Proceedings of the SeqBio 2015 workshop: String algorithms for bioinformatics

Alain Denise, Olivier Lespinet, Mireille Régnier

Algorithmique des séquences pour la bioinformatique
String algorithms for bioinformatics

Orsay, France

26-27 Novembre, 2015

Journées du groupe de travail COMATEGE

Editeurs

Alain Denise
Olivier Lespinet
Mireille Régnier

Site de l'évènement : http://www.gdr-bim.cnrs.fr/seqbio2015/

Attributions :
Couverture : Yann Ponty & Loic Paulevé
Crédits photo : Ioana Manolescu & étudiants du master BIBS
Édition et modèles LaTeX : Yann Ponty avec l'aide de la classe LaTeX `confproc`

Actes du workshop pluridisciplinaire SeqBio 2015
Orsay, France

Alain Denise, LRI & I2BC, Université Paris-Sud
Olivier Lespinet, I2BC, Université Paris-Sud
Mireille Regnier, Inria Saclay & LIX, Ecole Polytechnique

Novembre 26th, 2015

Préface

Ces journées SeqBio 2015 sont les seizièmes du nom. Journées nationales annuelles du groupe de travail COMATEGE commun au GdR Informatique-Mathématique et au GdR de Bioinformatique Moléculaire, elles rassemblent traditionnellement la communauté des informaticiens, bioinformaticiens et biologistes qui travaillent sur l'analyse des séquences, des aspects les plus théoriques à l'application à l'analyse des génomes.

Cette année elles sont organisées conjointement par trois équipes de l'Université Paris-Saclay : l'équipe Bioinformatique du LRI (Université Paris-Sud, CNRS), l'équipe AMIB du LIX (Ecole Polytechnique, CNRS, INRIA) et l'équipe Bioinformatique Moléculaire de l'I2BC (Université Paris-Sud, CNRS, CEA).

Nous avons le plaisir d'accueillir trois orateurs invités : Can Alkan (Université de Bilkent, Turquie), Morgane Tomas-Chollier (École Normale Supérieure de Paris) et Mark Daniel Ward (Université de Purdue, États-Unis). Les trois conférences invitées et les dix-sept exposés contributifs balaient tout le spectre scientifique de la communauté : combinatoire des mots, algorithmique des séquences et des structures, problématiques du séquençage à haut débit, génomique et évolution.

Nous remercions tout d'abord les responsables du groupe de travail COMATEGE, Thierry Lecroq et Irena Rusu, pour la confiance qu'ils nous ont accordée lors de la préparation de ces journées. Nous remercions aussi tous les organismes qui ont contribué au financement et à l'organisation de ces journées : le GdR Informatique-Mathématique du CNRS, le GdR de Bioinformatique Moléculaire du CNRS, l'Université Paris-Sud, Inria Saclay, l'Ecole Polytechnique, l'I2BC, le LIX, le LRI, et la COMUE Université Paris-Saclay qui fédère tous ces établissements. Nous remercions également les membres du comité de programme et ceux de l'équipe organisatrice, avec une mention spéciale pour Loïc Paulevé, Adeline Pierrot et Yann Ponty qui ont beaucoup donné de leur temps. Nous remercions enfin tous les orateurs et les participants, qui par leur nombre montrent que la communauté est active et dynamique.

Nous tenons à dédier ces journées à notre collègue et ami Bernard Prum qui nous a quittés le 21 octobre dernier, à l'âge de 69 ans. Bernard Prum était l'un des pionniers et l'un des piliers de l'analyse statistique des séquences génomiques, il a été parmi ceux qui ont grandement contribué à la structuration et à l'animation de la communauté de bioinformatique moléculaire. Bernard était aussi une personne d'une humanité rare.

November 22nd, 2015
Orsay, France

Alain Denise
Olivier Lespinet
Mireille Régnier

Sponsors

Groupement De Recherche Informatique Mathématique (GDR-IM)
CNRS Mathematical Computer Science research group

Groupement De Recherche BioInformatique Moléculaire (GDR-BIM)
CNRS Molecular Bioinformatics research group

Université Paris-Sud

Institut National de Recherche en Informatique & Automatique (Inria Saclay)
French Institute for Research in Computer Science and Automation

Ecole Polytechnique

Institut de Biologie Intégrative de la Cellule (I2BC)

Laboratoire d'Informatique de l'École Polytechnique (LIX)

Laboratoire de Recherche en Informatique (LRI)

COMUE Université Paris-Saclay

Comité de programme

Guillaume Blin	Université de Bordeaux, LaBRI
Laurent Brehelin	CNRS, LIRMM, Montpellier
Julien Clement	CNRS, GREYC, Université de Caen
Alain Denise	Université Paris-Sud, LRI et I2BC (co-chair)
Gabriele Fici	Université de Palerme
Christine Gaspin	INRA Toulouse
Gregory Kucherov	CNRS, LIGM, Marne-la-Vallée
Vincent Lacroix	Université Claude Bernard, LBBE, Lyon
Thierry Lecroq	Université de Rouen, LITIS
Olivier Lespinet	Université Paris-Sud, I2BC (co-chair)
Loïc Paulevé	CNRS, LRI
Pierre Peterlongo	Inria, Rennes
Adeline Pierrot	Université Paris-Sud, LRI
Yann Ponty	CNRS, Inria, LIX, École Polytechnique
Mireille Regnier	Inria, LIX, École Polytechnique (co-chair)
Éric Rivals	CNRS, LIRMM, Montpellier
Irena Rusu	Université de Nantes, LINA
Hélène Touzet	CNRS, Inria, CRIStAL, Lille

Comité d'organisation

Samer Abboud	I2BC, Université Paris-Sud
Alain Denise	LRI et I2BC, Université Paris-Sud
Christine Drevet	I2BC, Université Paris-Sud
Alice Héliou	LIX, École Polytechnique
Vincent Le Gallic	LRI, Université Paris-Sud
Olivier Lespinet	I2BC, Université Paris-Sud
Loïc Paulevé	LRI, Université Paris-Sud
Adeline Pierrot	LRI, Université Paris-Sud
Yann Ponty	LIX, École Polytechnique
Evelyne Rayssac	LIX, École Polytechnique
Mireille Regnier	LIX, École Polytechnique
Wei Wang	LRI, Université Paris-Sud
Alexandra Zaharia	LRI, Université Paris-Sud

Programme des journées

Keynote

Analyses of cis-regulatory elements from ChIP-seq and ChIP-exo experiments

Morgane Thomas-Chollier[1]

[1] *Département de Biologie, École Normale Supérieure Paris, France*

Abstract

The advent of high-throughput experimental methods have revolutionized the functional genomics field, now allowing for profiling transcription factor binding at the level of the genome. The widely-adopted ChIP-seq technique, and the high-resolution ChIP-exo approach, still necessitate computational analyses to precisely identify sequences responsible for recruitment of transcription factors to individual loci. We will present the RSAT suite (`http://rsat.eu`) and ExoProfiler (`http://github.com/ComputationalSystemsBiology/ExoProfiler`) tools to discover motifs and identify regulatory elements from ChIP-seq and ChIP-exo data.

Bio

Associate Professor in Bioinformatics at the Ecole Normale Supérieure (ENS) in Paris, Morgane Thomas-Chollier has obtained her Ph.D in 2008 as a joint work between the Université Libre de Bruxelles and the Vrije Universiteit Brussels in Belgium, under the joint direction of Luc Leyns and Jacques van Helden. MTC then worked as a postdoc in Brussels for a year, followed by three years in Germany (MPI MG Martin Vingron, Berlin, 2009-2012) thanks to an Alexander von Humboldt fellowship. MTC has joined the lab of Denis Thieffry in 2012, to reinforce the bioinformatics section of the Computational Systems Biology group at the Institute of Biology of the ENS (IBENS). MTC research interests mainly involve regulation of transcription, high-throughput functional genomics and evolution/development of metazoans. She is specialized in the detection of binding regions for transcription factors (motif detection, de-novo motif discovery, ChIP-seq, ChIP-exo) and is actively developing new computational methods and software, in particular within the RSAT suite (http://rsat.eu) in collaboration with Jacques van Helden. She has several collaborations with experimental groups, focusing on the glucocorticoid receptor, osteoporosis and evolution/development in sea urchin.

Abstract

Assemblability in RNA-seq data: Flagging complex regions in de Bruijn graphs

V. Lacroix [1,2], L. Lima[1,2], H. Lopez-Maestre[1,2], M.-F. Sagot[1,2], B. Sinaimeri[1,2]*

[1] *Université de Lyon, F-69000 Lyon; Université Lyon 1; CNRS, UMR5558, LBBE, F-69622 Villeurbanne, France*
[2] *INRIA Grenoble Rhône-Alpes, 38330 Montbonnot Saint-Martin, France*
***Corresponding author**: blerina.sinaimeri@inria.fr

Abstract

De novo assembly of RNA-seq data consists in reconstructing full-length transcripts from short sequenced fragments called reads. This problem is challenging because two transcripts, even from different genes, may very well share subsequences that are longer than the sequenced reads. This happens in the case of paralogous genes (which arose from a common ancestor through duplication). This also happens when genes host transposable elements (or other types of repeats) within their introns. Although introns are not supposed to be sequenced in RNA-seq, a fraction remains and may therefore cause major problems in transcriptome assembly.

The issue of efficiently dealing with repeats in de novo genome assembly is a long standing problem in the field. Most approaches use the fact that repeated regions will be covered by more reads. This assumption does not hold in transcriptome assembly, since coverage mostly reflects expression levels.

Most transcriptome assemblers are based on de Bruijn graphs and have no clear and explicit model for repeats in RNA-seq data, relying instead on heuristics to deal with them. Within these complex parts of the graph generated by repeats, any assembler will have to choose the "right" path(s) among the many present. This choice is not simple and often may lead to chimeric solutions. It is hence important to be able to identify transcripts coming from such complex regions in order to know that the solution presented is not the only one and furthermore may not be the right one.

In this work, we introduce a measure for flagging parts (subgraphs) in a de Bruijn graph that are "difficult" to assemble (we call them regions of *low assemblability*). We do not relay either on a reference genome or a repeat database.

We show on simulated data that this measure is relevant for identifying regions of low assemblability. We will also present some preliminary results on real data and discuss future directions.

Abstract

kissDE : a replicate-wise and annotation-free R package for testing the association between differential variants and experimental conditions in high throughput sequencing data

Camille Marchet[1]*, Lilia Brinza[5], Clara Benoit[3], Janice Kielbassa[4], Alice Julien-Laferrière[2], Frank Picard[2], Vincent Lacroix[2]

[1] *IRISA (UMR CNRS 6074), Rennes*
[2] *LBBE (UMR CNRS 5558 LBBE), Lyon*
[3] *CRCL (UMR INSERM 1052), Lyon*
[4] *Fondation Synergie Lyon Cancer, Lyon*
[5] *Bioaster, Lyon*
***Corresponding author**: camille.marchet@irisa.fr*

Abstract

As the price of short-reads technologies lowers, the sequencing of any species at an increased depth is enabled, in particular for RNA-seq data. Combined with mapping or *de novo* assembly methods, it is now possible to screen a wide range of variants of different nature such as SNPs, indels or alternative splicing isoforms among individuals and conditions. In the case of RNA-seq, the question of functional impact versus noise for variants is often raised. A significant association between a variant and a condition gives clues to point out interesting candidates for further investigation. We present a statistical method to test for the enrichment of a variant in an experimental condition. This method is implemented in a R package called kissDE. It is meant to work with any pairwise variations: two alleles of a gene, or two splice variants of a gene.

Tools already exist for the statistical analysis of splice variants across conditions. Either they rely on references (DEXSeq[1], CuffLinks[2]), which makes them not well-suited for non-model species and limited for novel variants, or modelise only single exon skipping (MATS[3] and overlook the variety of possible variant types across species. MISO[4] proposes a nice output format that helps visualising the usage of alternative exons, but does not handle replicates. For SNPs, methods rely on genotypes and do not deal with pooled data.

In our package kissDE, we take coverage information for pairwise alternative variants in two conditions or more (they can be SNPs, exon skipping, alternative donnor/acceptor, intron retention, indels...) and test whether a variant is enriched in one condition. kissDE takes replicates into account and requires at least two replicates per condition, with the advantage of not needing any annotation. Interestingly, it enables to deal with pooled samples.

kissDE was designed in particular in the scope of the local *de novo* transcriptome assembler KisSplice[5], and can be easily integrated in a pipeline with tools from KisSplice's suite. However, contingency tables provided by the user can be used as well. Counts are assumed to be distributed as a negative binomial as in standard RNAseq analysis, and we use the generalised linear models framework to build and compare two nested models with isoforms,

experimental condition and interaction in the second model as effects. We select the pairs of variants for which the interaction term has a significant effect with a likelihood ratio test. Then we output p-values and magnitudes of the effect for each pair of isoform. We are currently evaluating the performance of our method on different datasets: 1- differential splicing in cancer cell lines treated or not by retinoic acid and 2- differential abundance of alleles in subpopulations of model and non-model species.

References

[1] Alejandro Reyes, Simon Anders, and Wolfgang Huber. Analyzing RNA-seq data for differential exon usage with the DEXSeq package, 2012.

[2] Cole Trapnell, Adam Roberts, Loyal Goff, Geo Pertea, Daehwan Kim, David R Kelley, Harold Pimentel, Steven L Salzberg, John L Rinn, and Lior Pachter. Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature protocols*, 7(3):562–578, 2012.

[3] Shihao Shen, Juw Won Park, Jian Huang, Kimberly A Dittmar, Zhi-xiang Lu, Qing Zhou, Russ P Carstens, and Yi Xing. MATS: a bayesian framework for flexible detection of differential alternative splicing from RNA-seq data. *Nucleic acids research*, page gkr1291, 2012.

[4] Yarden Katz, Eric T Wang, Edoardo M Airoldi, and Christopher B Burge. Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nature methods*, 7(12):1009–1015, 2010.

[5] Gustavo AT Sacomoto, Janice Kielbassa, Rayan Chikhi, Raluca Uricaru, Pavlos Antoniou, Marie-France Sagot, Pierre Peterlongo, and Vincent Lacroix. KISSPLICE: de-novo calling alternative splicing events from RNA-seq data. *BMC bioinformatics*, 13(Suppl 6):S5, 2012.

Abstract

The Superstring Graph

Bastien Cazaux[12] and Eric Rivals[12]*

[1]*LIRMM, CNRS and Université de Montpellier, 161 rue Ada, 34095 Montpellier Cedex 5, France; Tel: (33) 4 67 41 86 64*
[2]*Institut Biologie Computationnelle, CNRS and Université de Montpellier, 860 rue Saint Priest, 34095 Montpellier Cedex 5, France*
***Corresponding author**: rivals@lirmm.fr*

Abstract

Merging words according to their overlap yields a superstring. This basic operation allows to infer long strings from a collection of short pieces, as in genome assembly. To capture a maximum of overlaps, the goal is to infer the shortest superstring of a set of input words. The Shortest Cyclic Cover of Strings (SCCS) problem asks, instead of a single linear superstring, for a set of cyclic strings that contain the words as substrings and whose sum of lengths is minimal. SCCS is used as a crucial step in polynomial time approximation algorithms for the notably hard Shortest Superstring problem, but it is solved in cubic time. The cyclic strings are then cut and merged to build a linear superstring. SCCS can also be solved by a greedy algorithm. Here, we propose a linear time algorithm for solving SCCS based on a Eulerian graph that captures all greedy solutions in linear space. Because the graph is Eulerian, this algorithm can also find a greedy solution of SCCS with the least number of cyclic strings. This has implications for solving certain instances of the Shortest linear or cyclic Superstring problems.

Abstract

Avoidability of long abelian-squares

Michaël Rao[1], Matthieu Rosenfeld[2]*

[1] *LIP, ENS de Lyon, Lyon, France*
[2] *LIP, ENS de Lyon, Lyon, France*
***Corresponding author**: matthieu.rosenfeld@ens-lyon.fr*

Abstract

The work of Thue on avoidability of repetitions in words motivated the study of avoidability of patterns in words. He contructed an infinite word over 3 letters that contains no square as a factor (a factor that can be written uu where u is non empty) and an infinite word over 2 letters that contains no cube as factor (a factor that can be written uuu). Similarly uv is an abelian square if u is a permutation of the letters of v. Answering a question asked by Erdős [1], Keränen constructed a word over 4 letters that does not contain abelian-squares [2]. A question arised naturally from that (see Mäkelä's question in [3]): Is it possible to avoid **long abelian square** over three letters?

In this talk we give a positive answer to this question. We first explain how to decide if a fixed point of a morphism is abelian-n-power-free. Then we show how this algorithm can also decide if the word obtained by applying a second morphism contains additive-powers or long-abelian-powers. In the rest of the talk we present morphisms for which we can decide and we can answer Mäkelä's question and also demonstrate the following results [4]:

- There is a infinite ternary word that does not contain any square of period greater than 5.
- It is possible to avoid additive squares over $\mathbb{Z}^2$ (with an alphabet of size 6).
- There is an infinite binary word that does not contain any 2-abelian-square of period greater than 60.

References

[1] P. Erdős. Some unsolved problems. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 6:221–254, 1961.

[2] V. Keränen. Abelian squares are avoidable on 4 letters. In *ICALP*, pages 41–52, 1992.

[3] V. Keränen. New abelian square-free DT0L-languages over 4 letters. *Manuscript*, 2003.

[4] M. Rao and M. Rosenfeld. On Mäkelä's conjectures: deciding if a morphic word avoids long abelian-powers. *Manuscript, http://arxiv.org/abs/1511.05875*, 2015.

Abstract

Fast Computation of Abelian Runs

Gabriele Fici [1], Tomasz Kociumaka [2], Thierry Lecroq [3]*, Arnaud Lefebvre [3], Élise Prieur-Gaston [3]

[1] *Dipartimento di Matematica e Informatica, Università di Palermo, Italy*
[1] *Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland*
[1] *Normandie Université, LITIS EA4108, NormaStic CNRS FR 3638, IRIB, Université de Rouen, 76821 Mont-Saint-Aignan Cedex, France*
***Corresponding author**: thierry.lecroq@univ-rouen.fr

Abstract

Let $\Sigma = \{a_1, a_2, \ldots, a_\sigma\}$ be a finite ordered alphabet of cardinality σ and let Σ^* be the set of finite words over Σ. We let $|w|$ denote the length of the word w. Given a word $w = w[0\,..\,n-1]$ of length $n > 0$, we write $w[i]$ for the $(i+1)$-th symbol of w and, for $0 \leq i \leq j < n$, we write $w[i\,..\,j]$ to denote a fragment of w from the $(i+1)$-th symbol to the $(j+1)$-th symbol, both included. This fragment is an occurrence of a substring $w[i]\cdots w[j]$. For $0 \leq i \leq n$, $w[i\,..\,i-1]$ denotes the empty fragment. We let $|w|_a$ denote the number of occurrences of the symbol $a \in \Sigma$ in the word w.

The *Parikh vector* of w, denoted by $\mathcal{P}_w$, counts the occurrences of each letter of Σ in w, that is, $\mathcal{P}_w = (|w|_{a_1}, \ldots, |w|_{a_\sigma})$.

Definition 1 (Abelian period [1]) *A factorization $w = u_0 u_1 \cdots u_{k-1} u_k$ satisfying $k \geq 1$, $\mathcal{P}_{u_1} = \cdots = \mathcal{P}_{u_{k-1}} = \mathcal{P}$, and $\mathcal{P}_{u_0} \subset \mathcal{P} \supset \mathcal{P}_{u_k}$ is called a* periodic factorization *of w with respect to $\mathcal{P}$. If a word w admits such a factorization, we say that $\mathcal{P}$ is an* (abelian) period *of w. If $k \geq 3$, we also say that w is* periodic *with period $\mathcal{P}$.*

Definition 2 (Abelian run) *A fragment $w[i\,..\,j]$ is called an* abelian run *with period $\mathcal{P}$ if it is periodic with period $\mathcal{P}$ and maximal with respect to this property (i.e., each of $w[i-1\,..\,j]$ and $w[i\,..\,j+1]$ does not exist or is not periodic with period $\mathcal{P}$).*

Matsuda et al. [2] recently presented an offline algorithm for computing all abelian runs of a word of length n in $O(n^2)$ time. Notice that, however, the definition of abelian run in [2] is slightly different from the one we consider here. Basically, our notion of abelian run is more general than the one of [2].

Given a word w of length n over an alphabet of cardinality σ and a Parikh vector $\mathcal{P}$, an abelian run of period $\mathcal{P}$ in w is a maximal occurrence of a substring of w having abelian period $\mathcal{P}$. The norm of a Parikh vector $\mathcal{P}$ is the sum of its components. We first give an online algorithm that finds all the abelian runs of period $\mathcal{P}$ in w in time $O(n)$ and space $O(\sigma + |\mathcal{P}|)$, for any given Parikh vector $\mathcal{P}$. We then present an online algorithm that computes all the abelian runs with periods of norm p in w in time $O(np)$, for any given norm p. Finally we give an $O(n^2)$ (resp. $O(n^2 \log n)$) -time offline randomized (resp. deterministic) algorithm for computing all the abelian runs of w.

These results improve those published in [3]

References

[1] Sorin Constantinescu and Lucian Ilie. Fine and Wilf's theorem for abelian periods. *Bulletin of the European Association for Theoretical Computer Science*, 89:167–170, 2006.

[2] Shohei Matsuda, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Computing abelian covers and abelian runs. In Jan Holub and Jan Zdárek, editors, *Prague Stringology Conference, PSC 2014*, pages 43–51. Czech Technical University in Prague, 2014.

[3] Gabriele Fici, Thierry Lecroq, Arnaud Lefebvre, and Élise Prieur-Gaston. Online computation of abelian runs. In Adrian Horia Dediu, Enrico Formenti, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings*, volume 8977 of *Lecture Notes in Computer Science*, pages 391–401. Springer, 2015.

Abstract

MixTaR: de novo tandem repeat detection using short and long reads

Guillaume Fertin, Géraldine Jean, Andreea Radulescu* and Irena Rusu

LINA UMR CNRS 6241, University of Nantes, Nantes, France
***Corresponding author**: andreea.radulescu@univ-nantes.fr

Abstract

Tandem repeats are parts of a genome sequence made of several copies of the same pattern, which are located next to each other. Present in the genome of most organisms, they play an important role in genome evolution and gene expression [1–4]. A significant number of tools were developed for tandem repeat detection on reference sequences. However, the tandem repeat detection remains a difficult problem in a de novo context (*i.e.* without a reference sequence) [5]. One of the main reasons is that the short reads obtained with the second-generation sequencing methods are not long enough to span regions that contain long repeats. The length limitation was tackled by the long reads obtained with the third-generation sequencing platforms such as Pacific Biosciences technologies [6]. Unfortunately, the read length expansion came with a significant increase of the error rate and the main objective of most of the research work on long reads is to handle their high error rate, up to 16% [7].

Our hybrid algorithm MixTaR [8] is the first method that combines the high-quality of short reads and the significant size of long reads in order to detect tandem repeats in a de novo context. Based on a de Bruijn graph, MixTaR has three main steps. The first one consists in detecting potential tandem repeat patterns by analysing the de Bruijn graph built from the short reads. Due to the fragmentation of the reads into k-mers, tandem repeats can form cycles in the de Bruijn graph. Thus, we detect and analyse the cycles in the graph to identify tandem repeat patterns. However, at this step we can only find potential patterns because of the reduced size of the short reads. The patterns are then verified using the long reads throughout the second step. A potential tandem repeat pattern is validated if at least two approximate copies of the pattern are identified in at least one long read. Obtaining the exact tandem repeat sequences directly from the long reads is a difficult problem since the mean error rate of the long reads is very high. Therefore, we use again the set of short reads. In the third and last step we obtain the tandem repeat sequences by computing local greedy assemblies of a selected set of short reads.

For a complete analysis of its robustness to read errors we tested MixTaR with both simulated and real reads with different error rates. The obtained results emphasize the high precision and sensibility of our method. MixTaR is able to accurately identify a significant number of tandem repeats with pattern lengths varying within a significant interval.

References

[1] Jerzy Jurka, Vladimir V. Kapitonov, Oleksiy Kohany, and Michael V Jurka. Repetitive sequences in complex genomes: structure and evolution. *Annual Reviews of Genomics Human Genetics*, 8:241–259, 2007.

[2] Subbaya Subramanian, Rakesh K. Mishra, and Lalji Singh. Genome-wide analysis of microsatellite repeats in humans: their abundance and density in specific genomic regions. *Genome Biology*, 4(2):R13, 2003.

[3] Christoph Mayer, Florian Leese, and Ralph Tollrian. Genome-wide analysis of tandem repeats in Daphnia pulex-a comparative approach. *BMC Genomics*, 11(1):277, 2010.

[4] Zhixin Zhao, Cheng Guo, Sreeskandarajan Sutharzan, Pei Li, Craig S. Echt, and Jie Zhang. Genome-Wide Analysis of Tandem Repeats in Plants and Green Algae. *G3: Genes— Genomes— Genetics*, 4(1):67–78, 2014.

[5] Todd J. Treangen and Steven L. Salzberg. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics*, 13(1):36–46, 2012.

[6] Mauricio O. Carneiro, Carsten Russ, Michael G. Ross, Stacey B. Gabriel, Chad Nusbaum, and Mark A DePristo. Pacific biosciences sequencing technology for genotyping and variation discovery in human data. *BMC Genomics*, 13(1):375, 2012.

[7] Mark J. Chaisson and Glenn Tesler. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics*, 13(1):238, 2012.

[8] Guillaume Fertin, Géraldine Jean, Andreea Radulescu, and Irena Rusu. Hybrid de novo tandem repeat detection using short and long reads. *BMC Medical Genomics*, 8(Suppl 3):S5, 2015.

