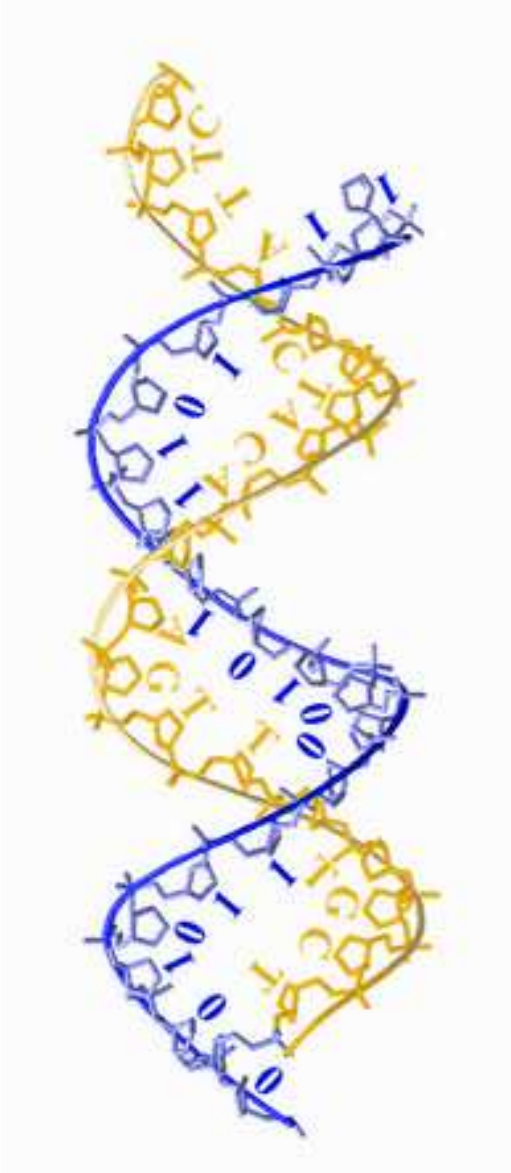


Crédit image : Université Würzburg

Actes des journées

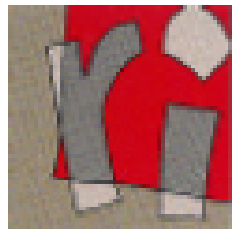
MATHSTIC

Algorithmique Génomique

24 – 25 novembre 2005

Faculté des Sciences d'Orsay

Édités par E. Rivals et S. Vialette



Réunion thématique
à Orsay
24-25 novembre 2005
http ://www.lirmm.fr/~rivals/RESEARCH/MATHSTIC/

Lieu : Faculté des sciences d'Orsay , Salle 103, Batiment 338, à Orsay.
Planning : Jeudi 24 nov. à 9h00 jusqu'au vendredi 25 nov. à 17h.

Notre objectif est de réunir la communauté nationale travaillant sur la thématique de l'algorithmique génomique pour promouvoir les échanges, les discussions et les collaborations dans ce domaine de recherche. Elle permettra de cerner les avancées récentes et les défis futurs dans ce domaine. Les principales *thématiques* abordées sont :

- algorithmique du texte
- recherche de motifs
- algorithmique des arbres
- combinatoire des mots
- modélisation et statistique des textes
- algorithmique des graphes

et leurs *applications* à :

- découverte de gènes
- comparaison de séquences
- génomique comparative
- détection de répétitions
- analyse d'ARN
- annotation du génome
- génomique fonctionnelle
- biologie systémique.

La réunion de novembre prochain se veut un lieu d'échanges francophones sur les dernières avancées scientifiques dans le domaine de l'algorithmique génomique et ses applications.

Comité de sélection :

- Alain Denise (LRI - Univ. Orsay)
- Eric Rivals (LIRMM - Univ. Montpellier)
- Hélène Touzet (LIFL - Univ. Lille)
- Stéphane Vialette (LRI - Univ. Orsay)

Comité d'organisation :

- A. Denise (LRI - Univ. Orsay)
- E. Rivals (LIRMM Univ. Montpellier)
- S. Vialette (LRI - Univ. Orsay)

Soutiens :

- CNRS Département STIC
- GDR Algorithmique, Langage et Programmation
- Laboratoire de Recherche en Informatique UMR 8623 (CNRS-Université Paris-Sud)
- Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier UMR 5506 (CNRS-Université Montpellier II)
- Université Paris-Sud

Liste des résumés des Journées « Algorithmique génomique »

1 Computing common intervals of K permutations, with applications to modular decomposition of graphs

Anne Bergeron[1], Cedric Chauve[1], Fabien de Montgolfier[2], Mathieu Raffinot[3]
[1]LaCIM, Université du Québec à Montréal, Canada.
[2]LIAFA, Université Denis Diderot - Case 7014, 2 place Jussieu, F-75251 Paris Cedex 05, France.
[3]CNRS - Laboratoire Génome et Informatique, Tour Evry 2, 523, Place des Terrasses de l'Agora, 91034 Evry, France.
{anne,chauve}@lacim.uqam.ca, fm@liafa.jussieu.fr, raffinot@genopole.cnrs.fr

The notion of *common interval* was introduced in [Uno and Yagiura, 2000] in order to model the fact that, when comparing genomes, a group of genes can be rearranged but still remain connected. In [Uno and Yagiura, 2000], Uno and Yagiura proposed a first algorithm that computes the set of common intervals of a permutation P with the identity permutation in time $O(n + N)$, where n is the length of P, and N is the number of common intervals. However, N can be of size $O(n^2)$, thus the algorithm of Uno and Yagiura has an $O(n^2)$ time complexity. Heber and Stoye [Heber and Stoye, 2001] defined a subset of size $O(n)$ of the common intervals of K permutations, called *irreducible* intervals, that forms a basis of the set of all common intervals: every common interval is a chain overlapping irreducible intervals. They proposed an $O(Kn)$ time algorithm to compute the set of irreducible intervals of K permutations, based on Uno and Yagiura's algorithm.

One of the drawbacks of these algorithms is that properties of Uno and Yagiura's algorithm are difficult to understand [Xuan et al., 2005]. Even the authors describe their $O(n + N)$ algorithm as *"quite complicated"*, and, in practice, simpler $O(n^2)$ algorithms run faster on randomly generated permutations [Uno and Yagiura, 2000]. On the other hand, Heber and Stoye's algorithms rely on a complex data structure that mimics what is known, in the theory of modular decomposition of graphs, as the PQ-trees of *strong intervals* [Booth and Lueker, 1976, McConnell and de Montgolfier, 2005]. An incentive to revisit this problem is the central role that these PQ-trees seem to play in the field of comparative genomics. Strong intervals can be used to identify significant groups of genes that are conserved between genomes [Landau et al., 2005], or as guides to reconstruct evolution scenarios [Bérard et al., 2005, Figeac and Varré, 2004].

In order to design alternative efficient algorithms to compute common intervals, we propose a theoretical framework for common intervals based on generating families of intervals. We formalize two concepts about common intervals, namely generators and canonical representation, that proved to have important algorithmic implications. Indeed, the combinatorial properties of these objects, and in particular the different links between them, are central in the design and the analysis of the simple optimal algorithms for computing common intervals of permutations we present.

For two permutations, these families can be computed by straightforward $O(n)$ algorithms that use only tables and stacks as data structures, and that upgrade trivially to the case of K permutations. Using these families, we compute common intervals with simple $O(n+N)$ and $O(n)$ algorithms whose properties can be readily verified. It is important to highlight that our algorithms are really "optimal" since they are based on very elementary manipulations of stacks and arrays. This is, we believe, a significant improvement over the existing algorithms that are based on intricate data structures, both in terms of ease of implementation and time efficiency, and in terms of understanding the underlying concepts [Heber and Stoye, 2001, Uno and Yagiura, 2000].

Moreover, we showed how, transposed in the more general context of modular decomposition of graphs, our results have a similar impact and lead to a significant simplification of some existing algorithms. Indeed, existing linear-time modular decomposition algorithms [Cournier and Habib, 1994, McConnell and Spinrad, 1994] are rather complex and many efforts have been made in the design of decomposition algorithms that are efficient in practice, even if they do not run in linear time but in quasi-linear time. Using the simple factorizing permutation algorithm of [Habib et al., 2004] and a simple right-modules identification algorithm, a generator of the interval-modules can easily be computed in linear time; our algorithm building the PQ-tree can then be used to compute the strong interval-modules, which also are the strong modules, and the *modular decomposition tree*.

During the talk we will talk about
– The notion of generators of common intervals
– How to compute generators of K permutations of size n in $O(Kn)$ time
– Explain how to generate the set of all N common intervals in $O(n + N)$ using a generator
– A new linear space basis of common intervals, called the canonical generator
– The relationship between this new basis and the classical basis of strong intervals
– A simple linear-time algorithm to construct the strong intervals basis, given the canonical generator
– An extension of our results to the modular decomposition of graphs.
Full version of the article can be found in [Bergeron et al., 2005]. This talk was presented in ESA'O5.

References

[Bérard et al., 2005] Bérard, S., Bergeron, A., and Chauve, C. (2005). Conserved structures in evolution scenarios. In *Comparative Genomics, RECOMB 2004 International Workshop, vol. 3388 of* Lecture Notes in Comput. Sci., pages 1–15.

[Bergeron et al., 2005] Bergeron, A., Chauve, C., de Montgolfier, F., and Raffinot, M. (2005). Computing common intervals of k permutations, with applications to modular decomposition of graphs. LIAFA technical report 2005-006 available at `http://www.liafa.jussieu.fr/web9/rapportrech/listrapport_fr.php?anscol=2005`.

[Booth and Lueker, 1976] Booth, S. and Lueker, G. (1976). Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-trees algorithms. *J. Comput. Syst. Sci.*, 13:335–379.

[Cournier and Habib, 1994] Cournier, A. and Habib, M. (1994). A new linear algorithm for modular decomposition. In *Trees in algebra and programming – CAAP'94, 19th International Colloquium, vol. 787 of* Lecture Notes in Comput. Sci., pages 68–84.

[Figeac and Varré, 2004] Figeac, M. and Varré, J.-S. (2004). Sorting by reversals with common intervals. In *Algorithms in Bioinformatics, 4th International Workshop, WABI 2004, vol. 3240 of* Lecture Notes in Comput. Sci., pages 26–37.

[Habib et al., 2004] Habib, M., de Montgolfier, F., and Paul, C. (2004). A simple linear-time modular decomposition algorithm for graphs, using order extension. In *SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, vol. 3111 of* Lecture Notes in Comput. Sci., pages 187–198.

[Heber and Stoye, 2001] Heber, S. and Stoye, J. (2001). Finding all common intervals of k permutations. In *Combinatorial Pattern Matching, 12th Annual Symposium, CPM 2001, vol. 2089 of* Lecture Notes in Comput. Sci., pages 207–218.

[Landau et al., 2005] Landau, G., Parida, L., and Weimann, O. (2005). Gene proximity analysis across whole genomes via pq trees. In *6th Combinatorial Pattern Matching Conference (CPM).* To appear in Journal of Computational Biology.

[McConnell and de Montgolfier, 2005] McConnell, R. M. and de Montgolfier, F. (2005). Algebraic operations on pq-trees and modular decomposition trees. In *WG'05, 31st International Workshop on Graph-Theoretic Concepts in Computer Science.*

[McConnell and Spinrad, 1994] McConnell, R. M. and Spinrad, J. (1994). Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 536–545. ACM/SIAM.

[Uno and Yagiura, 2000] Uno, T. and Yagiura, M. (2000). Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309.

[Xuan et al., 2005] Xuan, B. M. B., Habib, M., and Paul, C. (2005). From permutations to graph algorithms. LIRMM technical report RR-05021.

2 Recherche de motifs pour les sources corrélées

Jérémie Bourdon[1], **Brigitte Vallée**[2]
[1]LINA/CNRS, Université de Nantes,
[2]GREYC/CNRS, Université de Caen.
jeremie.bourdon@univ-nantes.fr, Brigitte.Vallee@info.unicaen.fr

Dans l'étude des algorithmes de recherche de motifs, on s'intéresse principalement à deux paramètres : le nombre d'occurrences d'un motif dans un texte, et le nombre de positions dans le texte où une occurrence peut se terminer. Ces deux paramètres peuvent être sensiblement différents, par exemple *aabaaab* contient 5 occurrences de l'expression régulière a^+b qui peuvent se terminer à 2 positions distinctes. Vis à vis de la complexité des algorithmes de recherche associés, le premier paramètre est lié à la complexité d'un algorithme qui énumère toutes les occurrences présentes dans le texte et le second est étroitement associé à la complexité d'un algorithme (souvent dynamique) de comptage du nombre d'occurrences (la mise à jour ne pouvant se faire que lorsqu'une nouvelle occurrence a été trouvée).

Nous étudions les comportements statistiques de ces deux paramètres dans un cadre très général. Tout d'abord, le cadre probabiliste est celui de sources corrélées – les sources dynamiques. C'est un processus aléatoire qui modélise entre autres les cadres classiques d'étude des chaînes de caractères (sources de Bernoulli et chaînes de Markov), mais permet aussi de modéliser des processus plus généraux, où il peut exister des interactions entre deux symboles quelconques du texte, quelle que soit leur distance.

Le type de motif recherché est lui aussi assez général. Dans le cas où le motif est décrit par une expression régulière, nous étudions le nombre de positions d'occurrences par le biais de l'automate induit par l'expression régulière. Ici, le nombre de composantes fortement connexes de l'automate contenant des états finaux intervient fortement dans l'analyse. Le résultat présenté concerne les (graphes des) expressions régulières à une composante fortement connexe mais la méthode s'étend naturellement à un nombre quelconque de composantes fortement connexes. Dans le cas où le motif est un ensemble de mots finis, nous étudions le nombre d'occurrences. Le graphe de De Bruijn, pondéré par les nombres d'occurrences ajoutés par chaque transition joue un grand rôle dans l'étude.

A chaque fois, le problème étudié peut s'exprimer sous la forme de graphes pondérés. La matrice de transition de ce graphe se traduit alors en une matrice d'opérateurs fonctionnels liés à la dynamique de la source probabiliste. En quelque sorte, cet opérateur matriciel combine à la fois la structure algébrique du problème et la dynamique de la source. Enfin, sur un espace de fonctions adapté et moyennant de bonnes conditions sur la source, un résultat de décomposition, à la Perron-Frobenius, de l'opérateur est obtenu.

Avec des hypothèses naturelles sur l'expression régulière considérée, la moyenne et la variance du nombre de positions où une occurrence d'une expression régulière peut apparaître dépendent linéairement de la taille du texte considéré lorsque le texte est produit par une "bonne" source dynamique. De plus, la variable aléatoire correspondante suit asymptotiquement une loi gaussienne. Le résultat est le même pour le nombre d'occurrences d'un ensemble de motif dans un texte.

Nos résultats étendent donc des résultats semblables, seulement obtenus précédemment dans le cadre classique (sources de Bernoulli et chaînes de Markov) [Flajolet et al., 2002, Flajolet et al., 2005, Goldwurm and Lonati, 2005].

Références

[Flajolet et al., 2002] Flajolet, P., Nicodème, P., and Salvy, B. (2002). Motif Statistics. *Theoretical Computer Science*, 287(2) :595–619.

[Flajolet et al., 2005] Flajolet, P., Szpankowski, W., and Vallée, B. (2005). Hidden Word Statistics. *à paraître dans Journal of the ACM.*

[Goldwurm and Lonati, 2005] Goldwurm, M. and Lonati, V. (2005). Pattern Occurrences in Multicomponent Models. In *Proceedings de STACS 2005*, pages 680–692.

3 Un cadre pour la combinaison d'heuristiques sur les alignements de séquence afin d'annoter les gènes

Sarah Djebali, Franck Delaplace, Hugues Roest Crollius
Laboratoire Dynamique et Organisation des Genomes (DyOGen), CNRS UMR 8541, ENS, 46 rue d'Ulm, 75005 Paris
djebali@ens.fr

Annoter les gènes d'une séquence d'ADN génomique eucaryote signifie localiser les extrémités de chacun des gènes qu'elle contient, et pour chacun d'entre eux déterminer tous les transcrits alternatifs. A l'échelle du génome cette tâche s'avère difficile pour plusieurs raisons. D'abord les gènes eucaryotes sont composés d'une succession d'exons et d'introns, ce qui rend leur structure complexe et très variable. Ensuite les gènes ne couvrent qu'une petite fraction des génomes eucaryotes (30% chez les mammifères), et les exons une fraction encore moindre (1 à 2% chez les mammifères). Les génomes eucaryotes contiennent aussi de nombreuses copies non fonctionnelles de gènes, appelées *pseudogènes*, qui sont souvent inclus dans les introns des gènes, et ont une composition en nucléotides très similaire à celle des gènes. Enfin chaque gène peut donner lieu, en fonction des besoins de la cellule à un instant et un endroit donné, à plusieurs transcrits alternatifs.

Cependant annoter les gènes demeure fondamental dans le domaine de la recherche biomédicale, puisque cela permet de faire le lien entre génotype et phénotype chez l'homme et les organismes modèles, et de façon générale de focaliser le travail des biologistes et des bioinformaticiens sur une partie fonctionnelle importante des génomes.

Des modèles mathématiques ont été proposés pour annoter les gènes de façon automatique dans l'ADN génomique. Ces modèles se divisent en trois catégories : d'une part les modèles dits *ab initio*, qui sont des modèles statistiques réalisant un apprentissage sur des séquences dont les gènes ont été annotés, afin de prédire ensuite ceux de séquences inconnues [Burge and Karlin, 1997, Krogh, 1997, Guigo, 1998] ; d'autre part les modèles dits *de similarité de séquence*, qui consistent à comparer une séquence d'ADN inconnue S à des séquences biologiques diverses (ADN, ARNm, protéine), de la même ou d'une autre espèce, afin d'en déduire la localisation des gènes de S [Gelfand et al., 1996, Mott, 1997, Meyer and Durbin, 2002, Blayo et al., 2003] (notons que les résultats de tels programmes sont aussi appelés *alignements de séquence*) ; enfin des modèles hybrides qui tentent de combiner les deux modèles précédents afin de prédire les gènes de façon plus fiable [Rogic et al., 2002, Howe et al., 2002, Allen and Salzberg, 2005].

Malgré ces avancées considérables, les sensibilités et specificités des programmes d'annotation de gènes sont au mieux de l'ordre de 40% en séquence codante exactement prédite, et la plupart de ces programmes ne prédisent encore qu'un transcrit par gène. De ce fait la référence en matière d'annotation reste l'*expert humain*. Nous nous sommes donc interrogés : comment l'expert humain procède-t-il pour annoter les gènes ? En réalité l'annotateur humain procède de façon semi-automatique puisqu'il se fonde pour annoter les gènes d'une séquence S, sur des résultats d'alignement avec des séquences biologiques qu'il considère comme fiables, telles que cDNAs (pleine longueur ou EST), et protéines de différentes espèces. Fondé sur ses connaissances à la fois de la biologie et des alignements dont il dispose, donc sur des *heuristiques*, l'annotateur humain combine ensuite ces alignements de façon à constituer les différents transcrits alternatifs des gènes.

Nous développons actuellement un cadre pour la combinaison d'heuristiques sur les alignements de séquence, dans le but idéal final d'annoter les gènes d'une séquence S de façon aussi fiable que l'humain. Ce cadre repose sur l'utilisation de multi-graphes orientés acycliques colorés, aussi appelés *DACMs* (Directed Acyclic Coloured Multigraphs), dont les noeuds représentent des objets biologiques, et dont les arêtes multiples orientées représentent des relations de nature structurelle ou de priorité entre ces objets. Le principe est le suivant : un premier DACM est construit à partir d'alignements de séquence entre S et des séquences biologiques (par exemple ARNm et protéines), puis des chemins maximaux dont les arêtes et les noeuds vérifient certaines propriétés (justement fondées sur les heuristiques précédemment décrites), sont recherchés dans ce DACM. Chaque chemin maximal donne lieu, par un processus appelé *réduction de DACM*, au noeud d'un deuxième DACM, dont les arêtes multiples orientées sont établies sur la base de connaissances sur ces nouveaux objets. Notons que ces objets (noeuds du DACM 2), sont biologiquement plus

complexes que les objets du DACM 1, puisqu'ils regroupent chacun plusieurs objets du DACM 1. On itère ensuite cette procédure de réduction, jusqu'à obtenir un point fixe, c'est-à-dire un DACM qui n'est plus modifié par cette procédure. Les noeuds de ce dernier DACM représentent alors les annotations recherchées. Dans la pratique ces annotations sont le plus souvent les différents transcrits alternatifs des gènes de S.

Une implantation de ce cadre, nommée *Exogean* (EXpert On GEne ANnotation), a été développée, et annote les gènes d'une séquence d'ADN génomique en combinant les heuristiques sur les alignements de séquence utilisées par les experts humains. Dans le cadre du concours EGASP'05, Exogean a été comparé à une vingtaine de programmes d'annotation de gènes, et donne d'excellents résultats.

Références

[Allen and Salzberg, 2005] Allen, J. and Salzberg, S. (2005). Jigsaw : integration of multiple sources of evidence for gene prediction. *Bioinformatics*, 18 :3596–3603.

[Blayo et al., 2003] Blayo, P., Rouzé, P., and Sagot, M.-F. (2003). Orphan gene finding - an exon assembly approach. *Theoretical Computer Science*, 290 :1407–1431.

[Burge and Karlin, 1997] Burge, C. and Karlin, S. (1997). Prediction of complete gene structures in human genomic dna. *Journal of Molecular Biology*, 268 :78–94.

[Gelfand et al., 1996] Gelfand, M., Mironov, A., and Pevzner, P. (1996). Gene recognition via spliced alignment. *Proceedings of the National Academy of Sciences of the USA*, 93 :9061–9066.

[Guigo, 1998] Guigo, R. (1998). Assembling genes from predicted exons in linear time with dynamic programming. *Journal of Computational Biology*, 5 :681–702.

[Howe et al., 2002] Howe, K., Chothia, T., and Durbin, R. (2002). Gaze : a generic framework for the integration of gene-prediction data by dynamic programming. *Genome Research*, 12 :1418–1427.

[Krogh, 1997] Krogh, A. (1997). Two methods for improving performance of an hmm and their application for gene finding. In *AAAI ISMB1997*, pages 179–186.

[Meyer and Durbin, 2002] Meyer, I. and Durbin, R. (2002). Comparative ab initio prediction of gene structures using pair hmms. *Bioinformatics*, 18 :1309–1318.

[Mott, 1997] Mott, R. (1997). Est-genome : a program to align spliced dna sequences to unspliced genomic dna. *Compututational Applications in the Biosciences*, 13 :477–478.

[Rogic et al., 2002] Rogic, S., Ouellette, B. F., and Mackworth, A. (2002). Improving gene recognition accuracy by combining predictions from two gene-finding programs. *Bioinformatics*, 18 :1034–1045.

4 Pyramid diagram: detecting and visualizing the organization of repetitive sequences in genomes

Patrick Durand, Frédéric Mahé, Mathieu Giraud, Anne-Sophie Valin, Jacques Nicolas
IRISA-INRIA, Projet Symbiose, Campus de Beaulieu, 35042 Rennes Cedex.
pdurand@irisa.fr

Abstract

We present the pyramid diagram, or pygram, a new visualization method which aims at abstracting the organization of the repeated structures in genomic sequences. The pygram is created with the idea of visualizing all exact maximal repeats (eMR) located within a (set of) sequence(s) without producing any link between pairs of eMR. By choosing to highlight all eMR in that way, a pygram not only displays all the possible repeats of sub-sequences, it also reveals their hierarchical organization throughout genome sequences.

Résumé

Nous proposons une nouvelle méhode de visualisation des structures de répétitions identitifiées dans les séquences génomiques. Cette méthode repose sur un diagramme, appelé le diagramme pyramide ou pygram, présentant graphiquement l'ensemble des répétitions maximales exactes (eMR) localisées sur une (ou un ensemble de) séquence(s) et sans produire les liens entre les paires d'eMR. Cette représentation offre l'avantage de visualiser toutes les répétitions possibles de sous-séquences, ainsi que leur organisation hiérarchique au sein des séquences génomiques.

Introduction

A wide number of in silico studies of repeated sequences have lead to the hypothesis that they play major roles in the structure, the function, the dynamics and the evolution of genomes in Archaea [Blount and Grogan, 2005, Mojica et al., 2005], Bacteria [Achaz et al., 2002, Pourcel et al., 2005] and Eukarya [Achaz et al., 2000, Friedman and Hughes, 2001, Kazazian, 2004]. Repeats may also be strictly conserved through the evolution, as revealed by comparisons of human, mouse, rat, chicken and dog genomes [Bejerano et al., 2004]. Complex mechanisms such as chromosome segments duplications, or even whole genome duplications, are thought to occur, explaining genome evolutions [Taylor et al., 2003, Dujon et al., 2004].

When targeting the analysis of repeated sequences, one has to face two problems: the definition of a repeat to target, and the analysis of observed repeats in a (set of) sequence(s).

A number of formal definitions have been proposed to capture the essence of observable repeats, such as tandem repeats (e.g., [Wexler et al., 2004] for a brief review), longest normal repeat [Karp et al., 1972], maximal repeats [Gusfield, 1997] and longest repeats with a block of don't cares [Crochemore et al., 2004]. Among them, the notion of exact maximal repeat (eMR) is quite attractive since it only focuses on sequences present in two largest common blocks, without possible left or right extension, and without any biological a priori. Using eMR as building blocks, it is possible to model more complex repeats, such as error-prone ones or including a bounded gap. Maximal repeats have nice properties: they can be computed in linear time using a suffix-tree based algorithm and their number is linear (at most n exact maximal repeats in a sequence of size n).

Visualization of genome sequence data is a widely explored issue in bioinformatics. Regarding the problem of interpreting repeats content in genome sequences, dedicated tools have been proposed, such as dotplot [Gibbs and McIntyre, 1970], chaos game [Jeffrey, 1999], percent identity plot [Schwartz et al., 2000], repeat-graph [Kurtz et al., 2001] and BARD [Spell et al., 2003]. However, interpretation of views created by these tools is quite difficult, especially for long genomes, since most of them rely on displaying pairs of repeats. They usually do not provide convenient zooming capabilities to facilitate the analysis of regions of particular interest. Moreover, they are not able to summarize the hierarchical organization of repetitive structures in a convenient way so

that they become interpretable by the end users. To overcome these limitations, we propose the pyramid diagram.

The pyramid diagram

A pyramid diagram, or pygram, is a bi-dimensional plot where a DNA sequence and all its eMR are mapped along the x-axis. Each eMR is represented as a coloured triangle (or pyramid) of height proportional to its size. Additional information, such as the number of occurrences, is also graphically represented on the diagram.

Construction of pygrams proceeds in two steps. First, we produce the whole set of eMR using an implementation of Gusfields eMR detection algorithm [Gusfield, 1997] acting on a generalized suffix-tree. This implementation can then be used to locate eMR either within a single sequence or between several sequences. Second, eMR are indexed. This index aims at ordering eMR along the sequences, so that we can achieve good performance for interactive pygrams creation. The index also allows rapid filtering of eMR to create pygrams with eMR verifying particular constraints, such as their size, their location (position on normal vs. reverse complement strand), and their conservation between various sequences, among others.

We have implemented a software suite that provides an efficient eMR browser allowing the user to in-depth analyse eMR organization at various levels, from the higher level (i.e., the entire sequence) down to the lower level (i.e., a single nucleotide).

We have used this visualization technique on Virus and Archaea genomes analysis. A first study illustrates the detection of highly repeated structures in the genome of Sulfolobus solfataricus (see Figure 1), whereas the second study targets the analysis of the integration of SIRV1 virus within the genome of its host, S. solfataricus.

These studies show that pygram visualization is well suited to display the complex organization of repeated sequences within a single genome sequence or between sequences. It also opens perspectives on the application of this technique to other types of repeats. More precisely, we are now interested in studying assembly of maximal repeats, both on the algorithmic side (detection) and on the visualization side (interpretation).

References

[Achaz et al., 2000] Achaz, G., Coissac, E., Viari, A., and Netter, P. (2000). Analysis of intrachromosomal duplications in yeast saccharomyces cerevisiae: a possible model for their origin. *Mol Biol Evol*, 17(8):1268–75.

[Achaz et al., 2002] Achaz, G., Rocha, E., Netter, P., and Coissac, E. (2002). Origin and fate of repeats in bacteria. *Nucleic Acids Res*, 30(13):2987–94.

