1 Distribution of repeats in a sequence

Once potentially important or intriguing words or patterns have been isolated, the next step in the experimental process of discovery is to study the distribution of these words in the sequence.

We use a standard mean for the biologist to represent the location of features in the sequence. As it is the case for genetic or physical maps, linear maps are associated with selected attributes or features computed in the dictionary. If the line represents the sequence, we call it a *biological map*. FOREST can compute and display several superposed maps for various features on the same sequence. E.g., positions of start and stop codons or, if it is known, position of genes may be associated to such maps. This simple mechanism allows to compare and to study the occurrence of various informations along the sequence. For instance, one may compare the positions of a given pattern with respect to known coding sequences. Since the underlying representation of those maps are boolean vectors, various logical operators (and, or, ...) may be easily applied to them in order to emphasize in a new map the intersection, union or difference in the distribution of the corresponding informations.

Figure 4 shows the distribution of Shine-Dalgarno pattern (SD) AGGAGGT in *E. coli* for each reading frame. It also provides the distribution of the palindromic sequence ACCTCCT.

Figure 5 shows a zooming on an interesting area, allowing to compare the position of SD with respect to the beginning of a gene.

Biological maps are just one particular case of a more general way to visualize the links between two kind of data. Instead of considering the distribution of an information in the biological sequence, one may choose another reference space. For example, consider the space of all subwords of length 3, lexicographically ordered. We can display the corresponding map, as it is the case for the sequence. If the information selected is "the number of occurences of subword is low (below a given threshold)", then the map provides the distribution of rare codons.

3.2.2 Study of two words associations

A particularly important case of map comparisons consists to detect close coocurrences of two words in a sequence. However, interesting attributes or features in that case are not associated to word themselves but to their surrounding context. In our previous example, once the SD pattern and the start pattern have been recognized, the most important feature would be to characterize the contexts where a translation occurs. It would allow to better predict the starting positions of unknown genes along the sequence, SD being not sufficient for that purpose.

The very same concepts defined so far, suffix trees and distribution maps, are sufficient to handle the issue with minor adaptations. Instead of considering the map of all positions of a word, one may select the position lying in a given range around the word. Intersecting such a map with the map of the second word determines the context areas in the sequence where both words occur. The set of all context areas may then be studied in the same way contigs have been studied. A suffix tree may be build and the whole process applied iteratively, adding new words or new constraints in the selection of contexts.

3.3 Filtering the information

Although all repeats are potentially relevant in a sequence, a biologist cannot take into account such a huge quantity of information, even it is hierarchically organized. He must focus on a few aspects, and it is generally in terms of extremal features. For instance, interesting words will be repeats with a particularly high number of occurrences or, on the contrary, particulary short words. The next question to be solved becomes then "how can we help to simplify the results of the analysis?".

We have studied several types of selectors to emphasize the most prominent aspects of a pattern with respect to a sequence and/or to determine relevant patterns to be kept while pruning the suffix tree. Necessary filters may be splitted in two main categories.

The first one corresponds to selection on *statistical criteria*. This selection can operates on the value of synthetized attributes in the tree. For instance, one may filter words according to their length. This allow to select only nodes (i.e. set of positions in the sequence) where at least two positions share a common word whose length is greater than a given threshold. In other words, one selects the longest repeats and as a particular case, the root provides the

longest repeat in the sequence. Statistical selection may be funded on much more elaborated criteria, as long as linear computations are possible. However, using only synthetized attributes may be unsufficient in this respect. We have brought in evidence the possible use of herited attributes, while studying the "significativity" of the number of occurrences in a sequence of a given word. The number of occurrences is easily computed as a synthetized attribute. For all leaves of the suffix tree, its value is 1. Then the value for any other node is the sum of values of its siblings. The biologist is most interested by words with unusual number of occurrences (abnormally low or high). But abnormality is a context-sensitive concept. A word may be abnormal either because it is made of subwords abnormal themselves or because of its proper contribution. It is clear that the second alternative is preferable. We therefore estimate for each node the conditionnal probability of the presence of the corresponding word with respect to the observed number of occurences of its parents. This allows to take into account known bias on the distribution of some words (e.g. the distribution of codons) while studying longer pattern including these words. We can see in figure 2 symbols associated to the "significativity" criteria (e.g. ++ for abnormally high number of repeats and – for abnormally low).

The second category of filter corresponds to selection on *symbolic criteria* (lexical or syntaxical). A simple example of lexical filter is the selection of words beginning with ATG, GTG or TTG. This restriction focuses the construction of the tree on areas of the sequence potentially at the beginning of a coding sequence.