Abstract

Smiles2Monomers: a link between atomic and monomeric structures of polymers

Yoann Dufresne[1,2]*, Laurent Noé[1,2], Valérie Leclère[1,2,3], Maude Pupin[1,2]

[1]Univ. Lille, CNRS, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, Lille, France
[2]Inria Lille Nord Europe, Bonsai team, Villeneuve d'Ascq, France
[3]Univ. Lille, EA 7394 - ICV - Institut Charles Viollette, Lille, France
Corresponding author: yoann.dufresne@ed.univ-lille1.fr

Abstract

Chemists (pharmacists) and biologists study natural compounds because they are huge sources of new active molecules. Both types of analysis, chemical and biological, are complementary and help in improving the knowledge on these compounds. So, we have designed an efficient and accurate algorithm, implemented in the tool `Smiles2Monomers` (`s2m`), to infer the monomeric structure of a polymer from its chemical structure (`http://bioinfo.lifl.fr/norine/smiles2monomers.jsp`). Our algorithm identifies the monomers occurring in a target polymer based on a database of monomers. We design dedicated algorithms and data representation for the two steps of `s2m`. In the first step, monomers are mapped on the atomic structure of the polymer by a subgraph-isomorphism algorithm. The mapping is efficient thanks to a Markovian index built by a dynamic programming algorithm. In the second step, the best tiling is computed so that non-overlapping monomers cover all the structure of the target polymer. A greedy algorithm combines the mapped monomers into a consistent monomeric structure. If necessary, a local branch and cut algorithm refines the structure.

`s2m` was tested on 705 polymers extracted from two manually annotated databases (Norine and the Chemical Component Dictionary). It reached a recall of 92% and a precision of 83%. The average computation time per polymer is 2 s.

Abstract

Reference-free compression of high throughput sequencing data with a probabilistic de Bruijn graph

Gaëtan Benoit[1]*, Claire Lemaitre[1], Dominique Lavenier[1], Erwan Drezen[1], Thibault Dayris[2], Raluca Uricaru[2], Guillaume Rizk[1]

[1] *INRIA/IRISA/GenScale, Campus de Beaulieu, 35042 Rennes, France*
[2] *University of Bordeaux, CNRS/LaBRI, F-33405, Talence*
***Corresponding author**: gaetan.benoit@inria.fr

Abstract

Data volumes generated by next-generation sequencing (NGS) technologies is now a major concern for both data storage and transmission. This triggered the need for more efficient methods than general purpose compression tools, such as the widely used gzip method.

Most *de novo* methods either use a context-model to predict bases according to their context, followed by an arithmetic encoder, or re-order reads to maximize similarities between consecutive reads and therefore boost compression. However, by simply re-ordering reads read-pairing information is lost, thus many downstream analysis become impossible to run. Existing tools based on reads re-ordering, either lose the pairing information and achieve high compression, or provide an option to keep it at the cost of a much lower compression ratio.

We present a novel reference-free method meant to compress data issued from high throughput sequencing technologies, in both FASTA and FASTQ format. Our approach, implemented in the software Leon, employs techniques derived from existing assembly principles. The method is based on a reference probabilistic *de Bruijn Graph*, built *de novo* from the set of reads and stored in a Bloom filter. Each read is encoded losslessly as a path in this graph, by memorizing an anchoring kmer and a list of bifurcations. The same probabilistic *de Bruijn Graph* is used to perform a lossy transformation of the quality scores, which allows to obtain higher compression rates without losing pertinent information for downstream analyses. Leon was run on various real sequencing datasets (whole genome, exome, RNA-seq or metagenomics). In all cases, Leon showed higher overall compression ratios than state-of-the-art compression software.

On a *C. elegans* whole genome sequencing dataset, Leon divided the original file size by more than 20, corresponding to 0.67 bits/base for DNA and 0.24 bits/quality score. Leon can compress large datasets, for example a 733 GB FASTQ file (whole human genome sequenced at 102x depth) is compressed in 11h using 9.5 GB of ram and 8 CPU threads.

Abstract

Faster de Bruijn graph representation using minimal perfect hashing

Rayan Chikhi[1]

[1] *CNRS, Univ. Lille, UMR 9189 - CRIStAL, F-59000 Lille, France*
***Corresponding author**: rayan.chikhi@univ-lille1.fr*

Abstract

Many bioinformatics methods use de Bruijn graphs, notably genome assemblers. Typically, these graphs contain a huge number of nodes and need to reside entirely in memory. It is therefore critical that their data structures (i) occupy small space and (ii) answer graph traversal queries rapidly. Recent results have mostly focused on the succinctness of representations, e.g. [1, 2]. In this work, I show that at the expense of slightly more space, one can design a navigational data structure [2] for de Bruijn graphs that answers graph traversal queries much faster than current approaches.

The representation is based on a minimal perfect hash function over the set of n nodes [3], which requires $2.61n + O(1)$ bits. An additional array, indexed by the perfect hash, encodes the presence/absence of all possible in- and out-neighbors for each node in a bit string. The total size of the structure is thus $(2\Sigma + 2.61)n + O(1)$ bits, i.e. around 10.61 bits per node on a DNA alphabet.

The proposed structure is compared to cascading Bloom filters [1], a practical de Bruijn graph representation that occupies around 8.5 bits per node. Preliminary experiments show that traversal queries are answered 4x – 14x faster (resp. on nodes of lengths 21 and 121). While the proposed structure is 1.25x larger, it is an attractive alternative due to much faster query operations. This data structure could serve as a basis for ultra-fast sequence analysis tools. An implementation is available in the GATB library (`http://github.com/GATB/gatb-core`).

References

[1] Kamil Salikhov, Gustavo Sacomoto, and Gregory Kucherov. Using cascading Bloom filters to improve the memory usage for de Brujin graphs. In *Algorithms in Bioinformatics*, volume 8126 of *Lecture Notes in Computer Science*, pages 364–376. Springer Berlin Heidelberg, 2013.

[2] Rayan Chikhi, Antoine Limasset, Shaun Jackman, Jared T Simpson, and Paul Medvedev. On the representation of de Bruijn graphs. In *Research in Computational Molecular Biology*, pages 35–55. Springer, 2014.

[3] Djamal Belazzougui, Paolo Boldi, Giuseppe Ottaviano, Rossano Venturini, and Sebastiano Vigna. Cache-oblivious peeling of random hypergraphs. In *Data Compression Conference (DCC), 2014*, pages 352–361. IEEE, 2014.

Abstract

Read Mapping on de Bruijn graph

Antoine Limasset*, Pierre Peterlongo

INRIA/IRISA/GenScale, Campus de Beaulieu, 35042 Rennes Cedex, France
***Corresponding author**: antoine.limasset@inria.fr

Abstract

Mapping reads on references is a central task in numerous genomic studies and huge efforts have been invested in this field to produce very efficient tools, e.g., Bowtie [1] or BWA [2]. Nowadays, references are obtained by assemblers that organize the read information in assembly graphs, such as the de Bruijn graph or the overlap graph. Once the graph is constructed, they output consensus sequences obtained by walking the graph following heuristic algorithms (see for instance Velvet [3] algorithms). Thus, the produced sequences are biased due to heuristic choices, and are fragmented mainly because of the repeats in the sequences.

Conversely to assembled sequences, assembly graphs contain all read information. This motivates us to consider genomes represented by graphs instead of flat sequences. Such a proposal is acceptable only if we provide efficient tool for mapping reads on such structure. This is why we explore here the problem of mapping reads on a graph, and in particular on a de Bruijn graph.

We have shown that the problem of mapping sequences on a de Bruijn graph is NP-complete [4] and no scalable generic tool exists yet. We present a practical solution and its implementation called BGREAT which handles real world instances with moderate resources. BGREAT is able to map million reads per CPU hour, even on a complex human de Bruin graph. Results show that mapping on graphs enables to map up to 22% more reads than mapping on assembled sequences. This contribution opens the way for major applications in various treatments of sequencing data as de novo assembly, correction, compression, quantification and alignment on reference-less individuals.

References

[1] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[2] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[3] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[4] Antoine Limasset and Pierre Peterlongo. Read mapping on de bruijn graph. *arXiv preprint arXiv:1505.04911*, 2015.

Keynote

Discovery of large genomic inversions using pooled clone sequencing

Can Alkan[1]

[1] *Dept. Computer Engineering, Bilkent University, Ankara, Turkey*

Abstract

There are many different forms of genomic structural variation that can be broadly classified as copy number variation (CNV) and balanced rearrangements. Although many algorithms are now available in the literature that aim to characterize CNVs, discovery of balanced rearrangements (inversions and translocations) remains an open problem. This is mainly because the breakpoints of such events typically lie within segmental duplications and common repeats, which reduce the mappability of short reads. The 1000 Genomes Project spearheaded the development of several methods to identify inversions, however, they are limited to relatively short inversions, and there are currently no available algorithms to discover large inversions using high throughput sequencing technologies (HTS). Here we propose to use a sequencing method (Kitzman et al., 2011) originally developed to improve haplotype resolution to characterize large genomic inversions. This method, called pooled clone sequencing, merges the advantages of clone based sequencing approach with the speed and cost efficiency of HTS technologies. Using data generated with pooled clone sequencing method, we developed a novel algorithm, dipSeq, to discover large inversions (¿500 Kbp). We show the power of dipSeq first on simulated data, and then apply it to the genome of a HapMap individual (NA12878).We were able to accurately discover all previously known and experimentally validated large inversions in the same genome. We also identified a novel inversion, and confirmed using fluorescent in situ hybridization.

Bio

Can Alkan is currently an Assistant Professor at the Department of Computer Engineering at Bilkent University since January 2012. He graduated from Bilkent University Dept. of Computer Engineering in 2000, and received his Ph.D. in Computer Science from Case Western Reserve University in 2005 after a brief visit to Simon Fraser University. During his Ph.D. he worked on the evolution of centromeric DNA, RNA-RNA interaction prediction and RNA folding problems. He then joined the Department of Genome Sciences of the University of Washington as a postdoctoral fellow. Since then his work includes computational prediction of human genomic structural variation, and characterization of segmental duplications and copy-number polymorphisms using next generation sequencing data.

Abstract

The pan-genome of the cultivated African rice *Oryza glaberrima* and its wild ancestor *Oryza barthii*

Cécile Monat[1]*, Christine Tranchant-Dubreuil[1], François Sabot[1]

[1]*Équipe RICE, UMR DIADE, Institut de Recherche pour le Dévelopement, Montpellier, France*
***Corresponding author**: cecile.monat@ird.fr

Abstract

The diversity of a species is the sum of the diversity of all its individuals. The pan-genome is one representation of this diversity [1], and is divided in three parts: (i) the core-genome, containing all the genes presents in all the individuals; (ii) the dispensable-genome, containing all the gene missing in at least one individual; (iii) and finally the individuals-specific genome, containing the genes presents in only one individual. To study then this diversity, we would need the sequence from all the species individuals, which is almost impossible. Thus we should have the highest possible number of these sequences, that we can afford through NGS and large re-sequencing projects. In our study, we re-sequenced 120 individuals from the cultivated African rice *Oryza glaberrima* and 74 from its wild ancestor *Oryza barthii*. We then performed read mapping for all individuals to a reference genome, cross these information with annotation, and finally obtained a presence/absence gene matrix. From that matrix, we will analyse the pan-genome of the two species to understand the effect of domestication on the genome structure. Preliminary results on the first chromosome with data from *O. glaberrima* shown about 48% of the genes belong to the dispensable genome.

References

[1] H. Tettelin, V. Masignani, M. J. Cieslewicz, C. Donati, D. Medini, N. L. Ward, S. V. Angiuoli, J. Crabtree, A. L. Jones, a. S. Durkin, R. T. Deboy, T. M. Davidsen, M. Mora, M. Scarselli, I. Margarit y Ros, J. D. Peterson, C. R. Hauser, J. P. Sundaram, W. C. Nelson, R. Madupu, L. M. Brinkac, R. J. Dodson, M. J. Rosovitz, S. a. Sullivan, S. C. Daugherty, D. H. Haft, J. Selengut, M. L. Gwinn, L. Zhou, N. Zafar, H. Khouri, D. Radune, G. Dimitrov, K. Watkins, K. J. B. O'Connor, S. Smith, T. R. Utterback, O. White, C. E. Rubens, G. Grandi, L. C. Madoff, D. L. Kasper, J. L. Telford, M. R. Wessels, R. Rappuoli, and C. M. Fraser, "Genome analysis of multiple pathogenic isolates of Streptococcus agalactiae: implications for the microbial "pan-genome".," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, pp. 13950–5, sep 2005.

Abstract

Ancestral reconstruction and investigations of genomic recombination on Campanulides chloroplasts

Bashar Al-Nuaimi[1,2]*, Roxane Mallouhi[1]*, Bassam AlKindy[1,3]*, Christophe Guyeux[1]*, Michel Salomon, and Jean-François Couchot[1]*

[1] *FEMTO-ST Institute, UMR 6174 CNRS, DISC Computer Science Department, Université de Franche-Comté, France*
[2] *Department of Computer Science, University of Diyala, Iraq*
[3] *Department of Computer Science, University of Mustansiriyah, Baghdad, Iraq*
***{bashar.al-nuaimi, bassam.al-kindy, christophe.guyeux, michel.salomon, jean-francois.couchot}@univ-fcomte.fr**
***roxanemallouhy@live.com**

Abstract

Chloroplasts are one of many types of organelles in the plant cell. They are considered to have originated from cyanobacteria through endosymbiosis, when an eukaryotic cell engulfed a photosynthesizing cyanobacterium, which remained and became a permanent resident in the cell. The term of chloroplast comes from the combination of plastid and chloro, meaning that it is an organelle found in plant cells that contains the chlorophyll. Chloroplast has the ability to convert water, light energy, and carbon dioxide (CO_2) in chemical energy by using carbon-fixation cycle [1] (also called *Calvin Cycle*, the whole process being called photosynthesis). This key role explains why chloroplasts are at the basis of most trophic chains and are thus responsible for evolution and speciation. Moreover, as photosynthetic organisms release atmospheric oxygen when converting light energy in chemical one, and simultaneously produce organic molecules from carbon dioxide, they originated the breathable air and represent a mid to long term carbon storage medium.

Consequently, exploring the evolutionary history of chloroplasts is of great interest, and *we propose here to investigate it by the mean of ancestral genomes reconstruction*. This reconstruction will be achieved in order to discover how the molecules have evolved over time, at which rate, and to determine whether evidences of their cyanobacteria origin can be presented by this way. This long-term objective necessitates numerous intermediate research advances. Among other things, it supposes to be able to apply the ancestral reconstruction on a well-supported phylogenetic tree of a representative collection of well-annotated chloroplastic genomes. Indeed, sister relationship of two species must be clearly established before trying to reconstruct their ancestor.

We have already explained in previous works how to obtain accurately both annotated chloroplasts and their phylogenetic relationship [2, 3, 4]. This is why, in this proposal, we consider that the gene content and order of each species is well known, and that an accurate phylogenetic inference has already been obtained *in the specific case of chloroplast sequences*. Our objective is then to design algorithms making it possible to reach the last universal common ancestor of these chloroplasts.

Ancestral genome reconstruction has already been investigated several times in the

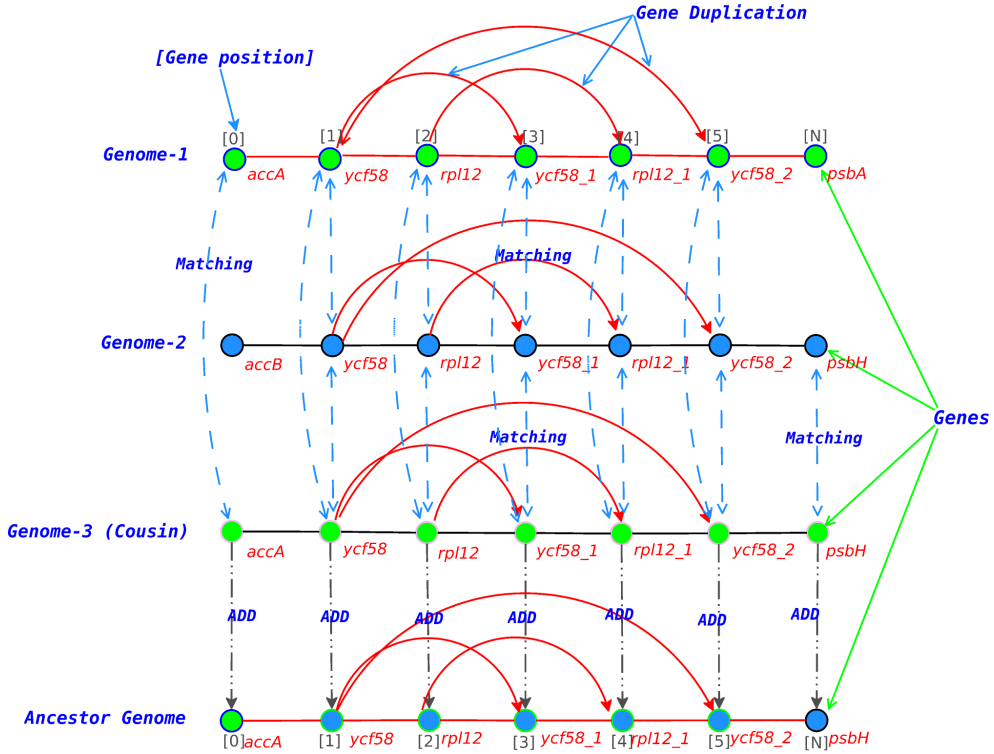

Figure 1. Example of ancestral reconstruction process between two genomes

literature [5, 6], but not in the specific case of chloroplastic genomes. Indeed it usually deals with permutations of integers: tools like Badger [7] or MLGO [8] do not support genomes of various length and with repeated/missing genes. Our problem applied to chloroplasts may appear as more difficult, as we relax the permutation hypothesis. However, in the classical Multiple Genome Rearrangement Problem [9], targeted genomes are bacterial or nucleus ones, which have much more genes than a chloroplast. Furthermore, gene order and content do not evolve so much when considering related plant species. Such observations explain why state-of-the-art algorithms cannot be applied to our particular problem, and why this latter should be solvable.

More precisely, two orders have been regarded in this research work, namely the *Apiales* and the *Asterales*. All complete genomes have been annotated using the accurate DOGMA tool [10], which is specific to such kind of genomes, and a well supported tree has been obtained based on their core genes and using RAxML. Then, after having reconstructed manually all ancestral genomes (ordered list of gene names) using a recursive naked eye comparison of each couple of sister species, we have designed dynamic programming based algorithms that have been able to recover all ancestral gene contents.

Fundamentally, the method consists in considering that all ordered sublist of genes shared by two sister species must be present too in their ancestor, while two or more cousins are used to solve conflict situations, as depicted in Figure 1.

Applying the proposed algorithms on the considered species, we found that the two studied families have not faced the same kind of genomic recombination. More precisely, *Apiales* family does not undergo insertions and deletions, while they are present in the *Asterales* case. Similarly, duplication of genes are quite different between the two considered families. Our intention, if this proposal is accepted, is to present both the algorithms and

the results obtained on *Apiales* and *Asterales*.

This proposal is an ongoing work regarding the design of ancestral reconstruction of chloroplastic genomes. We intend to continue both the theoretical investigations and its applications to the *Lamiids* clade, encompassing *Ericales*, *Solanales*, *Gentianales*, *Sapindules*, and *fabids* orders. Next steps of such research work is to reconstruct too the ancestral DNA sequences, to extend the algorithms to larger genomes (of bacteria, for instance), to apply them on larger sets of species (*e.g.*, the whole available complete genomes of chloroplasts), and to extract various knowledge from these ancestors regarding the evolution of genome sequences.

References

[1] Nigel Chaffey, B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. Molecular biology of the cell. *Annals of Botany*, 91(3):401–401, 2003.

[2] Bassam AlKindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques M Bahi. Gene similarity-based approaches for determining core-genes of chloroplasts. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 71–74. IEEE, 2014.

[3] Bassam AlKindy, Huda Al-Nayyef, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques M. Bahi. Improved core genes prediction for constructing well-supported phylogenetic trees in large sets of plant species. In Francisco Ortuño and Ignacio Rojas, editors, *Bioinformatics and Biomedical Engineering*, volume 9043 of *Lecture Notes in Computer Science*, pages 379–390. Springer International Publishing, 2015.

[4] Bassam Alkindy, Jean-François Couchot, Christophe Guyeux, Arnaud Mouly, Michel Salomon, and Jacques M. Bahi. Finding the core-genes of chloroplasts. *Journal of Bioscience, Biochemistery, and Bioinformatics*, 4(5):357–364, 2014.

[5] Mathieu Blanchette, Abdoulaye Baniré Diallo, Eric D Green, Webb Miller, and David Haussler. Computational reconstruction of ancestral dna sequences. In *Phylogenomics*, pages 171–184. Springer, 2008.

[6] Virginie Lopez Rascol, Pierre Pontarotti, and Anthony Levasseur. Ancestral animal genomes reconstruction. *Current opinion in immunology*, 19(5):542–546, 2007.

[7] Bret Larget, Donald L. Simon, Joseph B. Kadane, and Deborah Sweet. A bayesian analysis of metazoan mitochondrial genome arrangements. *Molecular Biology and Evolution*, 22(3):486–495, 2005.

[8] Fei Hu, Yu Lin, and Jijun Tang. MLGO: phylogeny reconstruction and ancestral inference from gene-order data. *BMC Bioinformatics*, 15:354, 2014.

[9] Sridhar Hannenhalli, Colombe Chappey, Eugene V Koonin, and Pavel A Pevzner. Genome sequence comparison and scenarios for gene rearrangements: A test case. *Genomics*, 30(2):299–311, 1995.

[10] Stacia K. Wyman, Robert K. Jansen, and Jeffrey L. Boore. Automatic annotation of organellar genomes with dogma. *Bioinformatics*, 20(17):3252–3255, 2004.

Abstract

FM-index of an alignment: A compressed index for highly similar strings

Joong Chae Na[1], Hyunjoon Kim[2], Heejin Park[3], Thierry Lecroq[4]*, Martine Léonard[4], Laurent Mouchard[4], Kunsoo Park[2]

[1] *Department of Computer Science and Engineering, Sejong University, Seoul 143-747, South Korea*
[2] *School of Computer Science and Engineering, Seoul National University, Seoul 151-742, South Korea*
[3] *Department of Computer Science and Engineering, Hanyang University, Seoul 133-791, South Korea*
[4] *Normandie Université, LITIS EA4108, NormaStic CNRS FR 3638, IRIB, Université de Rouen, 76821 Mont-Saint-Aignan Cedex, France*
***Corresponding author**: thierry.lecroq@univ-rouen.fr

Abstract

We introduce an alignment for a set of highly similar strings. Consider four highly similar strings: $S^1 = \texttt{atccaactccc\#}$, $S^2 = \texttt{attcaactcac\#}$, $S^3 = \texttt{atcttactcac\#}$, and $S^4 = \texttt{atacaactccc\#}$. These strings are the same except the underlined characters. Then, the string set can be compactly represented by combining common characters as follows:

$$\rho = \texttt{at}(\underline{\texttt{cca/tca/ctt/aca}})\texttt{actc}(\underline{\texttt{c/a/a/c}})\texttt{c\#},$$

in which the common characters are written once and the underlined characters are enumerated in parentheses. We call ρ the alignment for the string set. Formally, given a set of m highly similar strings $S^j = \alpha_1\Delta_1^j \cdots \alpha_r\Delta_r^j\alpha_{r+1}$ $(1 \le j \le m)$, the alignment for the string set is denoted by

$$\rho = \alpha_1(\Delta_1^1/\cdots/\Delta_1^m)\cdots\alpha_r(\Delta_r^1/\cdots/\Delta_r^m)\alpha_{r+1}$$

where α_i $(1 \le i \le r+1)$ is a common substring in all strings, and Δ_i^j $(1 \le i \le r)$ is a non-common substring in string S^j. In the example above, $\alpha_1 = \texttt{at}$, $\alpha_2 = \texttt{actc}$, $\alpha_3 = \texttt{c\#}$, $\Delta_1^1/\Delta_1^2/\Delta_1^3/\Delta_1^4 = \underline{\texttt{cca/tca/ctt/aca}}$, and $\Delta_2^1/\Delta_2^2/\Delta_2^3/\Delta_2^4 = \underline{\texttt{c/a/a/c}}$. Without loss of generality, we assume only α_1 can be empty among all the α_i's and Δ_i^j's. Furthermore, we assume that $|\Delta_i^1| = |\Delta_i^2| = \cdots = |\Delta_i^m|$ for each $1 \le i \le r$, since a special character '$-$' can be inserted in case of indels.

The suffix tree of an alignment has been proposed in [1] while the suffix array of an alignment has been presented in [2]. Here we propose the FM-index of an alignment [3], a compressed index for highly similar strings with the functionalities of pattern search and random access. For this, we first design a new and improved version of the suffix array of an alignment. We also design a version of the Burrows-Wheeler Transform adapted for an alignment. The FM-index of an alignment comprises a sampling of this suffix array of an alignment and a Burrows-Wheeler Transform of an alignment. The FM-index of an alignment supports the LF-mapping and backward search, the key functionalities of the FM-index, but the LF-mapping and backward search of our index is significantly more involved than the original FM-index. We implemented the FM-index of an alignment and did experiments on 100 genome sequences from the 1000 Genomes Project. The index size of the FM-index of an alignment is about a half of that of RLCSA (Run-Length Compressed Suffix Array) [4].

References

[1] Joong Chae Na, Heejin Park, Maxime Crochemore, Jan Holub, Costas S. Iliopoulos, Laurent Mouchard, and Kunsoo Park. Suffix tree of alignment: An efficient index for similar data. In Thierry Lecroq and Laurent Mouchard, editors, *Combinatorial Algorithms - 24th International Workshop, IWOCA 2013, Rouen, France, July 10-12, 2013, Revised Selected Papers*, volume 8288 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2013.

[2] Joong Chae Na, Heejin Park, Sunho Lee, Minsung Hong, Thierry Lecroq, Laurent Mouchard, and Kunsoo Park. Suffix array of alignment: A practical index for similar data. In Oren Kurland, Moshe Lewenstein, and Ely Porat, editors, *String Processing and Information Retrieval - 20th International Symposium, SPIRE 2013, Jerusalem, Israel, October 7-9, 2013, Proceedings*, volume 8214 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2013.

[3] Joong Chae Na, Hyunjoon Kim, Heejin Park, Thierry Lecroq, Martine Léonard, Laurent Mouchard, and Kunsoo Park. FM-index of alignment: A compressed index for similar strings. *Theoretical Computer Science*. to appear.

[4] Veli Mäkinen, Gonzalo Navarro, Jouni Sirén, and Niko Välimäki. Storage and retrieval of highly repetitive sequence collections. *Journal of Computational Biology*, 17(3):281–308, 2010.

Abstract

Minimized Compact Automaton for Clumps over Degenerate Patterns

E. Furletova[1*], J. Holub[2†] and M. Régnier[3‡]

[1]Institute of Mathematical Problems of Biology, 142290, Institutskaya, 4, Pushchino, Russia
[2]FIT, Czech Technical University in Prague, Thákurova 2700/9, Czech Republic
[3]AMIB project/team, Inria Saclay and LIX, Ecole Polytechnique, France
***Corresponding authors:** Furletova@lpm.org.ru, Jan.Holub@fit.cvut.cz and Mireille.Regnier@inria.fr

Abstract

Our study is motivated by the problem of functional fragments detection in biological sequences. Usually, methods select overrepresented occurrences of a pattern (set of specific words describing the fragments) to find potential functional fragments. The classical measures of overrepresentation of motifs are z-scores and p-values or score over probability weight matrices [Sto00]. It is tightly related to the combinatorics of clumps, sequences of overlapping occurrences of a given pattern, see [RS98] and [RFI14].

We present a minimized compacted automaton (Overlap walking automaton, *OWA*) recognizing all the possible clumps for degenerate patterns and its usage for computation of probabilities of sets of clumps. *OWA* is the minimization of the automaton presented in the paper [RFI14]. Degenerate (indeterminate) pattern [HS03] can be presented as a word in a degenerate alphabet, degenerate letter is a subset of a main alphabet Σ. A word $x = x_1, \ldots, x_m$ in Σ belongs to a degenerate pattern $\pi = \pi_1, \ldots, \pi_m$ if $x_i \in \pi_i$. An example of a degenerate pattern is an IUPAC consensus. We also present minimization of Aho-Corasick automaton *DGTrie* [AC75], recognizing all the sequences ending with pattern occurrences.

We use *DGTrie* as an auxiliary structure for *OWA* construction. The states of *DGTrie* are equivalence classes on the prefixes of words ($Pref(\pi)$) of the pattern π. The set of states of *OWA* is a subset of *DGTrie* states that correspond to overlaps between pattern words. We introduce the following equivalence relation:

Let $x, y \in Pref(\pi)$; $x \equiv y$ iff $x = y$ or the following conditions are satisfied:

$$1) \quad xt \in Pref(\pi) \Longleftrightarrow yt \in Pref(\pi), t \in \Sigma^* \ ; \tag{1}$$

$$2) \quad sl(x) \equiv sl(y) \ . \tag{2}$$

One can prove that *DGTrie* is also minimal automaton according to Nerode equivalence recognizing all the sequences ending with pattern occurrences. The introduced equivalence relation allow to construct *DGTrie* in linear time on the number of its states (it is bounded by 2^m, where m the length of pattern words). *Owa* can be constructed in linear time on the sum of its size and *DGTrie* size.

*Evgeniia Furletova was supported by the grants 14-04-32220-mol_a and 14-01-93106-NCNILa from RFBR and Metchnikov fellowship from Campus France.

†This research has been partially supported by the Czech Science Foundation (GAČR) as project No. GA13-03253S.

‡Mireille Régnier was supported by DIGITEO grant RNAomics and INRIA-CNRS grant CARNAGE

Computer experiments show that, for degenerate patterns, sizes of proposed automata are significantly smaller than the patterns sizes. For example, amino acids degenerate pattern `FLXXTXXXRXXXAXXQXXXLXXF` (symbols except of `X` stand for specific amino acids, `X` stands for any amino acid) describing protein familia GAS_VESICLE_C from the database PROSITE [SCea12] corresponds to a pattern containing about 10^{29} words. Straightforward enumeration of the words and construction of corresponding Aho-Corasick automaton are impossible. *DGTrie* for this pattern has only 1480 states and 4900 edges. And *OWA* has states 89 and 17676 edges. Efficiency of our approach increases for larger alphabets.

References

[AC75] A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[HS03] J. Holub and W. F. Smyth. Algorithms on indeterminate strings. In M. Miller and K. Park, editors, *Proceedings of the 14th Australasian Workshop On Combinatorial Algorithms*, pages 36–45. Seoul National University, Seoul, Korea, 2003.

[RFI14] M. Regnier, B. Fang, and D. Iakovishina. Clump Combinatorics, Automata, and Word Asymptotics. In Michael Drmota; Mark Ward, editor, *ANALCO' 2014, Portland, United States.*, pages 62–73, Jan 2014.

[RS98] G. Reinert and S. Schbath. Compound Poisson approximation for occurrences of multiple words in Markov chains. *Journal of Computational Biology*, 5(2):223–253, 1998.

[SCea12] C. J. A. Sigrist, E. Castro, and et al. New and continuing developments at PROSITE. *Nucleic Acids Res*, 2012.

[Sto00] G. D. Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16:16–23, 2000.

Abstract

Complexité moyenne d'algorithmes de pattern matching et optimalité

Gilles Didier[1]* et Laurent Tichit[1]

[1] *Aix-Marseille Université, CNRS, Centrale Marseille, I2M UMR7373, Marseille, France*
***Corresponding author**: gilles.didier@univ-amu.fr*

Abstract

Nous étudions la vitesse asymptotique d'algorithmes de pattern matching cherchant un motif w dans un texte aléatoire Bernoulli. La vitesse asymptotique est définie comme la limite, quand n tend vers l'infini, de l'espérance du rapport de n par le nombre d'accès en lecture effectués pour un texte aléatoire de taille n. L'étude est limitée aux algorithmes qui, à chaque itération et à partir de la position courante, lisent une position relative déterminée par leur état interne et, selon la lettre lue, changent d'état interne, incrémentent (ou pas) la position courante et, le cas échéant, signalent un match (a priori tous les algorithmes usuels peuvent être codés sous cette forme). Étant donné w, l'ordre d'un algorithme est la plus grande position, relativement à la position courante, qu'il lit en cherchant w (généralement $|w|-1$ ou $|w|$). Nous montrons que si un texte est Bernoulli alors la séquence des états internes d'un algorithme est une chaîne de Markov à états cachés dont on sait expliciter les probabilités de transition, ce qui permet d'en calculer la vitesse asymptotique.

Notre résultat principal est, qu'étant donnés un motif w, un entier $k \geq |w|-1$ et un modèle Bernoulli, il existe, parmi les algorithmes d'ordre k, un algorithme de vitesse asymptotique maximale dont l'ensemble des états internes est en bijection avec un sous-ensemble des fonctions partielles f de $\{0,\dots,n\}$ vers $\mathcal{A}$ telles que, pour tout $i < |w|$, si $f(i)$ est défini alors $f(i) = w_i$. Le nombre d'algorithmes vérifiant cette propriété étant fini, il est possible de tous les évaluer et de déterminer ainsi un algorithme optimal pour un motif, un ordre et un modèle Bernoulli donnés. La table ci-après présente les vitesses asymptotiques d'algorithmes usuels et de l'optimal à l'ordre 3 pour les fréquences 0.1/0.9 en a/b pour tous les motifs de taille 4.

	Naive	Morris-Pratt	KMP	Quicksearch	Horspool	FJS	TVSBS	EBOM	Hashq	Optimal
aaaa	0.900	0.909	1.000	2.179	3.298	1.973	1.680	1.486	0.666	**3.501**
aaab	0.900	0.911	1.000	0.521	1.765	0.528	0.270	1.367	0.663	**2.610**
aaba	0.894	0.917	0.999	0.918	0.910	0.872	1.546	1.238	0.660	**2.187**
aabb	0.894	0.923	0.998	0.564	0.382	0.534	0.288	0.703	0.635	**1.799**
abaa	0.834	0.911	0.983	1.186	1.672	1.240	1.627	1.238	0.661	**2.181**
abab	0.834	0.916	0.990	0.500	0.854	0.539	0.276	1.165	0.635	**1.805**
abba	0.787	0.916	0.981	0.919	0.926	0.866	0.816	0.627	0.613	**1.796**
abbb	0.787	0.974	0.974	0.556	0.327	0.574	0.312	0.312	0.462	**1.149**
baaa	0.500	0.527	0.527	1.193	2.496	1.487	1.332	1.367	0.662	**2.614**
baab	0.500	0.529	0.529	0.367	1.336	0.370	0.223	1.178	0.650	**1.746**
baba	0.483	0.529	0.552	0.580	0.910	0.804	1.158	1.165	0.629	**1.841**
babb	0.483	0.570	0.573	0.388	0.382	0.391	0.213	0.545	0.540	**1.055**
bbaa	0.358	0.552	0.555	0.750	1.672	1.096	1.059	0.703	0.626	**1.839**
bbab	0.358	0.573	0.578	0.289	0.854	0.312	0.228	0.545	0.537	**1.083**
bbba	0.291	0.595	0.604	0.416	1.003	0.897	0.726	0.312	0.358	**1.235**
bbbb	0.291	0.804	1.000	0.290	0.355	0.266	0.239	0.269	0.195	**1.047**

Keynote

On the Variety of Uncorrelated Shapes in Digital Trees

Mark Daniel Ward[1]

[1] *Department of Statistics, Purdue University, USA*

Abstract

We discuss collections of uncorrelated motifs in digital trees. We translate pattern matching problems in sequences into analogous problems in digital trees. We use examples from genomic sequences as examples of the methodology. We show that (under appropriate normalization) for a collection of uncorrelated motifs, any linear combination of the number of occurrences of each of the motifs in the data has a limiting normal distribution. We discuss some of the connections with the methodology of combinatorics on words, integral transforms, and poissonization. We endeavor to keep the discussion at an accessible level, for colleagues who are studying similar problems from different disciplinary perspectives.

This project was conducted jointly with Jeff Gaither (The Ohio State University, USA) and Hosam Mahmoud (The George Washington University).

Actes du workshop pluridisciplinaire SeqBio 2015

Extended Abstract

Résultats algorithmiques pour le design d'ARN avec contraintes de séquence

Vincent Le Gallic[1,2,3], Alain Denise[1], Yann Ponty[2,3]

[1] LRI, Université Paris-Sud, Orsay, France
[2] LIX, Ecole Polytechnique, Palaiseau, France
[3] EP Amib, Inria Saclay, France

Abstract

Le problème du design consiste à concevoir une ou des séquences d'ARN qui vont, dans un modèle énergétique, se replier en une structure cible. Les algorithmes déjà existants dans le domaine ne gèrent pour la plupart pas l'ajout au problème des contraintes de séquence, c'est-à-dire l'interdiction ou l'obligation d'utiliser certains motifs.

Zhou *et al.* [13] proposent un algorithme qui utilise de la génération aléatoire dans un langage formel conditionné par un automate fini qui gère les contraintes de motifs interdits/imposés et une grammaire non contextuelle qui gère les contraintes imposées par la structure recherchée.

On propose ici de se baser sur cet algorithme en y ajoutant des optimisations permettant de réduire sa complexité. Ce travail soulève également des questions ouvertes sur la construction efficace d'un automate (quasi)minimal, ainsi que l'incorporation de contraintes supplémentaires garantissant le repliement des séquences engendrées vers une structure cible à l'exclusion de tout autre repliement.

Keywords

RNA secondary structure — Sequence design — Formal languages

1. Introduction

Le problème du design d'ARN a été pour la première fois abordé par [6], qui propose l'algorithme `RNAinverse`, qui trouve une séquence par recherche stochastique.

Les algorithmes `RNA-SSD` [2], `INFO-RNA` [4] et `NUPack` [12] reposent également sur la recherche stochastique, en utilisant une heuristique *diviser pour régner* avec différentes stratégie de découpage du problème. `RNA-ensign` [7] et `IncaRNAtion` [11] utilisent la génération aléatoire, tandis que `FRNAKenstein` [8] et `RNAExInv` [3] sont des algorithmes génétiques. Il existe également des algorithmes exacts, de complexité expontentielle, comme `RNAiFold` [5].

Zhou *et al.* introduisent `CFGRNAD` [13] qui utilise des langages formels (automate et grammaire non-contextuelle) pour conditionner une génération aléatoire de séquences candidates à être un bon design. C'est cette méthode qu'on propose ici d'améliorer en optimisant certaines étapes du calcul.

2. Définition du problème

On modélise une structure secondaire d'ARN sans pseudo-noeuds par un mot bien parenthésé sur l'alphabet $\{(,), \bullet, \}$. L'absence de pseudo-noeud se traduit par le fait que les arcs ne se coupent pas dans une représentation linéaire de la molécule, comme illustré par la Figure 1.

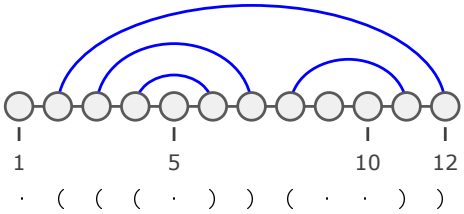

Figure 1. Représentation par arcs et par mot

Pour une séquence de base azotées s repliée selon une structure S, on peut, étant donné un modèle d'énergie des structures secondaire d'ARN (typiquement, celui de Turner [10]), calculer son énergie libre $E(s, S)$. Le problème du repliement consiste, pour une séquence s, à trouver l'ensemble de structures $\mathsf{MFE}(s)$ qui minimisent $E(s, S)$. On peut alors définir le problème du **repliement inverse**, aussi appelé design négatif, tel que suit:

Repliement inverse
Données : Structure secondaire S Résultat : Séquence s telle que $\mathsf{MFE}(s) = \{S\}$ si possible, ou $\varnothing$ sinon

La complexité de ce problème est actuellement inconnue, même si il est soupçonné NP-complet, et il n'existe actuellement aucun algorithme polynomial offrant des garanties théoriques, exactes ou approchées, pour ce problème.

Dans des contextes applicatifs, il peut parfois être utile de construire des séquences s' dont le repliement (MFE) n'est pas exactement S, mais une structure "proche". On veut de plus ajouter des contraintes supplémentaires au problème :

- **Contraintes positionnelles.** Exemple : À la position 27 de la séquence, il ne peut y avoir que A ou G;
- **Motifs interdits.** Exemple : AUGGG ne doit pas apparaître;
- **Motifs obligatoires.** Exemple : CAAU doit apparaître au moins une fois.

On peut maintenant définir le problème du **design d'ARN contraint** :

Design contraint
Données : Structure S de taille $\lvert S\rvert = n$, ensemble de mots interdits $\mathcal{F}$, ensemble de mots obligatoires $\mathcal{M}$, ensemble de contraintes positionnelles $PC = \{(i, C_i)/i \in [\![1, n]\!], C_i \subseteq \Sigma\}$. Résultat : Séquence s telle que $\forall w \in \mathcal{F}, w \not\preceq {}^a s$, $\forall w \in \mathcal{M}, w \preceq s$, $\forall i \in [\![1, n]\!], s_i \in C_i$, ou $\varnothing$ si aucune séquence ne satisfait ces contraintes [a] $u \preceq v$ signifie u facteur de v

3. Approche de Zhou *et al* [13]

Une solution exacte, en temps exponentielle, du problème a été proposée par [13], via une formulation issue de la théorie des langages.

Pour une structure S fixée, l'algorithme commence par créer une grammaire qui engendre le mot parenthésé représentant la structure S, comme l'illustre la Figure 2 :

$$
\begin{array}{lll}
S_1 \to \bullet S_2 & S_4 \to (S_5) & S_8 \to (S_9) \\
S_2 \to (S_3) & S_5 \to \bullet & S_9 \to \bullet S_{10} \\
S_3 \to (S_4) S_8 & & S_{10} \to \bullet
\end{array}
$$

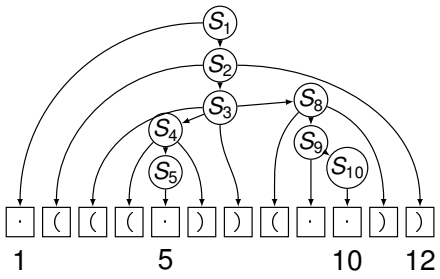

Figure 2. Dérivation du mot dans la grammaire $\mathcal{G}$

Cette grammaire est alors développée afin d'émettre une des lettres $\{\mathsf{A}, \mathsf{U}, \mathsf{G}, \mathsf{C}\}$ à chaque position non-appariée $\bullet$, et de remplacer les couples de parenthèses ( ) en correspondance par des paires $\{\{\mathsf{A}, \mathsf{U}\}, \{\mathsf{G}, \mathsf{C}\}, \{G, \mathsf{U}\}\}$. La grammaire $\mathcal{G}_S$ ainsi obtenue reconnait le langage de toutes les séquences compatibles avec S :

$$\begin{aligned} S_1 &\rightarrow aS_2 \mid uS_2 \mid gS_2 \mid cS_2 \\ S_2 &\rightarrow aS_3u \mid uS_3a \mid gS_3c \mid cS_3g \mid gS_3u \mid uS_3g \\ &\dots \\ S_5 &\rightarrow a \mid u \mid g \mid c \\ &\dots \end{aligned}$$

Pour un ensemble de motifs interdits $\mathcal{F}$, on définit l'automate $\mathcal{A}_\mathcal{F}$ qui est l'automate d'Aho-Corasick reconnaissant le language $\mathcal{F}$ (fig 3)

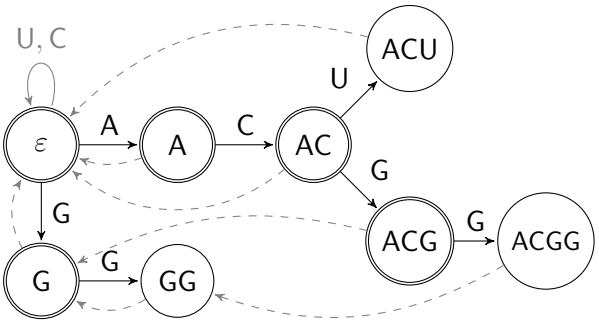

Figure 3. Automate $\mathcal{A}_\mathcal{F}$, où $\mathcal{F} = \{\mathsf{ACU}, \mathsf{ACGG}, \mathsf{GG}\}$, ici représenté avec des ε-transitions appelées *failure transitions*, pouvant être précisées en temps $\mathcal{O}(|\mathcal{A}_\mathcal{F}|)$ afin d'obtenir un automate déterministe.

Pour ajouter les contraintes de mots obligatoires, on peut créer $\mathcal{A}_\mathcal{M}$ qui reconnaît tous les mots ayant pour facteurs tous les mots de $\mathcal{M}$, dans n'importe quel ordre. Cela consiste à lire un mot de $\mathcal{M}$, puis n'importe quel *autre* mot de $\mathcal{M}$ jusqu'à en avoir lu le bon nombre (cf figure 4).

On obtient finalement $\mathcal{A}_{\mathcal{F},\mathcal{M}}$ en faisant le produit des automates $\mathcal{A}_\mathcal{F}$ et $\mathcal{A}_\mathcal{M}$, dont l'ensemble d'état est $Q_{\mathcal{F},\mathcal{M}}$ tel que

$$|Q_{\mathcal{F},\mathcal{M}}| \in O\left(2^{|M|}\left(\sum_{f\in\mathcal{F}}|f| + \sum_{m\in\mathcal{M}}|m|\right)\right).$$

Ainsi, $\mathcal{G}_\mathcal{S}$ gère les contraintes structurelles tandis que $\mathcal{A}_{\mathcal{F},\mathcal{M}}$ gère les motifs interdits et obligatoires. Les contraintes positionnelles sont prises en compte dans la grammaire $\mathcal{G}_\mathcal{S}$ via un filtrage des règles développées pour $\bullet$ et (). L'intersection d'un langage non contextuel et d'un langage régulier est un langage non contextuel, qui peut être reconnu en simulant l'exécution de l'automate au cours des dérivations de la grammaire, résultant en une grammaire ayant $|\mathcal{G}_\mathcal{S}| \cdot |Q_{\mathcal{F},\mathcal{M}}|^3$ dérivations.

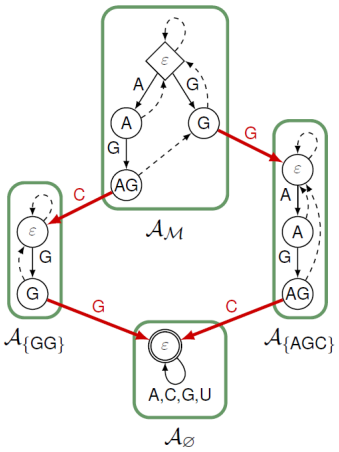

Figure 4. Exemple de $\mathcal{A}_{\mathcal{M}}$ avec $\mathcal{M} = \{\mathsf{AGC}, \mathsf{GG}\}$

4. Contributions

4.1 Programmation dynamique pour un motif unique placé

Une faiblesse de l'approche purement linguistique de Zhou *et al* [13] apparaît lorsqu'on cherche à forcer un motif unique placé ω, afin qu'il soit présent à une position spécifique i_ω, tout en étant interdit partout ailleurs. En effet, si l'on essaye d'exprimer cette contrainte dans la grammaire $\mathcal{G}_{\mathcal{S}}$, l'automate interdisant le mot ω, le produit automate-grammaire sera vide, la grammaire n'engendrant que des mots contenant le motif ω, refusé par l'automate. En revanche, si l'on essaye de capturer cette contrainte dans l'automate $\mathcal{A}_{\mathcal{F},\mathcal{M}}$, alors le nombre d'états de l'automate minimal croit en $\Omega(n)$, ce qui résulte en un surcoût de complexité en $O(n^3)$.

L'algorithme proposé ici permet de régler ce problème, à travers une fonction $C(i,j,q,q'')$ qui compte les mots sur l'intervalle $[i,j]$ remplissant le contrat *partant d'un état q de l'automate, on arrive après lecture du mot dans un état q''*. La valeur nous intéressant au final est donnée par $C(1,n,q_0,q_f)$ pour tout q_0 (respectivement q_f) initial (resp. final) dans $\mathcal{A}_{\mathcal{F},\mathcal{M}}$. Chaque contrat dépend de sous-contrats de taille strictement inférieure (au sens : la distance $j-i$ est plus faible), ce qui permet d'envisager la programmation dynamique.

L'algorithme calcule (et mémorise) chacun des $C(i,j,q,q'')$. Si la base i n'est pas appariée, remplir le contrat de $C(i,j,q,q'')$ consiste à choisir une lettre $a \in \Sigma$ et remplir $C(i+1,j,q_{\text{after}},q'')$ où $q \xrightarrow{a} q_{\text{after}}$ est une transition dans $\mathcal{A}_{\mathcal{F},\mathcal{M}}$. Si la base i est appariée à la base k, remplir $C(i,j,q,q'')$ consiste à choisir une lettre $a \in \Sigma$, un état $q' \in Q$ et remplir $C(i,k,q,q')$ et $C(k+1,j,q_{\text{after}},q'')$ où $q' \xrightarrow{a} q_{\text{after}}$ est une transition dans $\mathcal{A}_{\mathcal{F},\mathcal{M}}$.

On en déduit immédiatement des équations de programmation dynamique, dont on présente uniquement le cas apparié dans un soucis de concision :

$$C(i,j,q,q'') = \sum_{\substack{(a,b)\in C_i\times C_j \\ q'\in Q}} \begin{cases} 0 & \text{si } \delta(q,a)\in\mathcal{F} \text{ ou } \delta(q',b)\in\mathcal{F} \\ 0 & \text{si } \delta(q,a)=\omega \text{ et } i-(|\omega|-1)\neq i_\omega \\ 0 & \text{si } \delta(q',b)=\omega \text{ et } k-(|\omega|-1)\neq i_\omega \\ C(i+1,k-1,\delta(q,a),q')\cdot C(k+1,j,\delta(q',b),q'') & \text{sinon} \end{cases}$$

où les états de l'automate d'Aho Corasick (non minimal) sont confondus avec les suffixes maximaux de mots de $\mathcal{F}\cup\mathcal{M}\cup\{\omega\}$ par abus de notation. La complexité de l'algorithme reste donc en $O(n\cdot|Q|^3)$ malgré la contrainte de motif unique placé ω.

4.2 Amélioration empirique de la complexité

Le terme dominant de complexité de l'algorithme de Zhou *et al* [13] (ainsi que celui décrit en section 4.1) résulte de la simulation explicite de l'exécution de l'automate lors des règles de type *produit*. En effet, pour chaque règle de la grammaire de type *produit* $S \to (S')S''$ l'algorithme créé autant de règles qu'il existe de triplets $(q, q', q'') \in Q^3$. Cependant, pour certains triplets, il n'existe aucun chemin $q \to^{n'} q' \to^{n''} q''$ dans $\mathcal{A}_{\mathcal{F}}$, où n' et n'' sont les tailles des mots engendrés par S' et S'' respectivement.

Notre optimisation consiste donc à éviter la création de dérivations improductives, en pré-calculant les triplets (q, q', q'') tels qu'un tel chemin existe. On utilise pour cela un algorithme simple de programmation dynamique qui calcule et mémorise, pour chaque couple $(q, n') \in Q \times \mathbb{N}^2$, l'ensemble $d(q, n')$ (resp. $i(q, n')$) des états $q' \in Q$ tels qu'il existe au moins un chemin $q \to^{n'} q'$ dans $\mathcal{A}_{\mathcal{F}}$ (resp. $q' \to^{n'} q$). Le calcul des triplets pertinents est alors obtenu comme une simple intersection.

Une fois ce précalcul effectué, on constate une économie empirique substantielle en temps de calcul.

4.3 Complexité de la génération à mots imposés

Le fait que l'algorithme ait une complexité exponentielle en le nombre de motifs imposés est peu surprenant, car le problème du design contraint est NP-difficile.