[Bejerano et al., 2004] Bejerano, G., Pheasant, M., Makunin, I., Stephen, S., Kent, W., Mattick, J., and Haussler, D. (2004). Ultraconserved elements in the human genome. *Science*, 304(5675):1321–1325.

[Blount and Grogan, 2005] Blount, D. and Grogan, D. (2005). New insertion sequences of sulfolobus: functional properties and implications for genome evolution in hyperthermophilic archaea. *Mol Microbiol*, 55(1):312–25.

[Crochemore et al., 2004] Crochemore, M., Iliopoulos, C., Mohamed, M., and Sagot, M. (2004). Longest repeats with a block of don't cares. In *LATIN*, pages 271–278.

[Dujon et al., 2004] Dujon, B., Sherman, D., and Fischer, G. e. a. (2004). Genome evolution in yeasts. *Nature*, 430(6995):35–44.

[Friedman and Hughes, 2001] Friedman, R. and Hughes, A. (2001). Gene duplication and the structure of eukaryotic genomes. *Genome Res*, 11(3):373–81.

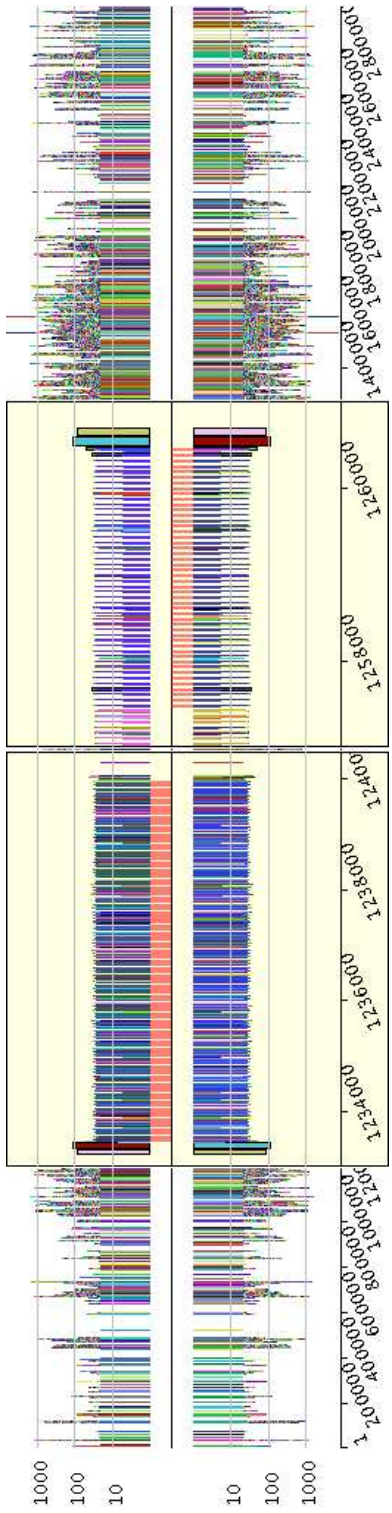


Figure 4.1: pygram of Sulfolobus solfataricus P2 genome. The whole genome sequence is presented with normal (N) strand view above the black horizontal line, and the reverse complement (RC) strand view below this line. X-axis corresponds to the sequence coordinate system. Logarithmic Y-axis corresponds to eMR size, and horizontal grey lines for eMR of size 10, 100 and 1000 nucleotides are displayed on both N and RC views. Due to the high density of information, each eMR located in the genome is sketched by a coloured vertical bar instead of a triangle. The small red boxes located between the black line and the N and RC views highlight the most repetitive eMR in this genome. The two yellow boxes represent magnify lenses that better reveal the repetitive organization of the two small regions (from 1,233,000 to 1,240,500 and from 1,257,000 to 1,261,500) where this eMR is located.

[Gibbs and McIntyre, 1970] Gibbs, A. and McIntyre, G. (1970). The diagram, a method for comparing sequences. its use with amino acid and nucleotide sequences. *Eur J Biochem*, 16(1):1–11.

[Gusfield, 1997] Gusfield, D. (1997). *Algorithms on strings, trees, and sequences.* Cambridge University Press.

[Jeffrey, 1999] Jeffrey, H. (1999). Chaos game representation of gene structure. *Nucleic Acids Res*, 18(8):2163–70.

[Karp et al., 1972] Karp, R., Miller, R., and Rosenberg, A. (1972). Rapid identification of repeated patterns in strings, trees and arrays. In *STOC '72: Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 125–136, New York, NY, USA. ACM Press.

[Kazazian, 2004] Kazazian, H. (2004). Mobile elements: Drivers of genome evolution. *Science*, 303(5664):1626–1632.

[Kurtz et al., 2001] Kurtz, S., Choudhuri, J., Ohlebusch, E., Schleiermacher, C., J., S., and Giegerich, R. (2001). Reputer: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Res*, 29(22):4633–4642.

[Mojica et al., 2005] Mojica, F., Diez-Villsenor, C., Garcia-Martinez, J., and Soria, E. (2005). Intervening sequences of regularly spaced prokaryotic repeats derive from foreign genetic elements. *J Mol Evol*, 60(2):174–182.

[Pourcel et al., 2005] Pourcel, C., Salvignol, G., and Vergnaud, G. (2005). Crispr elements in yersinia pestis acquire new repeats by preferential uptake of bacteriophage dna, and provide additional tools for evolutionary studies. *Microbiology*, 151:653–63.

[Schwartz et al., 2000] Schwartz, S., Zhang, Z., Frazer, K., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W. (2000). Pipmaker–a web server for aligning two genomic dna sequences. *Genome Res*, 10(4):577–586.

[Spell et al., 2003] Spell, R., Brady, R., and Dietrich, F. (2003). Bard: A visualization tool for biological sequence analysis. In *INFOVIS*.

[Taylor et al., 2003] Taylor, J., Braasch, I., Frickey, T., Meyer, A., and Van de Peer, Y. (2003). Genome duplication, a trait shared by 22000 species of ray-finned fish. *Genome Res*, 13(3):382–390.

[Wexler et al., 2004] Wexler, Y., Yakhini, Z., Kashi, Y., and Geiger, D. (2004). Finding approximate tandem repeats in genomic sequences. In *RECOMB*, pages 223–232.

5 Validation de tableaux de bords

Jean-Pierre Duval[1], **Thierry Lecroq**[2], **Arnaud Lefebvre**[2]
[1]LIFAR (Laboratoire d'Informatique Fondamentale et Appliquée de Rouen),
[2]ABISS (Atelier Biologie Informatique Statistiques et Sociolinguistique),
Université de Rouen.
{arnaud.lefebvre, thierry.lecroq}@univ-rouen.fr

Introduction

Un bord u d'un mot w est un préfixe et un suffixe de w tel que $u \neq w$. La notion de bord est duale de la notion de période. La périodicité des mots a déjà été largement étudiée et présente un intérêt évident pour un grand nombre d'applications telles que l'algorithmique génomique. Il est possible de calculer en temps linéaire la longueur des bords de tous les préfixes d'un mot. Le résultat de ce calcul constitue le tableau de bords d'un mot. Récemment, dans [Franěk et al., 2002], une méthode a été présentée pour vérifier si un tableau d'entiers f est un tableau de bords pour un mot w. Les auteurs donnent d'abord un algorithme linéaire « on-line » pour vérifier si f est un tableau de bords sur un alphabet de taille non bornée. Ils donnent ensuite un algorithme plus complexe dans le cas d'un alphabet borné. Dans [Duval et al., 2002] nous avons présenté un algorithme plus simple pour ce cas. De plus, si f est un tableau de bords, nous sommes capables de construire un mot w sur un alphabet de taille minimale pour lequel f est le tableau de bords.

Nous donnons ici une présentation plus simple de ce résultat qui permet également de générer l'ensemble de tableaux de bords valides.

Notations et définitions

Soit Σ un alphabet fini. Le bord d'un mot $w \in \Sigma^*$ est le plus long mot qui soit à la fois préfixe propre et suffixe propre de w. Soit $f[1..n]$ le tableau de bords d'un mot $w[1..n]$, étendu à $f[0] = -1$, défini par $f[i]$ = la longueur du bord de $w[1..i]$, $1 \leq i \leq n$. On dit qu'un tableau d'entiers $f[1..n]$ est *valide* si et seulement s'il est le tableau de bords d'au moins un mot $w[1..n]$ sur l'alphabet Σ. On définit la relation f sur $[0..n]$ par ifj si et seulement si $i = f[j]$. On note $\bar{f}$-classe la fermeture réflexive, symétrique et transitive de f sur $[1..n]$. Soit R la relation définie sur $[1..n]$ par iRj si et seulement si $(i-1)f(j-1)$ (autrement dit $i - 1 = f[j-1]$). On appelle R-chemin de i, $1 \leq i \leq n$, la suite des valeurs $(i_0 = i, i_1, \ldots, i_k = 0)$ telle que $i_{\ell-1}Ri_\ell$. On note δ' l'automate déterministe reconnaissant le langage $\Sigma^* w$, et δ son squelette (flèches sans étiquettes).

Résultats

Supposons $f[1..j]$ valide pour $1 \leq j < n$ alors $f[1..j+1]$ est valide si et seulement si $f[j+1]$ est le plus grand représentant de la $\bar{f}$-classe de $j+1$ sur le R-chemin de $j+1$ ($j+1$ exclu) (voir Fig.5.1).

Une condition nécessaire et suffisante pour construire un mot associé à un tableau de bords valide est la suivante : deux $\bar{f}$-classes différentes sur un même R-chemin doivent correspondre à deux lettres différentes.

Étant donné f le tableau de bords d'un mot w, il est possible de déterminer le squelette δ de l'automate reconnaissant le langage $\Sigma^* w$ et réciproquement. Il faut bien noter que la connaissance de w n'est pas nécessaire à ces calculs.

Perspectives

Il reste à étendre ces résultats à la fonction de faille de Knuth, Morris et Pratt [Knuth et al., 1977].

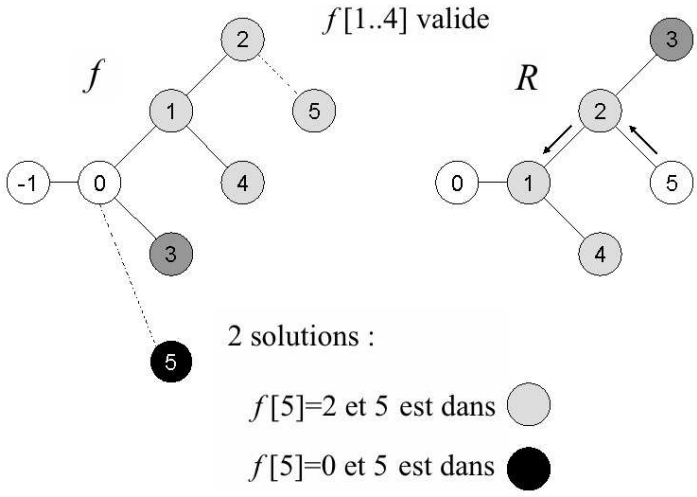


FIG. 5.1 – Exemple de test de validité. On suppose $f[1..4]$ valide. Les deux seules valeurs possibles pour $f[5]$ sont 2 et 0.

Références

[Duval et al., 2002] Duval, J.-P., Lecroq, T., and Lefebvre, A. (2002). Border array on bounded alphabet. In Balík, M. and Šimánek, M., editors, *Proceedings of the Prague Stringology Conference*, pages 28–35, Prague, République Tchèque. à paraître dans *Journal of Automata, Languages and Combinatorics*, (2005) 10.

[Franěk et al., 2002] Franěk, F., Gao, S., Lu, W., Ryan, P. J., Smyth, W. F., Sun, Y., and Yang, L. (2002). Verifying a border array in linear time. *Journal on Combinatorial Mathematics and Combinatorial Computing*, 42 :223–236.

[Knuth et al., 1977] Knuth, D. E., Morris, J. H., and Jr, V. R. Pratt (1977). Fast pattern matching in strings. *SIAM Journal on Computing*, 6(1) :323–350.

6 Combinatorial search on graphs motivated by bioinformatics applications: a case study and generalizations

Vladimir Grebinski[1] **et Gregory Kucherov**[2]
[1]Compugen USA, Inc., 77 Milltown Rd, East Brunswick, NJ 08816, USA,
[2]INRIA/LORIA, 615, rue du Jardin Botanique, B.P. 101, 54602, Villers-lès-Nancy.
grebinsk@cgen.com, Gregory.Kucherov@loria.fr

Identifying an unknown object via indirect queries is a type of problem that occurs very often in various applications. These problems are studied in the general area of *combinatorial search* [Aigner, 1988] or, for some more specific formulations, in its subarea called *group testing* [Du and Hwang, 1993].

The general applicative setting that motivates the present study can be described as follows. Assume we have a set of chemicals and each of them reacts with some others. Assume that we can mix two or several chemicals together and observe if some of them react with each other, or maybe even how many "reacting couples" there are in the mixture. Our goal is to recover all pairs of reacting chemicals by using as few experiments as possible. In bioinformatics, such type of problem arises, for example, in whole genome sequencing projects where contigs (contiguous sequenced fragments of the genome) are separated by gaps and the task consists in identifying the order of the contigs on the genome through a possibly minimal number of polymerase chain reactions (PCR) that reveal some information about the presence (and possibly the number) of neighbouring contigs from a given set of contigs.

Once chemicals and reactions are represented respectively by vertices and edges of a non-oriented graph, we come up with a problem of graph reconstruction. We first focus on a *boolean model* of queries where an arbitrary subset of vertices (chemicals) can be queried, yielding a 0-1 information depending on whether or not the subset contains at least two vertices related by an edge (reaction). For this model, we show that non-adaptive algorithms turn out to be less powerful than general adaptive algorithms: for example, Hamiltonian cycles can be reconstructed in $O(n \log n)$ adaptive queries [Grebinski and Kucherov, 1998] (which is an information-theoretic lower bound), whereas $\Theta(n^2)$ queries are necessary to make by non-adaptive algorithms [Alon et al., 2004]. Many other interesting classes of graphs require $\Theta(n^2)$ non-adaptive queries for their reconstruction [Alon et al., 2004, Alon and Asodi, 2004].

We then consider a *quantitative* model, where a query reveals the *number* of edges in the subgraph induces by the queried set of vertices. Here, the information-theoretic lower bound for Hamiltonian cycles is only $\Theta(n)$ queries and, somewhat surprisingly, it can be reached by a non-adaptive algorithm. Such an algorithm is based on two intricate combinatorial constructions: d-detecting and d-separating matrices. We then consider generalizations to some other graph classes that will further illustrate that under the quantitative model, non-adaptive algorithms often reach the asymptotic lower bound [Grebinski and Kucherov, 2000].

References

[Aigner, 1988] Aigner, M. (1988). *Combinatorial Search*. John Wiley & Sons.

[Alon and Asodi, 2004] Alon, N. and Asodi, V. (2004). Learning a hidden subgraph. In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 110–121. Springer. SIAM J. Discrete Math., to appear.

[Alon et al., 2004] Alon, N., Beigel, R., Kasif, S., Rudich, S., and Sudakov, B. (2004). Learning a hidden matching. *SIAM J. Computing*, pages 487–501. preliminary version in Proc. of the 43 IEEE FOCS, IEEE (2002), 197-206.

[Du and Hwang, 1993] Du, D.-Z. and Hwang, F. K. (1993). *Combinatorial Group Testing and its applications*, volume 3 of *Series on applied mathematics*. World Scientific.

[Grebinski and Kucherov, 1998] Grebinski, V. and Kucherov, G. (1998). Reconstructing a hamiltonian cycle by querying the graph: Application to DNA physical mapping. *Discrete Applied Mathematics*, 88:147–165.

[Grebinski and Kucherov, 2000] Grebinski, V. and Kucherov, G. (2000). Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28:104–124.

7 FADO : une méthode statistique pour détecter des distances évitées ou favorisées entre deux motifs le long d'une séquence

Gaelle Gusto et Sophie Schbath
INRA, Unité Mathématique, Informatique et Génome, Jouy-en-Josas.
{gusto,schbath}@jouy.inra.fr

L'étude des occurrences de motifs le long des génomes est devenue un thème classique de la génomique. Elle concerne par exemple leur fréquence ou leur répartition ; voir par exemple les deux récents ouvrages [Robin *et al.* (2003)] et [Lothaire (2005)]. Notre objectif ici est de modéliser la dépendance entre les occurrences d'un ou plusieurs motifs le long d'une séquence afin de déterminer de possibles corrélations. Ces dernières peuvent en effet révéler la participation de ces motifs à un même mécanisme biologique ou une interaction protéique induisant des contraintes d'espacement entre les motifs reconnus.

Notre méthode statistique consiste à estimer l'intensité du processus ponctuel $(U_i)_{i=1,\dots,I}$ formé des occurrences ordonnées d'un motif u. Nous utilisons pour cela un modèle de Hawkes qui considère que l'intensité dépend linéairement des distances aux occurrences précédentes du mot u et d'un deuxième mot v, via des profils d'influence à estimer. L'estimation de ces profils se fait en les décomposant dans une base de splines (polynômes par morceaux) et en estimant leur « coordonnées » par maximum de vraisemblance. Le nombre adéquat de noeuds équirépartis est choisi en maximisant un critère AIC de choix de modèles.

La procédure d'estimation a été validée sur des simulations puis appliquée à deux jeux de données biologiques. Le premier concerne la boite TATA et les gènes de *E. coli* ; le deuxième concerne des motifs impliqués dans la poly-adénylation des gènes de levure. Ce travail a donné lieu au programme FADO [Gusto and Schbath (2005)].

Références

[Gusto and Schbath (2005)] Gusto, G. and Schbath, S. (2005). FADO : a statistical method to detect favored or avoided distances between motif occurrences using the Hawkes' model. *Statistical Applications in Genetics and Molecular Biology.* **4** 1, Article 24, http ://www.bepress.com/sagmb/vol4/iss1/art24.

[Lothaire (2005)] Lothaire, M. (2005). *Applied Combinatorics on Words.* Cambridge University Press.

[Robin *et al.* (2003)] Robin, S., Rodolphe, F. and Schbath, S. (2003). *ADN, mots et modèles.* BELIN.

8 Édition et alignement de séquences arc-annotées selon une approche basée sur les arbres

Claire Herrbach, Alain Denise, Serge Dulucq, Hélène Touzet[3]
[1]LRI, Université Paris-Sud, 91405 Orsay Cedex,
[2]LaBRI, Domaine Universitaire, 351, cours de la Libération 33405 Talence Cedex,
[3]Laboratoire d'Informatique Fondamentale de Lille– UMR CNRS 8022 – Université de Lille 1
{herrbach,Alain.Denise}@lri.fr, dulucq@labri.fr, Helene.Touzet@lifl.fr

En toute généralité, le problème de la comparaison de structures secondaires d'ARN selon le modèle des séquences arc-annotées est NP-complet ([Lin et al., 2002, Blin et al., 2003]). D'un autre côté, en représentant les structures secondaires par des arbres ordonnés, plusieurs algorithmes polynomiaux existent ([Zhang and Shasha, 1989, Klein, 1998, Jiang et al., 1995]). Cependant, le fait que le problème soit polynomial dans ce contexte se paie naturellement par un certain nombre de contraintes. Par exemple, les opérations d'édition sont plus simples, et par conséquent bien moins réalistes d'un point de vue biologique que dans le modèle des séquences arc-annotées. La représentation par arbre ne permet pas notamment de prendre en compte simplement les opérations d'*arc-altering* et d'*arc-breaking*, qui affectent des appariements. Nous montrons qu'il est possible d'inclure ces opérations avec un algorithme qui a la même complexité que la comparaison d'arbres.

Références

[Blin et al., 2003] Blin, G., Fertin, G., Rusu, I., and Sinoquet, C. (2003). RNA sequences and the EDIT(NESTED, NESTED) problem. *technical report - LINA*.

[Jiang et al., 1995] Jiang, T., Wang, L., and Zhang, K. (1995). Alignment of trees - an alternative to tree edit. *Theoretical Computer Science*, 143(1) :137–148.

[Klein, 1998] Klein, P. (1998). Computing the edit-distance between unrooted ordered trees. In *6th European Symposium on Algorithms*, pages 91–102.

[Lin et al., 2002] Lin, G., Chen, Z.-Z., jiang, T., and Wen, J. (2002). The longest common subsequence problem for sequences with nested arc annotations. *Journal of Computer and System Sciences*, 65 :465–480.

[Zhang and Shasha, 1989] Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18(6) :1245–1262.

9 Détection des microsatellites dans les génomes : un problème insoluble ?

Sébastien Leclercq[1], Philippe Jarne[1], Eric Rivals[2]
[1]Centre d'Ecologie Fonctionnelle et Evolutive (C.E.F.E.),
CNRS, 1919 route de Mende, 34293 Montpellier cedex 5
[2]Laboratoire d'Informatique de Robotique de Microélectronique de Montpellier
(L.I.R.M.M.) - UMR 5506, CNRS et Université de Montpellier II
{sebastien.leclercq,philippe.jarne}@cefe.cnrs.fr, rivals@lirmm.fr

Les microsatellites sont des répétitions en tandem de pédiode courte (1 à 6 pb) présents dans les génomes de tous les organismes vivants. Chez certains organismes, ils représentent une part importante de l'ADN, comme chez l'homme où ils constituent environ 3% du génome [Consortium, 2001]. Certains de ces éléments ont une caractéristique remarquable qui est leur hypermutabilité, avec un taux de mutation de l'ordre de 0.001 chez l'homme [Goldstein and Schlötterer, 1999] et qui se traduit par l'ajout ou la suppression d'une ou de plusieurs répétitions. Ces changements de taille sont provoqués par un mécanisme particulier nommé glissement de polymérase, qui est complexe et encore mal compris [Goldstein and Schlötterer, 1999].

Depuis quelques années, la question de l'évolution des microsatellites commence à se poser. Une approche consiste à comparer des distributions de tailles théoriques (générées à partir de divers modèles de mutations) à des distributions réelles. Ces dernières sont issues de microsatellites connus et répertoriés [Jarne et al., 1998, Rolfsmeier et al., 2000, Dettman and Taylor, 2004] ou de détection de microsatellites dans les séquences génomiques via des algorithmes personnels [Kruglyak et al., 1998, Dieringer and Schlötterer, 2003, Sainudiin et al., 2004]. Une troisième solution possible est d'utiliser des algorithmes dédiés, qui sont plus puissants car fondés sur des notions algorithmiques poussées.

Plus d'une douzaine de ces algorithmes ont été publiés depuis 1997, sans compter les banques de données spécialisées accessibles via Internet. Il est possible de les regrouper en quatre classes principales :

- les méthodes d'alignement par rapport à une séquence consensus,
- les méthodes combinatoires de recherche de répétition,
- les approches heuristiques basées sur un critère statistique,
- les techniques basées sur la compression de l'information contenue dans la séquence.

Nous nous proposons donc de présenter plusieurs de ces algorithmes, et de comparer les différences majeures. Nous en avons choisi quatre, présents sous forme de logiciels, et représentant chacune des classes :

1. RepeatMasker [Smit et al.,] pour l'alignement de séquence,
2. Mreps [Kolpakov et al., 2003] pour la méthode combinatoire,
3. TRF [Benson, 1999] pour la méthode statistique, et enfin
4. STAR [Delgrange and Rivals, 2004] pour la méthode par compression.

Chacun de ces logiciels présente des paramétrages, des contraintes et des formats de sortie spécifiques rendant les comparaisons inter-algorithmes plus ou moins évidentes. Ces comparaisons sont basées sur les critères propres aux microsatellites à savoir leur taille, leur degré de perfection (pourcentage de mutation), et leur motif (ou classe de motif).

Les résultats, pour le chromosome X humain, montrent une réelle disparité de détection, certains algorithmes privilégiant la quantité de microatellites détectés et d'autres plutôt un fort degré de perfection. De plus, chaque programme possède des contraintes propres influençant directement les résultats (librairie de RepeatMasker par exemple), soit leur analyse (scores de pureté calculés différemment). Enfin, cette étude pose le problème de la définition même des microsatellites, et de leur caractérisation par un motif consensus.

Références

[Benson, 1999] Benson, G. (1999). Tandem repeats finder : a program to analyze dna sequences. *Nucleic Acids Res.*, 27(2) :573–80.

[Consortium, 2001] Consortium, I. H. G. S. (2001). Initial sequencing and analysis of the human genome. *Nature*, 409(6822) :860–921.

[Delgrange and Rivals, 2004] Delgrange, . and Rivals, E. (2004). Star : an algorithm to search to tandem approximate repeats. *Bioinformatics*, 20(16) :2812–20.

[Dettman and Taylor, 2004] Dettman, J. R. and Taylor, J. W. (2004). Mutation and evolution of microsatellite loci in neurospora. *Genetics*, 168(3) :1231–48.

[Dieringer and Schlötterer, 2003] Dieringer, D. and Schlötterer, C. (2003). Two distinct modes of microsatellite mutation processes : evidence from the complete genomic sequences of nine species. *Genome Res.*, 13(10) :2242–51.

[Goldstein and Schlötterer, 1999] Goldstein, D. B. and Schlötterer, C. (1999). *Microsatellites Evolution and Applications*. Oxford University Press.

[Jarne et al., 1998] Jarne, P., David, P., and Viard, D. (1998). Microsatellites, transposable elements and the x chromosome. *Mol. Biol. Evol.*, 15(1) :28–34.

[Kolpakov et al., 2003] Kolpakov, R., Bana, G., and Kucherov, G. (2003). mreps : Efficient and flexible detection of tandem repeats in dna. *Nucleic Acids Res.*, 31(13) :3672–8.

[Kruglyak et al., 1998] Kruglyak, S., Durrett, R., Schug, M., and Aquadro, C. (1998). Equilibrium distributions of microsatellite repeat length resulting from a balance between slippage events and point mutations. *Proc Natl Acad Sci U S A.*, 95(18) :10774–8.

[Rolfsmeier et al., 2000] Rolfsmeier, M. L., Dixon, M. J., and Lahue, R. S. (2000). Mismatch repair blocks expansions of interrupted trinucleotide repeats in yeast. *Mol. Cell*, 6(6) :1501–7.

[Sainudiin et al., 2004] Sainudiin, R., Durrett, R., Aquadro, C., and Nielsen, R. (2004). Microsatellite mutation models : insights from a comparison of humans and chimpanzees. *Genetics*, 168(1) :383–95.

[Smit et al.,] Smit, A., Hubley, R., and Green, P. Repeatmasker at http ://repeatmasker.org. *Nucleic Acids Res.*

10 Localisation à grande échelle de motifs nucléiques décrits par des matrices position-poids

Aude Liefooghe, Hélène Touzet et Jean-Stéphane Varré
LIFL – UMR CNRS 8022 – Université de Lille 1
{liefoogh,touzet,varre}@lifl.fr

Ce travail s'inscrit dans le cadre de la recherche de courts motifs nucléiques approchés modélisés par des *matrices position-poids*, qui décrivent le contenu informationnel d'un motif position par position. Ces matrices sont particulièrement populaires pour représenter des sites de fixation de facteurs de transcription, à l'image des banques Jaspar ([Sandelin et al., 2004]) ou Transfac ([Wingender et al., 2000]).

L'annotation des génomes fait émerger des nouveaux besoins de recherche de sites de fixation potentiels à grand échelle, que ce soit dans un but de génomique comparative ou d'extraction de sites sur-représentés pour des gènes co-régulés ([Elkon et al., 2003, Ho Sui et al., 2005]). Les programmes usuels de localisation de matrices, tels que Patser ([Hertz and Stormo, 1999]) ou Matinspector ([Quandt et al., 1995]), ne sont pas adaptés à ce type de travail. À titre de référence, l'identification de l'intégralité des matrices de vertébrés de Transfac sur le génome humain avec Patser nécessite plus de 24 heures de calcul sur un PC de bureau. Nous présentons une première tentative d'accélération du calcul en présence d'une banque de matrices. L'idée est de tirer parti du fait que les banques de matrices sont des données stables, que l'on peut pré-traiter.

Algorithme exact

De manière formelle, le problème de la recherche de matrices est défini par une séquence, un ensemble de matrices et une valeur seuil de score associée à chaque matrice. Il s'agit d'identifier pour chaque matrice toutes les occurrences donnant un score supérieur au seuil dans la séquence. L'algorithme que nous proposons repose essentiellement sur le pré-calcul des scores pour chaque motif dans une structure d'index. Au regard du nombre de colonnes des matrices (entre 9 et 30 bases en pratique), il est nécessaire de recourir à un index modulaire, organisé en sous-tables, ce qui revient à découper chaque matrice en petites sous-matrices. Cette démarche exploite en fait la propriété d'additivité du score. Le choix des tables, en terme de taille et de nombre, repose sur la banque de matrices à travers deux critères :

1. la répartition de la taille des matrices, qui n'a aucune raison d'être uniforme ;
2. l'anticipation éventuelle du résultat final à partir du résultat partiel portant sur la première table. Cette anticipation se fait sur la base des scores minimaux et maximaux associés à chaque matrice et à chaque table.

Ces observations conduisent à formuler un problème d'optimisation – trouver le nombre et la taille des tables qui minimisent le coût de calcul moyen pour une contrainte d'espace mémoire donnée sous l'hypothèse que la séquence suit un modèle de Bernoulli – qui se résout par programmation dynamique. Ainsi pour l'ensemble des matrices des banques Transfac 8.1 et Jaspar – soit 605 matrices – et une contrainte mémoire de 256 Mo, le découpage optimal conduit à quatre tables de longueur respective 9, 7, 9, et 5. En pratique, pour une séquence arbitraire de longueur 10^6, le gain en temps de calcul par rapport à Patser est alors un facteur 20 (5.8 secondes contre 137 secondes sur un PC de bureau).

Similitude entre matrices

La seconde direction de recherche que nous avons explorée est liée à la similitude entre matrices, et donc à la similitude entre les sites reconnus associés. Les matrices ne sont pas toutes indépendantes, et certaines s'organisent naturellement en groupes homogènes, que ce soit pour des raisons biochimiques intrinsèques ou historiques. La motivation pour cette analyse est double : il s'agit à la fois de faire le tri parmi les prédictions en détectant les redondances dues à la proximité

des matrices, et de chercher à exploiter cette redondance pour accéler éventuellement la localisation des sites.

Le problème de comparer des matrices a été abordé récemment ([Schones et al., 2005]), mais sans mener à la corrélation entre les prédictions de sites de fixation. Si la séquence est modélisée par un modèle de Bernoulli, nous montrons qu'il est possible de déterminer exactement la proportion de prédictions communes et de prédiction distinctes entre deux matrices. Le calcul se fait en étendant l'algorithme usuel de détermination de la P-valeur pour une matrice ([Claverie and Audic, 1996]) à un couple de matrices. Cette analyse permet également d'accélerer l'algorithme de localisation de la section précédente, en travaillant sur des groupes de matrices similaires. On obtient ainsi un algorithme approché, à grain plus ou moins fin suivant les taux maximums de faux positifs et faux négatifs tolérés. Par exemple, pour l'ensemble des matrices Transfac 8.1 et Jaspar, une P-valeur égale à e^{-5}, un taux d'erreur autorisé de 20%, une classification hiérarchique des matrices permet de mettre en évidence 57 familles de deux matrices au moins.

Références

[Claverie and Audic, 1996] Claverie, J. and Audic, S. (1996). The statistical significance of nucleotide position-weight matrix matches. *Computer Applications in the Biosciences*, 12(5) :431–439.

[Elkon et al., 2003] Elkon, R., Linhart, C., Sharan, R., Shamir, R., and Shiloh, Y. (2003). Genome-wide in silico identification of transcriptional regulators controlling the cell cycle in human cells. *Genome Research*, 13 :773–780.

[Hertz and Stormo, 1999] Hertz, G. and Stormo, D. (1999). Identifying DNA and protein patterns with statistically significant alignement of multiple sequences. *Bioinformatics*, 15(7-8) :563–77.

[Ho Sui et al., 2005] Ho Sui, S. J., Mortimer, J. R., Arenillas, D. J., Brumm, J., Walsh, C. J., Kennedy, B. P., and Wasserman, W. W. (2005). opossum : identification of over-represented transcription factor binding sites in co-expressed genes. *Nucleic Acids Res*, 33(10) :3154–64.

[Quandt et al., 1995] Quandt, K., Frech, K., Karas, H., Wingender, E., and Werner, T. (1995). Matind and Matinspector - new fast and versatile tools for detection of consensus matches in nucleotide sequence data. *Nucleic Acids Research*, 23 :4878–4884.

[Sandelin et al., 2004] Sandelin, A., Alkema, W., Engstrom, P., Wasserman, W., and Lenhard, B. (2004). JASPAR : an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Research*, Database issue :D91-4.

[Schones et al., 2005] Schones, E. D., Sumazin, P., and Zhang, M. (2005). Similarity of position frequency matrices for transcription factor binding sites. *Bioinformatics*, 21(3) :307–13.

[Wingender et al., 2000] Wingender, E., Chen, X., Hehl, R., Karas, I., Liebich, I., Matys, V., Meinhardt, T., Pruss, M., Reuter, I., and Schacherer, F. (2000). TRANSFAC : an integrated system for gene expression regulation. *Nucleic Acids Research*, 28(1) :316–319.

11 Filtre exact de sélection de longues répétitions approchées, utilisant une nouvelle structure de donnée : le tableau des bi-facteurs

Pierre Peterlongo[1], Nadia Pisanti, Frédéric Boyer[2], Marie-France Sagot[2]
[1]Institut Gaspard-Monge, Université de Marne-la-Vallée, 77454 Marne-la-Vallée Cedex 2
[2]Inria Rhône-Alpes, Projet HELIX - UMR 5558 Biométrie et Biologie Évolutive, Université Claude Bernard, Lyon I, 69622 Villeurbanne cedex
pierre.peterlongo@univ-mlv.fr, Marie-France.Sagot@inria.fr

La recherche de similarités dans les textes, notamment dans les séquences biologiques reçoit depuis quelques années une attention particulière. De nombreuses techniques de filtrage et d'indexation ont été mises au point afin d'accélérer la résolution de ce problème. Cependant les filtres précédents ont étés créés dans le but d'accélérer le ¡¡*pattern matching*¿¿ ou bien la recherche de motifs répétés deux fois dans un texte. À notre connaissance, il n'existe pas à l'heure actuelle de filtre pouvant être appliqué à la recherche de répétitions apparaissant plus de deux fois dans un texte. En réponse à cette absence, nous proposons un nouveau filtre, appelé Nimbus permettant de filtrer des données afin d'en extraire des sous parties répéteés au moins r fois avec $r \geq 2$ ou des sous parties apparaissant dans au moins r séquences d'un ensemble de séquences prises en paramètre.

Nimbus utilise le concept de graines espacées, indexées dans une nouvelle structure de donnée construite en temps et espace linéaires. Cette structure est appelée le tableau des bi-facteurs. Ce travail apporte également une nouvelle contribution en ce qui concerne la condition nécessaire appliquée pour le filtrage multiple.

Les résultats expérimentaux montrent que ce filtre peut être très performant. Par exemple, en filtrant avec Nimbus un ensemble de séquences d'ADN de 5 Mb dans lesquelles nous cherchions des éléments fonctionnels en utilisant un outil d'alignement multiple comme GLAM [Frith et al., 2004], le temps d'exécution total a été réduit de 10 heures à 6 minutes tout en conduisant aux mêmes résultats.

Références

[Frith et al., 2004] Frith, M. C., Hansen, U., Spouge, J. L., and Weng, Z. (2004). Finding functional sequence elements by multiple local alignment. *Nucleic Acids Res.*, 32.

12 Des arbres de suffixes aux vecteurs de suffixes

Élise Prieur Thierry Lecroq
ABISS, Université de Rouen, 76821 Mont-Saint-Aignan.
{elise.prieur,thierry.lecroq}@univ-rouen.fr

Introduction

Un vecteur de suffixes d'un mot y constitue une structure de données alternative à un arbre de suffixes de ce mot (noté $\mathcal{A}_C(Suff(y))$). Il permet de stocker les mêmes informations dans un espace moindre. Les vecteurs de suffixes ont été introduits par Monostori(voir [Monostori, 2002]). Le vecteur de suffixes du mot y consiste en une suite de boîtes placées à certaines positions sur le mot y (voir Figure 12.1). Ces boîtes contiennent les mêmes informations que les nœuds de l'arbre de suffixes de y. Nous décrivons ici deux algorithmes linéaires de conversion d'un arbre de suffixes à un vecteur non compact et inversement. L'intérêt des vecteurs de suffixes, outre leur taille réduite, réside dans la lecture linéaire de la structure ce qui permet de détecter plus facilement les répétitions que dans les arbres de suffixes. Afin d'exploiter cette propriété, nous montrons comment augmenter les boîtes des vecteurs d'un compteur pour calculer les répétitions.

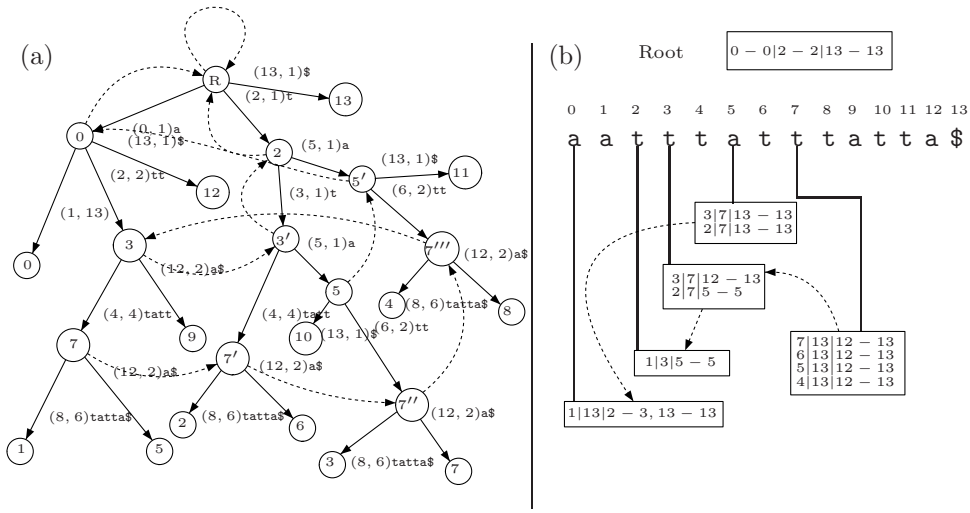


Fig. 12.1 – (a) Arbre de suffixes de `aatttatttatta$`. Les branches sont étiquetées par des couples (*position*, *longueur*). (b) Vecteur non compact de suffixes de `aatttatttatta$`. Les liens suffixes sont indiqués en pointillé. Un vecteur peut être compacté lorsque les transitions des lignes d'une boîte sont incluses dans les transitions d'une autre ligne d'une même boîte. On crée alors une boîte réduite.

De l'arbre au vecteur

Soit $u \in \mathcal{A}_C(Suff(y))$ un nœud de l'arbre des suffixes du mot y. On note $\textsc{Cible}(p, a)$, s'il existe, l'état q pour lequel a est la première lettre de l'étiquette de la branche de p à q. Soit $j = posd(u, y)$ la position droite de la première occurrence de u dans y. Alors dans le vecteur de suffixes de y, il y a une boîte B_j en position j. Dans B_j il existe une ligne h telle que $B_j[h, 0] = |u|$, le chemin naturel $B_j[h, 1]$ vaut $posd(\textsc{Cible}(u, y[j + 1]), y)$ et $\forall a \in A \setminus \{y[j + 1]\}$ telle que $\textsc{Cible}(u, a)$ est définie on a une transition $L[i] \in B_j[h, 2]$ telle que $L[i].f = posd(\textsc{Cible}(u, a), y)$ et $L[i].d = L[i].f - |\textsc{Cible}(u, a)| + 1 + |u|$ (voir Fig. 12.1). Le traitement de la racine peut se faire de manière analogue.

Du vecteur à l'arbre

Une méthode réciproque permet de convertir chaque ligne du vecteur de suffixes d'un mot y en un nœud de l'arbre des suffixes de ce mot. Les branches sortant du nœud correspondent au chemin naturel et aux transitions de la ligne.

Comptage répétitions

À chaque ligne d'une boîte correspond un nœud u de l'arbre pour lequel on pose $nbOcc(u)$ son nombre d'occurrences. Celui-ci est égal au nombre de feuilles du sous-arbre de racine u. Par un algorithme linéaire, nous complétons chaque ligne du vecteur de suffixes avec $nbOcc(u)$. Nous utilisons ensuite $nbOcc(u)$ pour compter les occurrences des facteurs de y d'une longueur donnée.

Implantation

Pour économiser l'espace mémoire, Monostori enregistre certaines données (nombre de nœuds, profondeur, ...) sur 1 ou 4 octets en utilisant un bit d'indication pour le nombre d'octets nécessaires. Afin d'améliorer cette implantation, nous proposons d'utiliser 2 bits indicateurs pour enregistrer ces données sur 1, 2, 3 ou 4 octets.

Perspectives

Nous travaillons sur un algorithme direct de construction des vecteurs compacts de suffixes afin de traiter de longues séquences. Ceci associé à l'outil de comptage permettrait de détecter efficacement des répétitions (notamment les éléments transposables) dans les séquences biologiques.

Références

[Monostori, 2002] Monostori, K. (2002). *Efficient Computational Approach to Identifying Overlapping Documents in Large Digital Collections.* PhD thesis, Monash University.

13 Nouvelle approximation de Poisson composée pour le comptage d'une famille de mots rare dans une séquence markovienne

Etienne Roquain
INRA-MIG 78352 Jouy en Josas,
ETIENNE.ROQUAIN@JOUY.INRA.FR

La recherche de mots significativement sous- ou sur-représentés dans une séquence d'ADN permet de mettre en évidence des motifs potentiellement associés à une fonction biologique. Il suffit pour cela de déterminer la fréquence attendue d'un mot donné sous un modèle de séquence et de la comparer à la fréquence observée dans la séquence. Dans le cas de mots rares (c'est-à-dire de fréquence faible), le comptage peut s'approcher par une loi de Poisson composée ([Schbath, 1995]) tandis que l'approximation gaussienne est adaptée pour des mots fréquents ([Prum et al., 1995]).

Souvent, un motif fonctionnel n'est pas un mot fixe mais correspond à un mot dégénéré (certaines lettres peuvent être ambiguës) ou à une famille de mots $\mathcal{W}$. Par exemple, chez la bactérie *H. influenzae*, la famille de mots $\mathcal{W} = \{\texttt{GATGGTGG,GCTGGTGG,GGTGGTGG,GTTGGTGG}\}$ joue un rôle très important : elle protège le génome de la bactérie d'une enzyme qui dégrade l'ADN double brin.

Pour approcher la loi du comptage d'une famille de mots rares dans une séquence markovienne, [Reinert and Schbath, 1998] ont exhibé une loi de Poisson composée. Cependant, dès que la famille $\mathcal{W}$ est recouvrante, c'est-à-dire dès les occurrences de $\mathcal{W}$ peuvent se recouvrir dans la séquence, cette approximation n'est pas satisfaisante. En outre, plus ces recouvrements sont importants c'est-à-dire plus il y a de lettres chevauchantes entres les occurrences de la famille, plus cette approximation est mauvaise. Nous proposons ici une nouvelle loi de Poisson composée qui approche le comptage d'une famille quelconque de mots. L'idée est de traiter le recouvrement entre les mots de $\mathcal{W}$ en remarquant que les occurrences des mots de $\mathcal{W}$ arrivent par "paquets" le long de la séquence appelés "trains multicolores" dans lesquels chaque couleur correspond à un mot de $\mathcal{W}$. La méthode de Chen-Stein fournit une borne théorique pour cette nouvelle approximation, et on vérifie que l'erreur tend bien vers 0 pour les familles de mots rares. Nous montrerons que la nouvelle approximation est meilleure que l'ancienne en utilisant la loi exacte du comptage (accessible dans certains cas).

Cette nouvelle approximation est implémentée dans le logiciel R'MES (disponible à l'adresse web : *http ://genome.jouy.inra.fr/ssb/rmes*). On verra sur des exemples issus de la biologie comment cette nouvelle loi permet de détecter des motifs fonctionnels dans des génomes.

Références

[Prum et al., 1995] Prum, B., Rodolphe, F., and Turckheim, É. (1995). Finding words with unexpected frequencies in DNA sequences. *J. R. Statist. Soc. B*, 57 :205–220.

[Reinert and Schbath, 1998] Reinert, G. and Schbath, S. (1998). Compound Poisson and Poisson process approximations for occurrences of multiple words in Markov chains. *J. Comp. Biol.*, 5 :223–253.

[Schbath, 1995] Schbath, S. (1995). Compound poisson approximation of word counts in DNA sequences. *ESAIM : Probability and Statistics*, 1 :1–16.

14 Approximation de Poisson pour le nombre de répétitions dans une séquence markovienne

Narjiss Touyar[1], **Sophie Schbath**[2]
[1]ABISS, Université de Rouen,
[2]INRA, Unité Mathématique, Informatique et Génome, Jouy-en-Josas.
narjiss.touyar@univ-rouen.fr, schbath@jouy.inra.fr

L'étude des répétitions dans les génomes fait partie des problèmes importants de l'analyse des séquences et de nombreux algorithmes ont été développés pour rechercher ces répétitions de façon automatique. Néanmoins, pour distinguer les répétitions dues au hasard de celles qui semblent exceptionnelles, une analyse statistique de leur significativité est nécessaire.

Pour cela, nous avons étudié la loi du nombre de répétitions de longueur (au moins) t donnée dans une chaîne de Markov. L'avantage d'un modèle markovien réside dans la possibilité de tenir compte de la composition de la séquence analysée en oligonucléotides courts. Jusqu'ici seul un résultat existait dans le cas d'une suite de lettres indépendantes [Arratia *et al.* (1996)]. En utilisant la méthode de Chen-Stein [Arratia *et al.* (1989)], nous avons démontré que le nombre de répétitions pouvait s'approcher par une variable aléatoire de Poisson pourvu que t soit de l'ordre de $\log(n)$, n étant la longueur de la séquence. Nous donnons une formule analytique du paramètre de cette loi de Poisson limite qui dépend de n, de t, de l'ordre et la matrice de transition de la chaîne de Markov [Touyar *et al.* (2005)]. Cette loi de Poisson permet ainsi d'approcher la p-value pour un nombre observé de répétitions dans une séquence donnée et ainsi détecter la taille des répétitions significativement fréquentes dans un génome.

Références

[Arratia *et al.* (1989)] Arratia, R., Goldstein, L. and Gordon, L. (1989). Two moments suffice for Poisson approximations : the Chen-Stein method. *Ann. Prob.* **17** 9–25.

[Arratia *et al.* (1996)] Arratia, R., Martin, D., Reinert, G. and Waterman, M. (1996). Poisson process approximation for sequence repeats and sequencing by hybridization. *J. Comp. Biol.* **3 (3)** 425–463.

[Touyar *et al.* (2005)] Touyar, N., Schbath, S., Cellier, D. and Dauchel, H. (2005). Poisson approximation for the number of repeats in a Markov chain model. *Soumis.*