We are working on more structural filters. For instance, it would be useful to be able to select the presence of a word at a given distance from another one. But suffix trees are "directed" structures. They allow to study what follows a given word in a sequence but not what is preceding this word. The most general solution to handle this probleme is to build a second tree for the reverse sequence. A second possibility is already been describe before: it consits to build a new tree representing the context of the word.

4 Conclusion

We have developped a tool allowing to browse in a large sequence, in the search of repeats. To our knowledge, it is a totally new way of approaching the analysis of sequences and we claim that it becomes necessary due to the size of available sequences. However, what we have called repeats in this paper is mainly the exact copy of words. It is well known that the biological concept of repeat is far more complex. The mechanism of duplication includes evenements such as inversions and mutations. Preliminary works suggest that these useful extensions might be taken into account in our framework [Leu91], [NEM96].

Acknowledgements

Part of this research has been funded by the GIP GREG, with a contract on the classification and characterization of sequences (decision 59), with the collaboration of A. Danchin and I. Moszer.

References

[AB85] A. Blumer, J. Blumer, D. Haussler, H. Ehrenfeucht, M. Chen, and J. Seiferas. The samallest automaton recognizing the subwords of a text. *Theoret. Comput. Sci.*, Vol. 40, 1985, pp. 31–55.

[Apo85] A. Apostolico. Thy myriad virtures of subword trees. In: *Combinatorial Algorithms on Words*, 1985, pp. 85–96.

[CS85] M.T. Chen and J. Seiferas. Efficient and elegant subword-tree construction. In: *Combinatorial Algorithms on Words*, 1985, pp. 97–107.

[JN96] J. Nicolas, R. Gras (J. Lebbe, T. Basavannepa, R. Vignes). Forest, un fouineur de répétitions dans les grandes séquences biologiques. *Rapport de recherche INRIA*, 1996.

[KMGL87] S. Karlin, M. Morris, G. Ghandour, and M.Y. Leung. Efficient algorithms for molecular sequence analysis. In: *Proc. Natl. Acad. Sci. USA*, 1987, pp. 841–845.

[KVD95] F. Kunst, A. Vassarotti and A. Danchin. Organization of the European bacillus subtilis genome sequencing project. *Microbiology*, Vol. 141, February, 1995, pp. 249–255.

[Leu91] M.Y. Leung. An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *Mol. Biol.*, No. 221, 1991, 1367–1378.

[LI93] C. Lefèvre and J. Ikeda. The position end-set tree: a small automaton for word recognition in biological sequences. *CABIOS*, Vol. 9, No. 3, 1993, pp. 343–348.

[McC76] E.M. McCreight. A space-economial suffix tree construction algorithm. *Journal of the Association for Computing Machinery*, Vol. 23, No. 2, 1976, pp. 262–272.

[MGD95] I. Moszer, P. Glaser and A. Danchin. Subtilist: a relational database for the bacillus subtilis genome. *Microbiology*, Vol. 141, February, 1995, pp. 261–268.

[MVHD93] C. Médigue, C. Viari, A. Hénaut, and A. Danchin. Colibri: a functional data base for the escherichia coli genome. *Microgiol. Rev.*, VOl. 57, 1993, pp. 623–654.

[NEM96] N. El-Mabrouk and M. Crochemore. Boyer-moore strategy to efficient approximate string matching. *Lecture Notes in Computer Science*, Vol. 1075, 1996, pp. 24–38.

[SA90] S.F. Altschul, W. Miller (E.W. Myers, D.J. Lipman). Basic local alignment search tool. *J. Biol. Mol.*, Vol. 215, 1990, pp. 403–410.

[Wei73] P. Weiner. Linear pattern matching algorithms. In ': *Proc. 14th Annual Symposium on Switching and Automata Theory*, IEEE Computer Society, 1973, pp. 1–11.