En effet, rappelons le problème **Superstring** :

Superstring
Données : Ensemble $\{u_1, \ldots, u_k\}$ des mots sur un alphabet Σ et $l \in \mathbb{N}$.
Résultat : Vrai si il existe u tel que $\forall i \in [\![1, k]\!], u_i \preceq u$, et $\lvert u \rvert \leqslant l$, Faux sinon

Ce problème est NP-difficile [9]. Si on considère le problème du design contraint avec les contraintes suivantes : $S = \bullet^l$ (structure "vide" de taille l), $\mathcal{F}$, $\mathcal{M} = \{u_1, \ldots, u_k\}$, alors, résoudre ce problème de design contraint et tester si le résultat est vide est équivalent au problème du **Superstring**.

Ainsi il n'est pas surprenant que la dépendance de la complexité en le nombre de mots interdits ($k = m = |\mathcal{M}|$) soit exponentielle et il est peu probable qu'un algorithme vienne améliorer significativement la dépendance exponentielle en le nombre de motifs obligatoires.

5. Problèmes soulevés

Ce travail soulève les questions suivantes :

- L'automate $\mathcal{A}_{\mathcal{F}}$ est-il *génériquement* optimal [1] pour gérer la contrainte imposée par $\mathcal{F}$? Une minimisation *a posteriori* est en effet coûteuse, et entraîne des difficultés au niveau de la programmation dynamique.
- Notre algorithme ne garantit pas le repliement des séquences engendrées vers une structure cible S. Comment *biaiser* la génération aléatoire pour favoriser les séquences dont le repliement est "proche" de S ?
- Nous souhaitons enfin caractériser des classes d'automates/contraintes telles que l'économie obtenue par notre précalcul est asymptotiquement significative.

References

[1] Omar AitMous, Frédérique Bassino, and Cyril Nicaud. An efficient linear pseudo-minimization algorithm for aho-corasick automata. In *Combinatorial Pattern Matching - 23rd Annual Symposium, CPM 2012, Helsinki, Finland, July 3-5, 2012. Proceedings*, pages 110–123, 2012.

[2] Mirela Andronescu, Anthony P. Fejes, Frank Hutter, Holger H. Hoos, and Anne Condon. A new algorithm for RNA secondary structure design. *J Mol Biol*, 336(3):607–624, Feb 2004.

[3] Assaf Avihoo, Alexander Churkin, and Danny Barash. RNAexinv: An Extended Inverse RNA Folding from Shape and Physical Attributes to Sequences. *BMC Bioinformatics*, 12(1):319, Aug 2011.

[4] Anke Busch and Rolf Backofen. INFO-RNA–a fast approach to inverse RNA folding. *Bioinformatics*, 22(15):1823–31, Aug 2006.

[5] Juan Antonio Garcia-Martin, Peter Clote, and Ivan Dotu. RNAiFold: a web server for RNA inverse folding and molecular design. *Nucleic Acids Res*, 41(Web Server issue):W465–W470, Jul 2013.

[6] I. L. Hofacker, W. Fontana, P. Stadler, L. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie / Chemical Monthly*, 125(2):167–188, 1994.

[7] Alex Levin, Mieszko Lis, Yann Ponty, Charles W O'Donnell, Srinivas Devadas, Bonnie Berger, and Jérôme Waldispühl. A global sampling approach to designing and reengineering RNA secondary structures. *Nucleic Acids Res*, 40(20):10041–52, Nov 2012.

[8] Rune B Lyngsø, James Wj Anderson, Elena Sizikova, Amarendra Badugu, Tomas Hyland, and Jotun Hein. Frnakenstein: multiple target inverse RNA folding. *BMC Bioinformatics*, 13:260, 2012.

[9] D Maier and J.A. Storer. A note on the complexity of superstring problem. Technical Report Report 233, Computer Science Lab, Princeton University, 1977.

[10] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of rna secondary structure. *J Mol Biol*, 288(5):911–940, May 1999.

[11] Vladimir Reinharz, Yann Ponty, and Jérôme Waldispühl. A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotide distribution. *Bioinformatics*, 29(13):i308–i315, Jul 2013.

[12] Joseph N Zadeh, Brian R Wolfe, and Niles A Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *J Comput Chem*, 32(3):439–52, Feb 2011.

[13] Yu Zhou, Yann Ponty, Stéphane Vialette, Jérôme Waldispühl, Yi Zhang, and Alain Denise. Flexible RNA design under structure and sequence constraints using formal languages. In *ACM Conference on Bioinformatics, Computational Biology and Biomedical Informatics*, Bethesda, Washigton DC, United States, September 2013.

Actes du workshop pluridisciplinaire SeqBio 2015

Extended Abstract

Counting, generating and sampling tree alignments

Cedric Chauve[1]*, Julien Courtiel[1,2] and Yann Ponty[2,3]

[1] *Department of Mathematics, Simon Fraser University, Canada*
[2] *Pacific Institute for the Mathematical Sciences, Vancouver, Canada*
[2] *CNRS-LIX, Ecole Polytechnique, France*
***Corresponding author**: cedric.chauve@sfu.ca

Abstract
Pairwise ordered tree alignment are combinatorial objects that appear in important applications, such as RNA secondary structure comparison. However, the usual representation of tree alignments as supertrees is ambiguous, i.e. two distinct supertrees may induce identical sets of matches between identical pairs of trees. This ambiguity is uninformative, and detrimental to any probabilistic analysis. In this work, we consider tree alignments up to equivalence. Our first result is a precise asymptotic enumeration of tree alignments, obtained from a context-free grammar by mean of basic analytic combinatorics. Our second result focuses on alignments between two given ordered trees S and T. By refining our grammar to align specific trees, we obtain a decomposition scheme for the space of alignments, and use it to design an efficient dynamic programming algorithm for sampling alignments under the Gibbs-Boltzmann probability distribution. This generalizes existing tree alignment algorithms, and opens the door for a probabilistic analysis of the space of suboptimal alignments.

Keywords
Tree alignment — RNA secondary structure — Dynamic programming

1. Introduction

Tree alignments are the natural analog of sequence alignments, and have been introduced by Jiang, Wang and Zhang [1] to model and quantify the similarity between two (ordered[1]) trees. Tree alignment has been used in a wide array of applicative contexts ranging from web crawling [2] to software engineering [3] through RNA Bioinformatics [4]. The minimal cost tree alignment between two trees of size n_1 and n_2, under classic insertion/deletion/(mis)-match operations, can be computed using dynamic programming (DP). The current best algorithms have a worst-case time and space complexity respectively in $\mathcal{O}(n_1 n_2 (n_1+n_2)^2)$ and $\mathcal{O}(n_1 n_2 (n_1+n_2))$ [1] algorithms, and an average-case time and space complexity (on uniformly drawn instances) in $\mathcal{O}(n_1 n_2)$ [5].

In the context of sequence alignments, the enumeration of alignments has been the object of much interest in Computational Biology [6, 7, 8]. Alignments between two sequences over an alphabet Σ can be encoded as sequences over an extended alphabet Σ_a, representing insertions, deletions and (mis)matches (*e.g.* $\Sigma = \{a, b\}$, $\Sigma_a = \{(a,-),(-,b),(a,b),(a,a),(b,a),(b,b)\}$). Many sequences over Σ_a are equivalent if one considers only (mis)matches of the alignments, *i.e.* they align sequence of same lengths and induce the same sets of matched positions (*e.g.* $(a,-),(-,b)$ and $(-,b),(a,-)$). It is

[1] In this work, unless explicitly specified, all trees will be rooted and ordered.

a natural problem to enumerate distinct sequence alignments for two sequences of cumulated length n [9, pp. 188]. Beyond purely theoretical considerations, the decompositions introduced for enumerating distinct sequence alignments were adapted into DP algorithms, *e.g.* for probabilistic alignment based on expectation maximization [10], or to compute Gibbs-Boltzmann measures of reliability [11].

In the present work, we consider similar questions on *tree alignments*. We are first interested in counting distinct tree alignments, *i.e.* enumerating, up to equivalence, ordered trees whose vertices are labeled in Σ_a (called *supertrees* from now). For trees, the notion of equivalence of alignments generalizes that of sequence alignments, *i.e.* two alignments are *equivalent* when they align the same pairs of trees, and induce the same sets of (mis)matched positions. Unfortunately, contrasting with the case of sequence alignments, existing DP algorithms for computing an optimal tree alignment [1, 12, 13] cannot be easily adapted into enumeration schemes for tree alignments up to equivalence. This additional difficulty is due to the existence of ambiguities of different nature.

Our main contribution is a grammar for (distinct) tree alignments, which provably generates a single representative for each equivalence class. We use the symbolic method [14] to obtain the generating function of tree alignments, and asymptotic equivalents for various statistics of interest can easily be derived, such as the average number of alignments over trees of total size n. Finally, and, perhaps more importantly from an applied point of view, the grammar can be transformed into an unambiguous and complete DP algorithm for aligning two input trees. The resulting algorithm has the same asymptotic worst-case and average-case complexities, up to reasonable constants, as the current best – ambiguous – algorithm [1, 12]. The main interest of such an algorithm is that it opens immediately the way to new applications for the tree alignment model, including a critical assessment of the reliability of optimal alignments, either obtained by counting co-optimal alignments, or by sampling suboptimal alignments according to a Gibbs-Boltzmann distribution (see [15] for an example of this approach for the RNA folding problem).

2. Background

Trees and supertrees. Let Σ be an *alphabet*. A tree T on Σ is a rooted plane tree whose vertices are labeled by elements of Σ. We denote by V_T the set of vertices of T. We *remove a non-root vertex* v from a tree T by contracting the edge between v and its parent u, that keeps its label. Removing the root r of a tree consists in creating a forest composed of the subtrees rooted at the children of r. We denote the operation of removing a vertex v from T by $T - v$. We denote by Σ_a the alphabet defined by $\Sigma_a = (\Sigma \cup \{-\})^2 - \{(-,-)\}$. An element $(x, y) \in \Sigma_a$ is an *insertion* (resp. *deletion*, *match*) if $y = -$ (resp. $x = -$, $(x, y) \in \Sigma^2$). A *supertree* A is a tree on Σ_a; a vertex of A is an insertion (resp. deletion, match) if its label is an insertion (resp. deletion, match). The size of a supertree A is the number of its insertions and deletions, plus twice the number of its matches. A *superforest* is an ordered sequence of supertrees.

Given a supertree A on Σ, we define two forests $\pi_1(A)$ and $\pi_2(A)$ as follows: $\pi_1(A)$ (resp. $\pi_2(A)$) is obtained by (1) iteratively removing all insertion (resp. deletions) of A, in an arbitrary order, and (2) replacing the label (x, y) of each remaining vertex by x (resp. y). We refer to Fig. 1 for an illustration. We extend the notations π_1 and π_2 on vertices: for a non-insertion (resp. non-deletion) vertex v of A, we denote by $\pi_1(v)$ (resp. $\pi_2(v)$) the corresponding vertex in $\pi_1(A)$ (resp. $\pi_2(A)$). A vertex x of $\pi_1(A)$ such that $\pi_1^{-1}(x)$ is an

insertion (resp. match) is said to be inserted (resp. matched) in A. Similarly, a vertex y of $\pi_2(A)$ such that $\pi_2^{-1}(y)$ is a deletion (resp. match) is said to be deleted (resp. matched) in A.

Tree alignments. As forests $\pi_1(A)$ and $\pi_2(A)$ are embedded into the supertree A, the latter implicitly defines an *alignment* between the forests $\pi_1(A)$ and $\pi_2(A)$, *i.e.* a set of correspondences between vertices of $\pi_1(A)$ and $\pi_2(A)$, that is consistent with the structure of both forests [1]. We refer to Fig. 1 for an illustration.

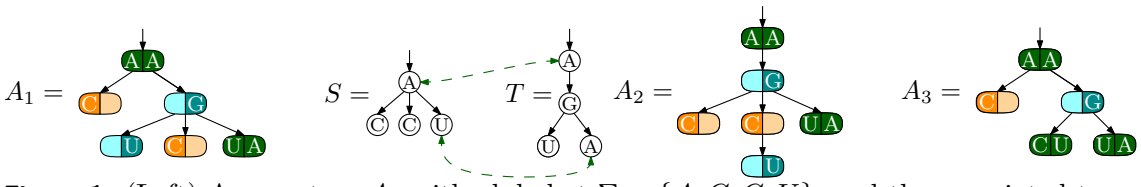

Figure 1. (Left) A supertree A_1 with alphabet $\Sigma = \{A, C, G, U\}$, and the associated trees $S = \pi_1(A_1)$ and $T = \pi_2(A_1)$. represent the insertion vertices; the blue vertices, labeled by a blank then a star, represent the deletion vertices; the green vertices, labeled by a double star, represent the match vertices. The alignment of S and T defined by A is composed of two pairs of matched (A, A) and (U, A), indicated by dashed arrows. (Right) Two non-equivalent supertrees, representing two different tree alignments. However, while the supertrees A_1 and A_2 from the left part of the figure are equivalent.

We now turn to the central notion of *equivalent alignments*, *i.e.* alignments of identical pairs of trees, that contain exactly the same set of matched vertices. Given a supertree A, representing an alignment between two trees $S = \pi_1(A)$ and $T = \pi_2(A)$, the *set of matches of* A is formed by the elements (x, y) of $V_S \times V_T$ such that $\pi_1^{-1}(x) = \pi_2^{-1}(y)$ (*i.e.* there exists a vertex v of A such that $\pi_1(v) = x$ and $\pi_2(v) = y$). Two supertrees A_1 and A_2 are *equivalent* if $\pi_1(A_1) = \pi_1(A_2)$, $\pi_2(A_1) = \pi_2(A_2)$, and the sets of matches of A_1 and A_2 are identical (see Fig. 1 for an illustration).

A *tree alignment* is then defined as an equivalence class over supertrees with respect to the above-defined equivalence relation, for which $\pi_1(A)$ and $\pi_2(A)$ are trees. The notion of *forest alignment* is similarly defined when $\pi_1(A)$ and $\pi_2(A)$ are not restricted to trees. Given a set $\mathcal{S}$ of tree (resp. forest) alignments, a set $\mathcal{T}$ of supertrees (resp. superforests) is said to be *representative of* $\mathcal{S}$ if it contains exactly one supertree (resp. superforest) for each alignment (*i.e.* equivalence classes of supertrees and forests) in $\mathcal{S}$. Tree alignments will now be the focus of our work.

3. Results

A grammar for tree alignments. In this section, we describe a context-free grammar for a set $\mathcal{A}$ of supertrees that is representative of the set of all tree alignments.

Theorem 1 *The set of supertrees $\mathcal{A}$ generated by the grammar (1)-(8) is representative of the set of all tree alignments; i.e. $\mathcal{A}$ contains exactly one supertree for each equivalence class of supertrees.*

Application: Enumerating tree alignments. For the sake of simplicity, we will restrict our attention to $|\Sigma| = 1$, *i.e.* the alphabet is restricted to a single letter. The general case follows easily, and will be described in an extended version of the paper. Using the *symbolic*

$$\begin{aligned}
\mathcal{A} &= \mathcal{V}^{\varnothing} \oplus \mathcal{T}_I \oplus \mathcal{T}_D \oplus \mathrm{InsRoot}\,(\mathcal{F}_I \circ \mathcal{T}_D) && (1)\\
\mathcal{T}_I &= \mathrm{InsRoot}\,(\mathcal{F}_I)\,, \quad \mathcal{F}_I = \{\text{empty supertree}\} \oplus \mathrm{InsRoot}\,(\mathcal{F}_I) \circ \mathcal{F}_I && (2)\\
\mathcal{T}_D &= \mathrm{InsRoot}\,(\mathcal{F}_D)\,, \quad \mathcal{F}_D = \{\text{empty supertree}\} \oplus \mathrm{InsRoot}\,(\mathcal{F}_D) \circ \mathcal{F}_D && (3)\\
\mathcal{V}^{\varnothing} &= \mathcal{V}^{\uparrow} \oplus \mathrm{InsRoot}\,(\mathcal{VH}) && (4)\\
\mathcal{V}^{\uparrow} &= \mathrm{MatchRoot}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}\right) \oplus \mathrm{DelRoot}\left(\mathcal{F}_D \circ \mathcal{V}^{\uparrow} \circ \mathcal{F}_D\right) && (5)\\
\mathcal{VH} &= \mathcal{F}_I \circ \mathcal{VH} \oplus \mathcal{V}^{\varnothing} \circ \mathcal{F}_I \oplus \mathrm{DelRoot}\left(\mathcal{H}_{\mathsf{I|D},\leftrightarrow,\varnothing}\right) \circ \mathcal{F}_I && (6)
\end{aligned}$$

For every ν, M, M' with $\nu \in \{\mathsf{I|D}, \mathsf{D}\}$ and $M, M' \in \{\varnothing, \leftrightarrow, \rightarrow\}$:

$$\mathcal{H}_{\nu,M,M'} = \bigoplus \begin{cases} \{\text{empty supertree}\} & \text{if } (M,M') = (\varnothing,\varnothing) \\ \mathcal{T}_I \circ \mathcal{H}_{\nu,M,M'} & \text{if } \nu \neq \mathsf{D} \text{ and if } M \neq \leftrightarrow \\ \mathcal{T}_D \circ \mathcal{H}_{\mathsf{D},M,M'} & \text{if } M' \neq \leftrightarrow \\ \mathcal{V}^{\varnothing} \circ \overline{\mathcal{H}}^{\mathbf{1},\mathbf{1}}_{M,M'} & \\ \mathrm{InsRoot}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\leftrightarrow}\right) \circ \overline{\mathcal{H}}^{\mathbf{1},+}_{M,M'} & \\ \mathrm{DelRoot}\left(\mathcal{H}_{\mathsf{D},\leftrightarrow,\varnothing}\right) \circ \overline{\mathcal{H}}^{+,\mathbf{1}}_{M,M'} & \end{cases} \quad (7)$$

For every $M, M' \in \{\varnothing, \leftrightarrow, \rightarrow\}$ and $i, j \in \{\mathbf{1}, +\}$:

$$\overline{\mathcal{H}}^{i,j}_{M,M'} = \mathcal{H}_{\mathsf{I|D},\alpha(M),\alpha(M')} \oplus \begin{cases} \mathcal{F}_I & \text{if } M = \varnothing \text{ and } M' = \rightarrow \\ \mathcal{F}_I & \text{if } M = \varnothing, M' = \leftrightarrow \text{ and } j = + \\ \mathcal{F}_D & \text{if } M = \rightarrow \text{ and } M' = \varnothing \\ \mathcal{F}_D & \text{if } M = \leftrightarrow,\ M' = \varnothing \text{ and } i = + \\ \varnothing & \text{otherwise} \end{cases} \quad (8)$$

where $\alpha(\varnothing) = \varnothing$ and $\alpha(\leftrightarrow) = \alpha(\rightarrow) = \rightarrow$.

Figure 2. A context-free grammar for $\mathcal{A}$, a representative set of all tree alignments.

method [14], one classically translates the specification described by Eqs. (1)-(8) into a system of functional equations relating the generating functions of the sets of supertrees and forests. This leads to the following set of enumerative results, that characterize precisely the set of all tree alignments.

Theorem 2 *The generating functions $T(t,z)$ and $F(t,z)$ of tree and forest alignments, whose size and number of matches are marked by t and z respectively, satisfy*

$$T(t,z) = \left(t^2 + t - t^2 z + \frac{t}{\sqrt{1-4t}}\right) F(t,z), \quad (9)$$

$$(tzC(t)^2 - t^2C(t)^2 + 2t)F(t,z)^2 + (t^2C(t)^4 - 2tC(t)^2 - 1)F(t,z) + C(t)^2 = 0, \quad (10)$$

where $C(t) = (1-\sqrt{1-4t})/2t$ is the generating function of Catalan numbers.

Theorem 3 *The number of tree alignments of size n is asymptotically equivalent to $\kappa \times n^{-3/2} \times 6^n$, where $\kappa = \sqrt{2}(3-\sqrt{3})/(24\sqrt{\pi})$.*

Corollary 4 *The average number of tree alignments for a random pair of trees of cumulated size n is $\kappa' \times 1.5^n$, where $\kappa' = \sqrt{2}(3-\sqrt{3})/6$.*

Application: Sampling alignments between two given trees. We now consider two *fixed* trees S and T, and consider the task of sampling a tree alignment A such that $\pi_1(A) = S$ and $\pi_2(A) = T$, with respect to the Gibbs-Boltzmann probability distribution. , This can be used to assess the stability of a prediction. We refer the interested reader to our introduction for examples of further motivation and possible applications.

Let $\mathcal{T}_{S,T}$ be the set of all supertrees A such that $\pi_1(A) = S$ and $\pi_2(A) = T$, and $\mathcal{A}_{S,T}$ be a representative set of $\mathcal{T}_{S,T}$. In other words, $\mathcal{A}_{S,T}$ can be interpreted as the set of all alignments between S and T. For any supertree $A \in \mathcal{T}_{S,T}$, we define its *edit score* $s(A)$ as the sum of the number of insertions, deletions and matches (x, y) such that $x \neq y$.[2] For a given positive constant kT, the *partition function* $Z_{S,T}$ of $\mathcal{A}_{S,T}$ and the *Gibbs-Boltzmann probability* $\Pr(A)$ of an alignment $A \in \mathcal{A}_{S,T}$ are defined as

$$Z_{S,T} = \sum_{A \in \mathcal{A}_{S,T}} e^{-s(A)/kT}, \quad \Pr(A) = \frac{e^{-s(A)/kT}}{Z_{S,T}}.$$

When kT tends to 0, this distribution tends to the uniform distribution over supertrees of minimum edit score, while, when kT tends to $+\infty$, it tends toward the uniform distribution over $\mathcal{A}_{S,T}$.

We consider the following problem: given two trees S and T, and a positive constant K, design a sampling algorithm for alignments between S and T under the Gibbs-Boltzmann probability distribution. This problem generalizes the classic combinatorial optimization problem of computing a tree alignment between S and T having minimum edit score.

To address this problem, we rely on dynamic programming, by the approach described, among others, in [15] for RNA folding. We proceed in two steps: adapting the grammar introduced in Section 3 into a grammar for $\mathcal{A}_{S,T}$, then using this grammar as the skeleton of an efficient Dynamic Programming sampling algorithm. This leads to the following result.

Theorem 5 *Let S and T be two trees of respective sizes n_S and n_T. Sampling a supertree from $\mathcal{A}_{S,T}$ under the Gibbs-Boltzmann distribution can be done with worst-case time and space complexities in $\mathcal{O}(n_S\, n_T\, (n_S + n_T)^2)$, while the average-case time and space complexities are in $\mathcal{O}(n_S\, n_T)$.*

Following the program outlined in [15], we are currently using our grammar and derived dynamic programming schemes to revisit the alignment of 3D models of RNA structures. More generally, it would be interesting to characterize the conditions under which an instance-agnostic grammar, enumerating a search space, could be adapted into a decomposition for a specific instance. Such a theory, at the confluence of enumerative combinatorics and algorithmic design, could provide another principled ways to design dynamic-programming algorithms.

[2] The present results can be trivially extended to any edit scoring system that is a positive linear combination of the numbers of insertions, deletions and matches.

References

[1] Tao Jiang, Lusheng Wang, and Kaizhong Zhang. Alignment of trees - an alternative to tree edit. *Theor. Comput. Sci.*, 143(1):137–148, 1995.

[2] Yanhong Zhai and Bing Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 76–85, New York, NY, USA, 2005. ACM.

[3] Wuu Yang. Identifying syntactic differences between two programs. *Softw. Pract. Exper.*, 21(7):739–755, June 1991.

[4] Matthias Höchsmann, Thomas Töller, Robert Giegerich, and Stefan Kurtz. Local similarity in rna secondary structures. *Proc IEEE Comput Soc Bioinform Conf*, 2:159–168, 2003.

[5] Claire Herrbach, Alain Denise, and Serge Dulucq. Average complexity of the Jiang-Wang-Zhang pairwise tree alignment algorithm and of a RNA secondary structure alignment algorithm. *Theor. Comput. Sci.*, 411(26-28):2423–2432, 2010.

[6] A. Dress, B. Morgenstern, and J. Stoye. The number of standard and of effective multiple alignments. *Applied Mathematics Letters*, 11(4):43 – 49, 1998.

[7] Angela Torres, Alberto Cabada, and Juan J. Nieto. An exact formula for the number of alignments between two dna sequences. *DNA Seq*, 14(6):427–430, Dec 2003.

[8] Helena Andrade, Ivan Area, Juan J. Nieto, and Angela Torres. The number of reduced alignments between two dna sequences. *BMC Bioinformatics*, 15:94, 2014.

[9] Michael S. Waterman. *Introduction to Computational Biology: Maps, Sequences, and Genomes*. CRC Press, 1995.

[10] ChuongB. Do, SamuelS. Gross, and Serafim Batzoglou. Contralign: Discriminative training for protein sequence alignment. In Alberto Apostolico, Concettina Guerra, Sorin Istrail, PavelA. Pevzner, and Michael Waterman, editors, *Research in Computational Molecular Biology*, volume 3909 of *Lecture Notes in Computer Science*, pages 160–174. Springer Berlin Heidelberg, 2006.

[11] Martin Vingron and Patrick Argos. Determination of reliable regions in protein sequence alignments. *Protein Engineering*, 3(7):565–569, 1990.

[12] Guillaume Blin, Alain Denise, Serge Dulucq, Claire Herrbach, and Hélène Touzet. Alignments of RNA structures. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 7(2):309–322, 2010.

[13] Stefanie Schirmer and Robert Giegerich. Forest alignment with affine gaps and anchors, applied in RNA structure comparison. *Theor. Comput. Sci.*, 483:51–67, 2013.

[14] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.

[15] Yann Ponty and Cédric Saule. A combinatorial framework for designing (pseudo-knotted) RNA algorithms. In Teresa M. Przytycka and Marie-France Sagot, editors, *Algorithms in Bioinformatics - 11th International Workshop, WABI 2011, Saarbrücken, Germany, September 5-7, 2011. Proceedings*, volume 6833 of *Lecture Notes in Computer Science*, pages 250–269. Springer, 2011.

Index

SeqBio 2015

gdr iM

GdR BiM