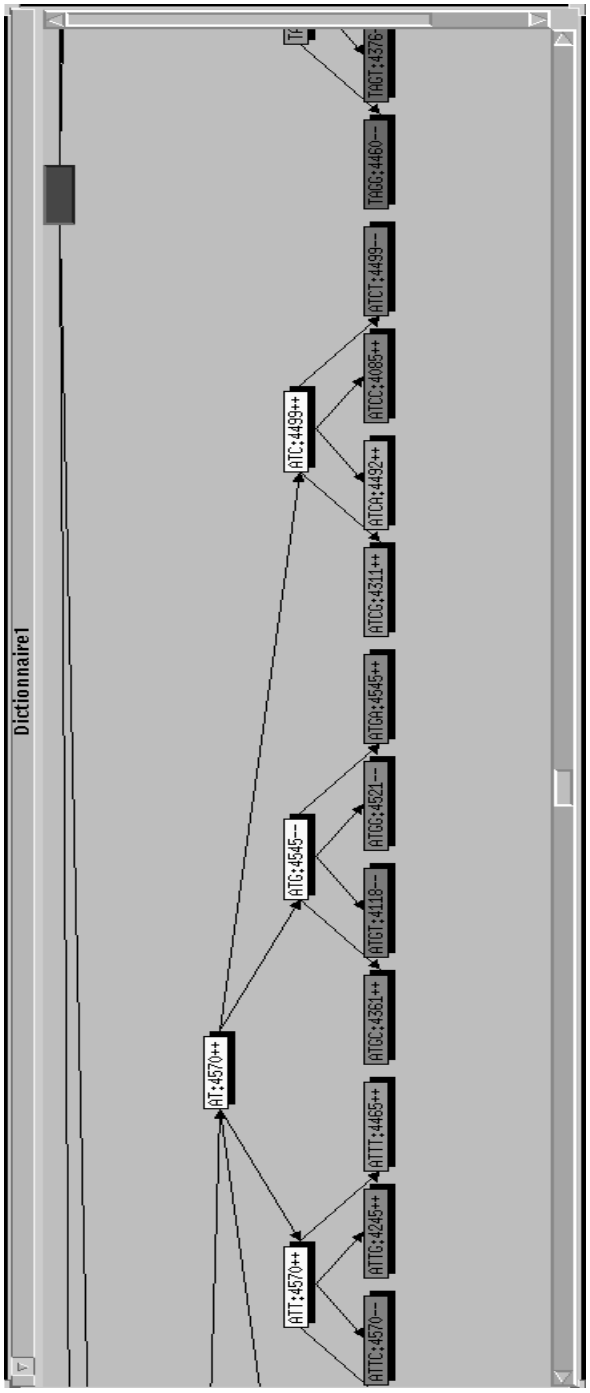

Figure 2:

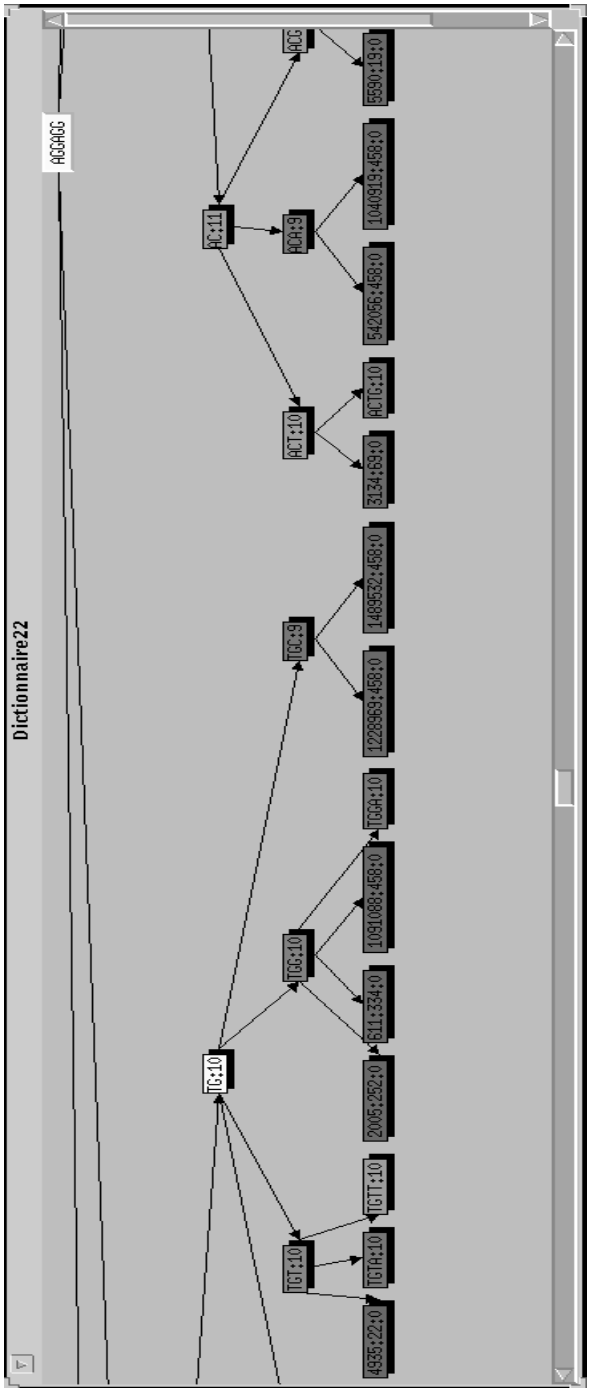

Figure 3:

Figure 4:

Figure 5: