Algorithmic Contributions to Computational Molecular Biology

Stéphane Vialette

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

INSTITUT GASPARD MONGE

HABILITATION DIRIGER DES RECHERCHES

prsente par

Stphane VIALETTE

Algorithmic Contributions to Computational Molecular Biology

Soutenue publiquement le 1 Juin 2010

devant le jury compos de

Marie-Pierre Béal	Professeur, Universit Paris-Est, Examinateur
Christian Choffrut	Professeur, Universit Paris 7, Examinateur
Maxime Crochemore	Professeur, Universit Paris-Est, Examinateur
Alain Denise	Professeur, Universit Paris-Sud 11, Examinateur
Gregory Kucherov	Directeur de Recherche CNRS, Rapporteur
András Sebő	Directeur de Recherche CNRS, Rapporteur

Contents

Remerciements

Gad Landau, Gregory Kucherov et András Sebő m'ont honor en acceptant d'łtre les rapporteurs de ce mmoire. Je les en remercie vivement.

Je remercie sincrement Marie-Pierre Bal, Alain Denise, Maxime Crochemore et Christian Choffrut de m'avoir fait l'honneur de participer au jury.

Merci Marie-Piere Bal de m'avoir chaleureusement accueilli mon arrive Marne-la-Valle et y avoir favoris le dveloppement de l'algorithmique pour la biologie. Toujours disponible pour m'couter et m'encourager malgr le peu de temps que lui laisse ses responsabilits.

Je me rappelle trs bien ma premire rencontre avec Alain Denise. Dbut 2002, il m'avait demand une copie de mes transparents la suite d'un expos l'ENS o je venais de commencer mon postdoc. Qu'il puisse trouver quelque intrłt mes travaux m'avait alors la fois tonn et enthousiasm. Hasard (ou pas ?), je le rejoignais au LRI l'anne suivante. J'ai normment appris ses cots, bien plus qu'il pourrait le penser.

Maxime Crochemore tait dj membre du jury de ma thse, il en tait en fait rapporteur. Sa bonne humeur et ses qualits humaines font de nos (trop rares !) rencontres des moments des plus agrables.

Christian Choffrut. Que dire ... ? Je sais tout ce que je lui dois.

Mon parcours est jalonn de rencontres et d'expriences qui seront dterminantes. Merci donc ...

tous les membres du LIGM pour leur accueil. Un merci tout particulier Line Fonfrde et Gabrielle Brossard pour leur aide et leur disponibilit.

l'quipe Bioinformatique du LRI, Christine Froidevaux et Alain Denise (encore lui !) en tłte, pour m'avoir donn ma chance.

Claude Jacq pour m'avoir donn l'opportunit de travailler dans un laboratoire de gntique. Ces deux annes de postdoc ont t – et reste – une exprience extraordinaire (pour łtre tout fait honnłte, je pouvais mettre une blouse si l'envie m'en prenait mais il m'tait interdit de toucher quoi que ce soit sur les paillasses !).

Guillaume Fertin pour les heures de discussion, le travail en commun, les coups de fil hebdomadaires (pas forcment pour parler travail), les voyages (malheureusement bien moins frquents depuis 2006), ... mon collaborateur prfr en somme !

Guillaume Blin pour toutes ces petites choses qui font de notre bureau un endroit particulirement agrable … en n'oubliant pas les musiques improbables qu'il m'inflige (j'ai bien failli ne pas survivre la compil' spciale annes 80).

Romeo – *incredible reduction* – Rizzi et Danny Hermelin, collgues et amis, pour tout le travail en commun et le temps pass ensemble. C'est toujours un rel plaisir de les retrouver.

mes (autres) co-auteurs et/ou amis : Gaëlle Brevier, Riccardo Dondi, Mike Fellows, Isabelle Fagnot, Philippe Gambette, Sylvain Guillemot, Anthony Labarre, Matthieu Raffinot, Dror Rawitz, Irena Rusu, Éric Tannier, …

Annelyse Thvenin et Florian Sikora qui ont support mes techniques exprimentales de direction de thse et de pdagogie.

Merci Patricia, Jeanne et Nathan sans qui, tout simplement, rien ne serait possible.

Foreword

The time has come to write it! For most of us, writing our habilitation thesis is not an easy matter, and I did not depart from the rule. How to expose our works? How to establish a logical link between our works? Some of many questions that should be answered before anything else.

So first the haunting question: What should be the best format? Fortunately or not (I would go for not here), there are as many answers as people! Answers go from "extended abstract"to "PhD-like manuscript", and, as for what should be in, "cumulative"(based on previous research) or "monographical"(specific unpublished results) . . . well, answers do not help much here. The most appropriate format is often the simplest but I guess it varies from person to person and depends on what you are expecting or looking for in that exercise. So what am I expecting from this writing exercise? To cut a long story short, something useful (at least for me). Therefore, to avoid a tedious and useless exercise, my choice has been to write a survey of my works together with a brief exposition of the related results; something to which I can refer to for further research. This should explain the rather long form of the present manuscript.

As for this famous "logical link", quite naturally, I decided to focus on algorithms, and more specifically on computational molecular biology, *i.e.*, those algorithmic and combinatorial topics that are connected to molecular biology. With this in mind, I have wilfully chosen to move some of my published papers apart from my habilitation. It is not that there are too many of them but some definitively do not fit in the scope of this manuscript. First, some are clearly totally (I did not want to write "too"here) biologically oriented and have very little algorithmic content (and actually I am not really involved anymore in this activity). This includes [Lelandais et al., 2004a,b, 2006; Margeot et al., 2002; Sylvestre et al., 2003]. I shall not discuss this part here. Neither shall I discuss about my interest in graph labelings and our recent works on alliances and secure sets in graphs. Whereas my interest in these topics is clearly algorithmic, they are not related in any way to any of the four parts of the present manuscript. This includes [Fertin and Vialette, 2009; Vialette, 2006] as well as [Blin et al., 2009a]. One may argue that some parts of this habilitation thesis are quite far from any combinatorial topic connected to molecular biology (still this famous "logical link"). I have, for example, in mind Section 2.3, Section 2.4, Section 2.6, Section 10.3, . . . , and I do agree these topics are very far – not to say independent – from any computational molecular biology consideration. However, I came across these topics as special cases or relaxations of combinatorial objects that are, from my point of view, clearly connected to exploring molecular biology (I do no claim practical applications for all of them): d-intervals, linear graphs, exon shuffling, . . . My opinion is thus that these topics have their rightful places in my manuscript and, even more important from my point of view, gathering together these topics with more practical issues such as "querying PPI network"or "designing fast heuristics for comparative genomics"clearly reflects my way of doing research.

Introduction

Computational biology is (should be?) an interdisciplinary field that applies the techniques of computer science, applied mathematics and statistics to address biological problems. It encompasses many fields, ranging from *Computational biomodeling* to *Computational biochemistry*. If I would really have to place myself in this field (always such a difficult question for me, my preferred answer would be just algorithmic as I never have laid down myself to restrict to this area) I would say: *Bioinformatics* (in the very very precise sense of designing algorithms to the interpretation, classification and understanding of biological datasets) and *Comparative genomics* (as a part of *Computational genomics*).

For the prerequisites, the reader is expected to be familiar with basic graph theory, classical complexity theory and parameterized complexity theory. We only recall some basic definitions (the two following paragraphs should constitute sufficient preparation).

In computer science and operations research, *approximation algorithms* are algorithms used to find *approximate solutions* to optimization problems (the best general references are [Vazirani, 2002] and [Ausiello et al., 1999]). Approximation algorithms are often associated with **NP**-hard problems since it is unlikely that there can ever be efficient polynomial-time exact algorithms solving **NP**-hard problems, one settles for polynomial-time sub-optimal solutions. Unlike heuristics, which usually only find reasonably good solutions reasonably fast, one wants provable solution quality and provable run time bounds. Ideally, the approximation is optimal up to a small constant factor. Given an instance x of an optimization problem P, the *performance guarantee* (or *approximation ratio*) $R(x, y)$ of a solution y to the instance x is defined as

$$R(x, y) = \max\left(\frac{\mathbf{opt}(x)}{f(y)}, \frac{f(y)}{\mathbf{opt}(x)}\right),$$

where $\mathbf{opt}(x)$ is the value of an optimum solution for the instance x and $f(y)$ is the value of the solution y for the instance x. Clearly, the performance guarantee is greater than or equal to 1 (and equal to 1 if and only if y is an optimal solution). If an algorithm A guarantees to return solutions with a performance guarantee of at most $r(n)$, then A is said to be an $r(n)$-approximation algorithm and has an approximation ratio of $r(n)$. Likewise, a problem with an $r(n)$-approximation algorithm is said to be $r(n)$-approximable or have an approximation ratio of $r(n)$. The class **APX** (an abbreviation of "*approximable*") is the set of (**NPO**)

optimization problems that allow polynomial-time approximation algorithms with approximation ratio bounded by a constant (or constant-factor approximation algorithms for short). In simple terms, problems in this class have efficient algorithms that can find an answer within some fixed percentage of the optimal answer. A PTAS is an algorithm which takes an instance of an optimization problem and a parameter $\varepsilon > 0$ and, in polynomial-time, produces a solution that is within a factor ε of being optimal. Notice that the running time of a PTAS is required to be polynomial in n for every fixed ε but can be different for different ε. Thus, an algorithm, running in $O(n^{1/\varepsilon})$ time or even $O(n^{\exp(1/\varepsilon)})$ time counts as a PTAS. A practical problem with PTAS algorithms is that the exponent of the polynomial could increase dramatically as ε shrinks, for example if the runtime is $O(n^{1/\varepsilon})$. One way of addressing this is to define the *efficient polynomial-time approximation scheme* or EPTAS, in which the running time is required to be $O(n^c)$ for a constant c independent of ε. This ensures that an increase in problem size has the same relative effect on runtime regardless of what ε is being used; however, the constant under the big-O can still depend on ε arbitrarily.

For many applications the trade-offs inherent to approximation algorithms and heuristics are not satisfactory. Fixed-parameter algorithms can provide an alternative by providing optimal solutions with useful runtime guarantees (the best general references are [Downey and Fellows, 1999; Flum and Grohe, 2006; Niedermeier, 2006]). The core concept is formalized as follows: An instance of a *parameterized problem* consists of a problem instance x and a parameter k. A parameterized problem is *fixed-parameter tractable* if it can be solved in $f(k)|x|^{O(1)})$ time, where f is a computable function solely depending on the parameter k, not on the input size $|x|$. For **NP**-hard problems, $f(k)$ will of course not be polynomialsince otherwise we would have an overall polynomial-time algorithm – but typically be exponential like 2^k . Clearly, fixed-parameter tractability captures the notion of "*efficient for small parameter values*": for any constant k, we obtain a polynomial-time algorithm. Moreover, the exponent of the polynomial must be independent of k, which means that the combinatorial explosion is completely confined to the parameter. The *standard parameterization* of an optimization problem such as VERTEX COVER or CLIQUE takes the size of the solution as the parameter. Accompanying the work on designing efficient and practical parameterized algorithms, a theory of *parameterized intractability* has been developed (Downey and Fellows 1999 monograph [Downey and Fellows, 1999] gives a fairly complete picture of the theory then). In parameterized complexity, to classify fixed-parameter intractable problems, a hierarchy, the so-called **W[-]**hierarchy $\bigcup_{t\geq 0} \mathbf{W}[t]$, where $\mathbf{W}[t] \subseteq \mathbf{W}[t+1]$ for all $t \geq 0$ has been introduced, in which the 0-th level $\mathbf{W}[0]$ is the class **FPT**. The hardness and completeness have been defined for each level $\mathbf{W}[t]$ of the **W[-]**hierarchy for $t \geq 1$, and a large number of W [i]-hard parameterized problems have been identified [Downey and Fellows, 1999]. For example, the VERTEX COVER problem is known to be fixed-parameterized tractable for the standard parameterization whereas the CLIQUE problem has been proved to $\mathbf{W}[1]$-complete. The fundamental conjecture $\mathbf{FPT} \neq \mathbf{W}[1]$ is very much analogous (but clearly weaker) to the conjecture that $\mathbf{P} \neq \mathbf{NP}$. Notice that, from an algorithmic point of view, it is usually sufficient to distinguish between $\mathbf{W}[1]$-hardness and membership in **FPT**.

This habilitation thesis is organized in four parts. I would say (i) algorithmic of (not so) linear structures, (ii) pattern matching in graphs, (iii) comparative genomics, and (iv) what is left and does not fit well in any of the three first parts. If I would have to give a chronology, Part I contains the problems I was first interested in as I came across 2-intervals as early as during my PhD thesis (actually as a naive attempt to build an abstract model for autocatalytic group I introns). Part II and Part IV follow. My interest in comparative genomics (Part III) is more recent; as far as I remember my first research activity in this field dates back to 2005. Let me now introduce briefly these four parts (to facilitate access to the individual topics, the chapters are rendered as self-contained as possible).

Part I is concerned with families of high-dimensional intervals, linear graphs, permutations and arc-annotated sequences, all those graph-like combinatorial objects I can draw from left to right, align and search for a pattern in. It is composed of three chapters. Chapter 1 is devoted to algorithmic aspects of high-dimensional intervals and more specifically to algorithmic aspects of 2-intervals. This chapter encompasses recognition of restricted 2-interval graphs and combinatorial problems on families of 2-intervals. Chapter 2 focuses on linear graphs and linear matchings, those graph with linearly ordered vertices. This chapter can be seen as a follow-up of Chapter 1 as most questions could have been raised in the general framework of 2-intervals (for disjoint interval ground sets). However, as we shall see, linear graphs and linear matchings deserve a separate chapter as they raise specific and important (and hard!) questions about permutation patterns. In Chapter 3, we consider some algorithmic issues of arc-annotated sequences, a popular object to represent RNA sequences. Common to these three chapters is the notion of relative positioning: for any two disjoint objects, either one precedes the other, is included in the other, or they are crossing each over. I have tried to develop in this manuscript a general and common framework (including notations) that encompasses 2-intervals, linear graphs and arc-annotated sequences. Clearly, this is the part of my research that was the most followed [Li and Li, 2006, 2009a; Jiang, 2007b, 2008, 2007a; Chen et al., 2007a; Gramm, 2004a,b; Gramm et al., 2002; P. Thébault et al., 2006].

Part II is devoted to pattern matching issues (in the broad sense) in graphs. It is composed of three chapters. Chapter 4 is concerned with pattern matching in the common sense: finding an exact or an approximate occurrence of a motif (given in the form of a graph) in a target graph. We focus in Chapter 4 on edge-conservation and injective mappings. With protein-protein interaction (PPI) networks in mind, we consider additional restrictions: (i) each vertex of the pattern is associated to a (small) set of vertices of the target graph it can be mapped to, and (ii) both the motif and the target graphs are vertex-colored and any vertex of the motif must be mapped to a vertex of the target graph with the same color. We do believe that a better approach would consist in using a set of colors instead of one color (thereby allowing for a greater flexibly in the design) but we will not develop this point in this manuscript. Chapter 5 differs from Chapter 4 by renouncing to topology conservation (this is actually a weak renouncement as we shall see), we only require the occurrences to be connected. This recent problem (introduced in the context of metabolic networks [Lacroix et al., 2006]) raises new, elegant and original questions. Finally, brief Chapter 6 is devoted to presenting our contribution to a somewhat more classical view of pattern matching in PPI networks where one is allowed to insert and delete vertices in the occurrence. Most of the interest in our contribution (based on feedback vertex sets) is the PADA1 software that performs as well as QNet (the state-of-the-art software to query PPI networks) on tree patterns while allowing for general graph patterns (the tree decomposition based approach of QNet for dealing with general graph patterns has never been implemented due to its complexity).

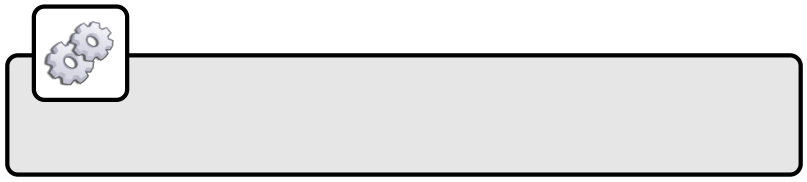
Part III is concerned with comparative genomics. Comparative genomics is a field by itself and we shall only consider genome rearrangements with duplicate genes. It is composed of two chapters. Chapter 7 is by far the longest of the two. In this chapter we consider the problem of computing a distance (or a (dis)similarity) between two genomes with duplicate genes from a pure algorithmic point of view. As we shall see, most – not to say all - problems are intractable and sometimes even hard to approximate within any ratio. Again, I have tried in this chapter to develop a common general framework (in the form of permutations associated to matchings) that encompasses all these problems and allows me give a unified exposition. Chapter 8 is devoted to presenting and analyzing a simple LCS-like problem that aims at overcoming the difficulties I have raised in Chapter 7.

Part IV is actually concerned with two different topics. Chapter 9 is devoted to algorithmic aspects of selenocysteine insertion and could be seen as a follow-up of [Backofen et al., 2002] where the problem of computing an mRNA sequence of maximum codon-wise similarity to a given mRNA (and hence, to a given protein) that additionally satisfies some secondary structure constraints was introduced. Chapter 10

is devoted to a covering problem. My initial motivation for studying this problem came from a paper by Bodlaender *et al.* [Bodlaender et al., 1995], who described an application in the context of protein folding (the authors actually referred to this problem as the DICTIONARY GENERATION problem). Indeed, many proteins seem to be composed of relatively small regions which fold independently of other regions, and the theory of *exon shuffling* proposes that all proteins are concatenations of such regions, where the regions are drawn from a common ancestral dictionary [Dorit and Gilbert, 1991; Patthy, 1991].

To avoid any confusion, the citations of external items (algorithms, theorem, ...) appear in the body of the text as references whereas my results are – most of time, I have tried to stick to this rule as much as possible – given in the form of propositions. Notable exceptions for this rule are pages 28 (a lemma by Noga Alon) and 101 (the Local-Ratio lemma [Bar-Yehuda and Even, 1985]).

Finally, all along this document, I use *thinking notes* or *perspective notes* (sometimes also referred to as *headache notes* in the text for obvious reasons) in the following form



to point out facts, important questions or even perspectives. It is not about all problems or special cases left open in this manuscript (I would have to put such a note on each page I guess), but to shed light on points I am particularly interested in. Therefore, it is worth keeping in mind that these notes are concerned with both problems I have spent weeks (sometimes months) on ... without much success, and perspectives for further research.

Last point, some new results are announced in this manuscript, most without proof. Two notable exceptions are Proposition 2.4.1 (page 20) and Proposition 10.3.2 (page 103).

Part I

Structures: from 2-intervals to annotated sequences . . . throught permutations

Introduction

This part is devoted to presenting our works on (not so) linear structures. Well, what are those (not so) linear structures? In our context these will be all those combinatorial objects that I can draw from left to right, align and search for a pattern in. More specifically, we will be concern with high-dimensional intervals, linear graphs, linear matchings, permutations and arc-annotated sequences. The rationale for bringing together these combinatorial objects is a common notion of relative positioning I am particularly interested in: for any two disjoint objects, either one precedes the other, one is included in the other, or they are crossing each over. My interest in such a property started with 2-intervals. It turns out that this property is also at the heart of the algorithmic of linear graphs and arc-annotated sequences. This manuscript gave me the opportunity to develop a general and common framework (including notations) that encompasses 2-intervals, linear graphs and arc-annotated sequences: an family of objects is type $\mathcal{M}$ if any two objects in it are comparable for a binary relation in $\mathcal{M}$.

High-dimensional (or multi-dimensional, or d-interval) intervals are the union of disjoint intervals, and a multi-dimensional interval graph is the intersection graph of a family of multi-dimensional intervals. Multi-dimensional intervals together with multi-dimensional interval graphs constitute a natural generalization of intervals and interval graphs (one of the most studied class of intersection graphs). We shall be interested mainly on 2-intervals, *i.e.*, unions of pairs of disjoint intervals. Our concern is twofold: recognition of some restricted 2-interval graphs and algorithmic aspects of 2-intervals.

Linear graphs are graphs with linearly ordered vertices. These graphs certainly constitute a special case of 2-intervals. Adopting the same strategy as for 2-intervals, we will be concerned with finding motifs in linear graphs. This general problem includes both finding an occurrence of a linear graph in another linear graph and finding a common motif (a linear graph here) that occurs in each input linear graph. Of particular importance, linear graphs are a generalization of permutations, and hence a portion of this chapter will be devoted to the problem of finding motifs in permutations. Indeed, as we shall see, hardness of finding motifs in linear graphs (and in 2-intervals) originates from permutations.

Arc-annotated sequences can be seen both as a generalization of standard sequences (a string together with some edges) and as a special case of linear graphs (a vertex-labeled linear graph). Arc-annotated sequences have recently proved to be useful for modeling RNA structures. Again, we will adopt the very same strategy as for 2-intervals and linear graphs.

1

Algorithmic aspects of 2-interval sets

Contents

1.1 Introduction

Let $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ be a family of sets. The *intersection graph* of $\mathcal{F}$, usually denoted $\Omega(\mathcal{F})$, is the graph having $\mathcal{F}$ as vertex set with S_i adjacent to S_j if and only if $i \neq j$ and $S_i \cap S_j = \emptyset$ ([McKee and McMorris, 1999] and [Brandstädt et al., 1999] are our favorite references here). A graph G is an intersection graph if there exists a family $\mathcal{F}$ such that $\Omega(\mathcal{F}) \simeq G$, where we typically display this isomorphism by writing $\mathbf{V}(G) = \{u_1, u_2, \ldots, u_n\}$ with each u_i corresponding to S_i and $\{u_i, u_j\} \in \mathbf{E}(G)$ if and only if $S_i \cap S_j \neq \emptyset$. When $\Omega(\mathcal{F}) \simeq G$, the family $\mathcal{F}$ is called a *representation* if G. Notice that every graph is an intersection graph (this property is ascribed to Marczewski in [McKee and McMorris, 1999]) Therefore, while every graph has a set representation, intersection graph theory uses properties of the set representations and various conditions imposed thereon, rather than the conventional graph-theoretic approach that, in some sense, forgets the sets.

We shall be concerned in this chapter with *high-order intervals* (also referred as *multidimensional intervals*). Notice, however, that, whereas we will present all definitions in the general setting of d-dimensional intervals, most of out our concern will be 2-dimensional intervals. The term "d-*dimensional interval*"originated in the late 1970s [Trotter and Harary, 1979; Griggs and West, 1979; Scheinerman and West, 1983], where the focus was on determining how small d can be so that a given graph is a d-interval graph. The first references devoted to algorithmic aspects of d-interval graphs are [Golumbic, 1980] (actually in the form of an exercise) and [West and Shmoys, 1984]. For an up-to-date survey of the algorithmic aspects of 2-intervals, we refer the reader to our recent entry in the Encyclopedia of Algorithms [Vialette, 2008].

1.2 Bestiary and definitions

Let us start by presenting some intersection graphs we will be concerned with in this manuscript (d-box graphs are presented for the sake of completeness – as another way to generalize interval graphs – but we shall not consider them in the sequel).

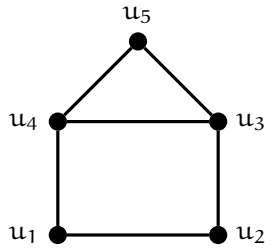
A d-*interval* (or *multiple interval*) is a set of the real line which can be written as the union of d disjoint closed intervals $[a_i, b_i]$. Clearly, 1-intervals are the intervals. The intersection graph of a family of d-intervals is a d-*interval graph*. The smallest d for which G is a d-interval graph is the *interval number* $i(G)$.

A d-*track interval* is a union of d intervals, one each from d parallel lines (actually separate lines would be a better definition as, for example, defining piercing sets for d-track intervals by vertical lines is a bit confusing if they are defined on parallel lines). A graph is a d-*track interval graph* if it is the intersection graph of d-track intervals. The intervals graphs are precisely the 1-track interval graphs (and also the 1-interval graphs). The *multitrack interval number* of a graph G is the smallest d for which G is a d-track interval graph. Notice that a d-track interval graph is the union of d interval graphs with the same vertex set.

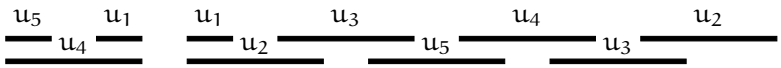
Closely related are d-boxes and d-box graphs. A d-*box* is the Cartesian product of intervals $[a_i, b_i]$, $1 \leq i \leq d$. A graph is a d-*box graph* if it is the intersection graph of d-boxes. Hence interval graphs are precisely the 1-box graphs. Of interest in our context, a d-box graph is the intersection of d interval graphs with the same vertex set. Notice that d-box graphs are not contained in d-interval graphs, neither d-interval graphs are included in d-box graphs. Indeed, $K_{3,6}$ is a 2-box graph but not a 2-interval graph, and the graph obtained by subdivising edge each of K_5 is a 2-interval graph but not a 2-box graph. The *boxicity* of a graph G is the minimum d for which G is a d-box graph. We shall not develop d-box graphs in the sequel.

For a d-interval (resp, d-track interval, d-box) graph G, a d-*interval* (resp, d-*track interval*, d-*box*) *representation* of G is a family of d-intervals (resp, d-track intervals, d-boxes) $\mathcal{F}$ for which G is the intersection graph of.

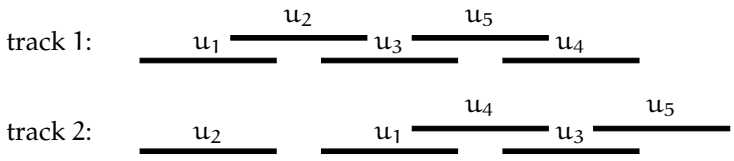
***Example* 1** Let G be the graph defined as follows:



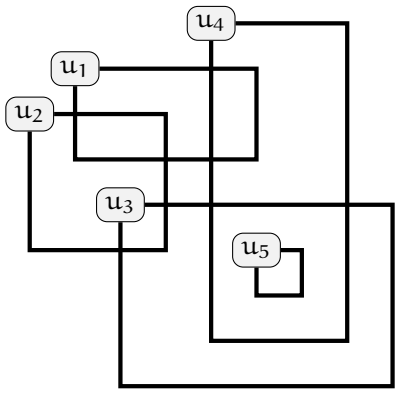
A 2-interval representation of G is given by



a 2-track interval representation of G is given by:



and a 2-box representation of G is given by:



For algorithmic considerations, convenient d-intervals are needed. For the sake of brevity, we define restricted d-interval object but the same definitions do apply in a natural way for d-track intervals as well.

A d-interval $I = (I_1, I_2, \ldots, I_d)$ is *balanced* if $|I_1| = |I_2| = \ldots = |I_d|$. Notice that this restriction has been introduce for RNA considerations [Crochemore et al., 2008]. A *balanced* d-*interval graph* is the intersection graph of a family of balanced d-intervals.

A d-interval $I = (I_1, I_2, \ldots, I_d)$ is *unit* if it is composed of d intervals of length 1. A *unit* d-*interval graph* is the intersection graph of a family of unit d-intervals. Clearly, unit d-interval graphs are balanced d-interval graphs whereas the converse is not necessarily true.

A d-interval $I = (I_1, I_2, \ldots, I_d)$ with integer endpoints is *type* $(l_1, l_2, \ldots, l_d)$ if $|I_i| = l_i$ for all $1 \leq i \leq d$. A d-*interval graph type* $(l_1, l_2, \ldots, l_d)$ is the intersection graph of a family of d-intervals type $(l_1, l_2, \ldots, l_d)$. Notice that unit d-intervals are d-intervals type $(1, 1, \ldots, 1)$ and that d-intervals type $(l, l, \ldots, l)$, $l \in \mathbb{N}^*$, are balanced d-intervals. We can also notice that 2-interval graphs type $(1, 1)$ are exactly line graphs: each interval of length 1 of the ground set can be considered as the vertex of a root graph and each 2-interval as an edge in the root graph. This implies, for example, that the coloration problem is also **NP**-complete for 2-interval graphs type $(2, 2)$ and wider classes of graphs. It is also known that the complexity of the MAXIMUM INDEPENDENT SET problem is **NP**-complete on 2-interval graphs type $(2, 2)$ [Bafna et al., 1996]. Recognition of 2-union graphs type $(1, 2)$, a related class (restriction of multitrack interval graphs), has been also proved to be **NP**-complete [Halldórsson and Karlsson, 2006].

The *depth* of a family of d-intervals is the maximum number of intervals that share a common point. The *representation depth* of a d-interval graph is the minimum depth of any d-interval representation of the graph. Notice that any d-interval (or d-track interval) representation of a triangle-free graph must have depth at most 2. On the other hand, for any constant $d \geq 2$, it is easy to construct a d-interval (or d-track interval) representation of depth 2 of a triangle.

1.3 Recognizing multidimensional interval graphs

In this section, we shall mostly focus on $d = 2$. We study some restrictions of 2-interval graphs, and their position in the hierarchy of graph classes as illustrated Figure 1.1.

Recognizing restricted graph classes in an ubiquitous problem in intersection graph theory, and indeed there has been considerable interest in recognizing d-interval graphs (and related graph classes). The first explicit reference to this question we are aware of is in [Golumbic, 1980]. A classical result of West and Shmoys [West and Shmoys, 1984] states that, for any constant $d \geq 2$, recognizing d-interval graphs is **NP**-complete (moreover, for any constants $d \geq 2$ and $r \geq 3$, recognizing d-interval graphs of representation depth at most r is also **NP**-complete). The class of d-track interval graphs is clearly contained in the class of d-interval graphs. Notice that the containment is proper as the complete bipartite graph $K_{d^2+d-1,d+1}$ is a

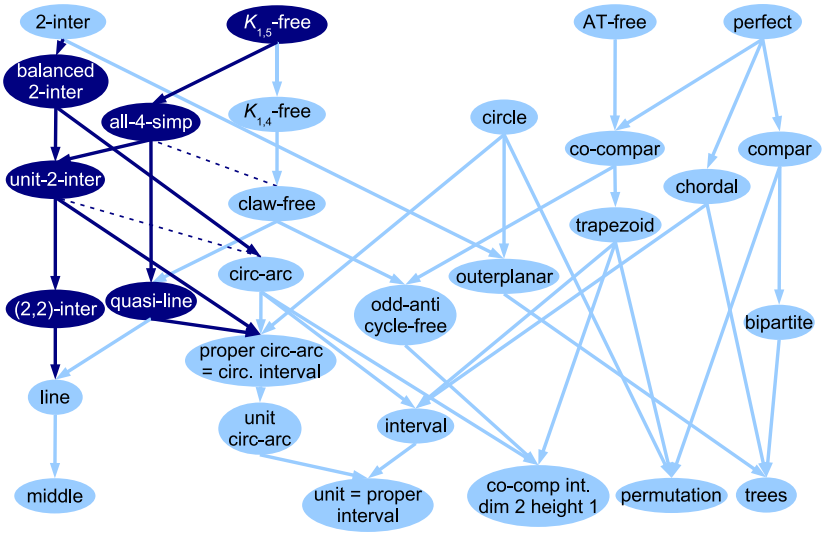


Figure 1.1: Graph classes related to 2-interval graphs and its restrictions. A class pointing towards another strictly contains it, and the dashed lines mean that there is no inclusion relationship between the two. Dark classes correspond to classes not yet present in the ISGCI Database.

.

d-interval graph but is not a d-track interval graph [West and Shmoys, 1984]. Gyárfás and West [Gyárfás and West, 1995] have however proved that recognizing 2-track interval graphs is **NP**-complete (their proof also implies that, for any constant $r \geq 3$, recognizing 2-track interval graphs of representation depth at most r is **NP**-complete). It is still an open problem (but conjectured to be true) to prove that, for any constant $d \geq 2$, recognizing d-track interval graph is **NP**-complete ... To be honest, the problem is not really open any longer as M. Jiang has recently communicated us a – correct as far as we can assess – **NP**-hardness proof for this problem.

Our contributions for balanced 2-interval graphs is two-fold. We have shown that the class of balanced 2-interval graphs is strictly included in the class of 2-interval graphs. The rationale for this question was concerned with approximation: does any approximation result for balanced 2-intervals propagate to (general) 2-intervals? The answer is No, unfortunately. Moreover, we have settled the complexity of recognizing balanced 2-interval graphs.

Proposition 1.3.1 ([Gambette and Vialette, 2007]). *The class of balanced 2-interval graphs is strictly included in the class of 2-interval graphs.*

Our proof is by exhibiting a 2-interval graph that has no balanced 2-interval realization (this latter point being of course the hardest part of the proof). Without going into the details, the construction is by connecting a bunch of gadgets $K_{5,3}$ (the complete bipartite graph $K_{5,3}$ is indeed not a unit 2-interval graph) together with additional vertices to enforce an unbalanced 2-interval representation. Notice that we also proved that the class of balanced 2-interval graphs strictly contains circular-arc graphs (see [Brandstädt et al., 1999] for definitions).

Proposition 1.3.2 ([Gambette and Vialette, 2007]). *Recognizing balanced 2-interval graphs is* **NP***-complete.*

To prove Proposition 1.3.2, we have adapted the proofs of [West and Shmoys, 1984] and [Gyárfás and West, 1995], and gave a reduction from the HAMILTONIAN CYCLE problem for 2-regular triangle-free graphs, a problem which has been proved to be **NP**-complete in [West and Shmoys, 1984]. Moreover, it is easy enough

to check that Gyárfás and West's proof of **NP**-hardness of recognizing 2-track interval graphs [Gyárfás and West, 1995] can be adapted, by adjusting the interval lengths in the representation (more or less as we did for 2-interval graphs [Gambette and Vialette, 2007]), to show that recognizing balanced 2-track interval graphs is **NP**-complete as well.

As for 2-interval graphs type (l, l), we have obtained the following results.

Proposition 1.3.3 ([Gambette and Vialette, 2007]). *For any $l \in \mathbb{N}^*$, $l \geq 2$, the class of 2-interval graphs type (l, l) is strictly contained in the class of 2-interval graphs type $(l+1, l+1)$.*

Proper containment of 2-interval graphs type (l, l) in 2-interval graphs type $(l+1, l+1)$ is illustrated in Figure 1.2.

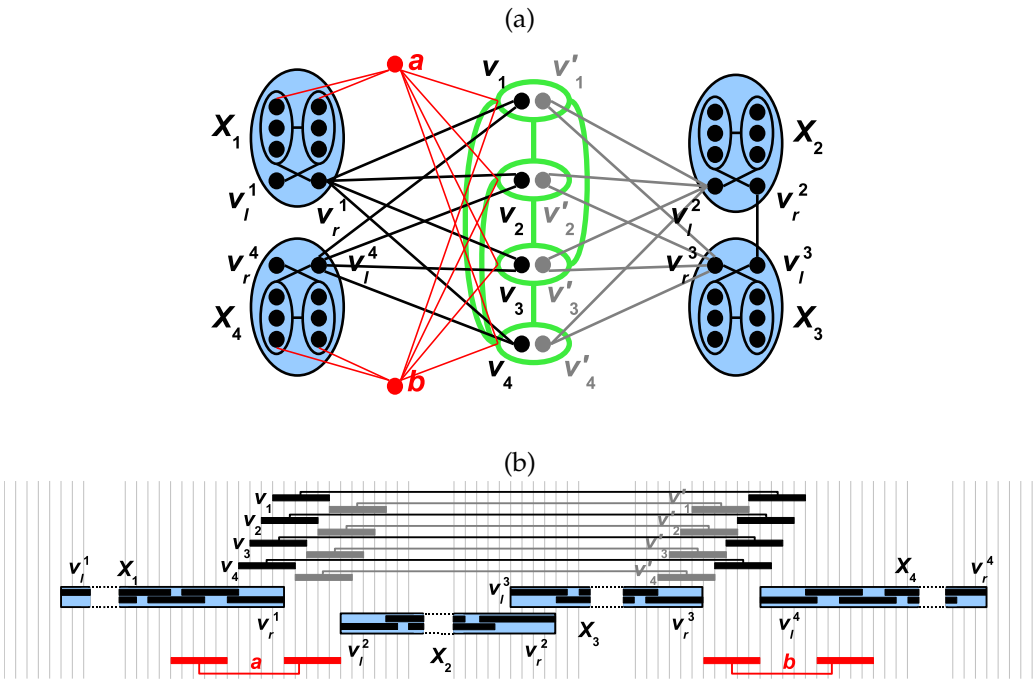


Figure 1.2: The graph K_4' (a) is (5,5)-interval but not (4,4)-interval.

Proposition 1.3.4 ([Gambette and Vialette, 2007]).

$$\{\textit{unit 2-interval graphs}\} = \bigcup_{l \in \mathbb{N}^*} \{\textit{2-interval graphs type } (l, l)\}.$$

According to Proposition 1.3.4, if recognizing 2-interval graphs type (l, l) is polynomial-time solvable for any $l \in \mathbb{N}^*$, then recognizing unit 2-interval graphs is polynomial-time solvable. This problem has not been settled yet.

Aiming at further deciphering the precise nature of unit 2-interval graphs, we have obtained the following inclusion between proper circular-arc graphs (circular-arc graphs such that no arc is included in another in the representation) and 2-interval graphs (recall that circular-arc graphs are balanced 2-interval graphs but that circular-arc graphs are not necessarily unit 2-interval graphs).

Proposition 1.3.5 ([Gambette and Vialette, 2007]). *The class of proper circular-arc graphs is strictly included in the class of unit 2-interval graphs.*

Determining the complexity of recognizing unit 2-interval graphs is still an open problem. More generally, what is the complexity of recognizing d-interval graphs type $(2, 2, \ldots, 2)$? The question is also open for 2-interval graphs type $(1, 1)$. A first step could be to focus on 2-track interval graphs type $(1, 1)$. Indeed, 2-track interval graphs type $(1, 1)$ are subclasses on unit 2-interval graphs.

In [Gambette and Vialette, 2007], we have considered a class of graphs that generalizes quasi-line graphs and contains unit 2-interval graphs. *Quasi-line graphs* are those graphs whose vertices are bisimplicial, *i.e.*, the closed neighborhood of each vertex is the union of two cliques. This graph class has been introduced as a generalization of line graphs and as a useful subclass of claw-free graphs [Ben Rebea, 1981; Faudree et al., 1997; Chudnovsky and Seymour, 2005; King and Reed, 2007]. Let $k \in \mathbb{N}^*$. A graph G is *all-k-simplicial* if the neighborhood of each vertex $u \in \mathbf{V}(G)$ can be partitioned into at most k cliques. The class of quasi-line graphs is thus exactly the class of all-2-simplicial graphs.

Proposition 1.3.6 ([Gambette and Vialette, 2007])**.** *The class of unit 2-interval graphs is strictly included in the class of all-4-simplicial graphs.*

1.4 Combinatorial problems on 2-intervals

1.4.1 Introduction

Multiple-interval graphs are a natural generalization of interval graphs, and hence there is a natural interest in studying standard combinatorial problems for multiple-intervals (and multiple-interval graphs, the distinction is important since computing a multiple-interval representation of a graph is **NP**-complete). Of particular interest, three standard graph problems, namely MINIMUM VERTEX COVER, MINIMUM DOMINATING SET and MAXIMUM CLIQUE, for d-intervals are considered in [Butman et al., 2007]. Their results can be summarized as follows: the MINIMUM VERTEX COVER problem is approximable within ratio $(2 - 1/d)$ (a ratio which equals the best known ratio for $2d1$ bounded degree graphs), the MINIMUM DOMINATING SET problem is approximable within ratio d^2, and the MAXIMUM CLIQUE problem is approximable within ratio $(d^2d + 1)/2$.

We present here two contributions in this area. First, we discuss the 2-INTERVAL PATTERN problem which can be seen as a generalization of the MAXIMUM INDEPENDENT SET for 2-intervals. Standard complexity and approximation are considered. Second, we consider parameterized issues of some natural combinatorial problems for 2-intervals. Parameterized issues of the 2-INTERVAL PATTERN problem are part of an ongoing work with S. Guilemot and D. Hermelin, we shall only mention them briefly.

1.4.2 The 2-INTERVAL PATTERN problem

The 2-INTERVAL PATTERN problem is concerned with finding large constrained patterns in families of 2-intervals. Given a single-stranded RNA molecule, a sequence of contiguous bases of the molecule can be represented as an interval on a single line, and a possible pairing between two disjoint sequences can be represented as a 2-interval, which is merely the union of two disjoint intervals. Therefore, 2-interval representation considers thus only the bonds between the bases and the pattern of the bonds, such as hairpin structures, knots and pseudoknots. A maximum cardinality pairwise disjoint subfamily of a candidate family

Does there exist a polynomial-time algorithm for finding a maximum cardinality clique in a family of 2-intervals? In other words, given an family of 2-intervals $\mathcal{F} = \{D_1, D_2, \ldots, D_n\}$, is the problem of finding a maximum cardinality subset $\mathcal{F}' \subseteq \mathcal{F}$ of pairwise intersecting 2-intervals in **P**? The MAXIMUM CLIQUE problem in its natural decision setting is **NP**-complete for families of 3-intervals [Butman et al., 2007], approximable within ratio $(d^2 - d + 1)/2$ for families of d-intervals (and hence within ratio 3/2 for families of 2-intervals) [Butman et al., 2007], and fixed-parameter tractable for families of d-intervals for parameters d and k (k is the size of the clique we are looking for) [Fellows et al., 2007].
The MAXIMUM CLIQUE problem for families of d-intervals is (or is likely to be) strongly related to the *interval piercing number*. Let $\mathcal{F}$ be a family of d-intervals. A *piercing set* for $\mathcal{F}$ is a set of points P on the real line such that, for any d-interval $D \in \mathcal{F}$, $D \cap P \neq \emptyset$. The piercing number of $\mathcal{F}$, denoted $\tau(\mathcal{F})$, is the size of a minimum cardinality piercing set of $\mathcal{F}$. Gyárfás has proved that, for any family of pairwise intersecting 2-intervals $\mathcal{F}$, it holds that $\tau(\mathcal{F}) \leq 3$. [Gyárfás and Lehel, 1970] (see also [Gyárfás, 2003]).
A stronger result holds for families of 2-track intervals and has proved to be useful. Indeed, if $\mathcal{F}$ is a set of pairwise intersecting 2-track interval set, then $\tau(\mathcal{F}) \leq 2$ and there is a piercing set $\{p_1, p_2\}$ of $\mathcal{F}$ with p_1 on track one and p_2 on track two [Gyárfás and Lehel, 1970] (see also [Gyárfás, 2003]). Starting from this property, we can prove that the MAXIMUM CLIQUE problem for 2-track intervals is in **P** (D. Hermelin, R. Rizzi and S. Vialette, Unpublished result). As another step towards determining the complexity of the MAXIMUM CLIQUE problem for 2-intervals, we announce the following result: The MAXIMUM CLIQUE problem for 3-track intervals is **APX**-hard (D. Hermelin, M. Jiang and S. Vialette, Unpublished result).

of 2-intervals restricted to certain prespecified geometrical constraints can provide useful valid approximation for RNA secondary structure determination. Therefore, the geometric properties of 2-intervals provide a possible guide for understanding the computational complexity of finding structured patterns in RNA sequences. Using a model to represent non sequential information allows us for varying restrictions on the complexity of the pattern structure. Indeed, two disjoint 2-intervals, *i.e.*, two 2-intervals that do not intersect in any point, can be in precedence order ($<$), be allowed to nest ($\sqsubset$) or be allowed to cross ($\between$). Furthermore, the family of 2-intervals and the pattern can have different restrictions, *e.g.*, all intervals have the same length or all the intervals are disjoint. These different combinations of restrictions alter the computational complexity of the problems, and need to be examined separately. This examination produces efficient algorithms for more restrictive structured patterns, and hardness results for those less restrictive.

Let $I = [a, b]$ be an interval on the line. Write $\text{start}(I) = a$ and $\text{end}(I) = b$. A 2-interval is the union of two disjoint intervals defined over a single line and is thus denoted by $D = (I, J)$, I is completely to the left of J. Write $\text{left}(D) = I$ and $\text{right}(D) = J$. Two 2-intervals $D_1 = (I_1, J_1)$ and $D_2 = (I_2, J_2)$ are said to be *disjoint* (or *non-intersecting*) if the two 2-intervals share no common point, *i.e.*, $(I_1 \cup J_1) \cap (I_2 \cup J_2) = \emptyset$. For such disjoint pairs of 2-intervals, three natural binary relations, denoted $<$, $\sqsubset$ and $\between$, are of special interest (these three relations will be recurrent in the first part of this manuscript):

- $D_1 < D_2$ (D_1 *precedes* D_2), if $I_1 \prec J_1 \prec I_2 \prec J_2$,
- $D_1 \sqsubset D_2$ (D_1 is *nested* in D_2), if $I_2 \prec I_1 \prec J_1 \prec J_2$, and
- $D_1 \between D_2$ (D_1 *crosses* D_2), if $I_1 \prec I_2 \prec J_1 \prec J_2$,

where $\prec$ denotes the usual precedence between (1-)intervals.

A pair of 2-intervals D_1 and D_2 is said to be R-*comparable* for some $R \in \{<, \sqsubset, \between\}$, if either $D_1 R D_2$ or $D_2 R D_1$, *i.e.*, D_1 and D_2 are comparable by R. Note that any two disjoint 2-intervals are R-comparable for some $R \in \{<, \sqsubset, \between\}$ (good, we shall not miss something). A family of pairwise disjoint 2-intervals $\mathcal{F}$ is said to be *type* $\mathcal{M}$ for some $\mathcal{M} \subseteq \{<, \sqsubset, \between\}$, $\mathcal{M} \neq \emptyset$, if any pair of distinct 2-intervals in $\mathcal{F}$ is R-comparable for some $R \in \mathcal{M}$. The non-empty subset $\mathcal{R}$ is usually called a *model* for $\mathcal{F}$. It is implicitly assumed here that $\mathcal{R}$ is as small as possible.

Given a family of 2-intervals, the 2-INTERVAL PATTERN problem asks to find in a maximum cardinality subset of pairwise compatible 2-intervals. In the present context, compatibility denotes the fact that any two 2-intervals in the solution are (i) non-intersecting and (ii) satisfy some prespecified geometrical constraints. The 2-INTERVAL PATTERN problem is formally defined as follows.

2-INTERVAL PATTERN

- **Input** : A family of 2-intervals $\mathcal{F}$ and a model $\mathcal{M} \subseteq \{<, \sqsubset, \between\}$.
- **Solution** : A subfamily $\mathcal{F}' \subseteq \mathcal{F}$ (of pairwise disjoint 2-intervals) type $\mathcal{M}$.
- **Measure** : The size of $\mathcal{F}'$, *i.e.*, $|\mathcal{F}'|$.

Some additional definitions are needed for further algorithmic analysis. Let $\mathcal{F}$ be a family of 2-intervals. The *width* (resp. *height*, *depth*) is the size of a maximum cardinality subset $\mathcal{F}' \subseteq \mathcal{F}$ type $\{<\}$ (resp $\{\sqsubset\}$, $\{\between\}$). The *interleaving distance* of a 2-interval $D_i \in \mathcal{F}$ is defined to be the distance between the two intervals of D_i, *i.e.*, $\mathrm{start}(\mathrm{right}(D_i)) - \mathrm{end}(\mathrm{left}(D_i))$. The *total interleaving distance* of the family of 2-intervals $\mathcal{F}$, written $\mathcal{L}(\mathcal{F})$, is the sum of all interleaving distances, *i.e.*, $\mathcal{L}(\mathcal{F}) = \sum_{D_i \in \mathcal{F}} \mathrm{start}(\mathrm{right}(D_i)) - \mathrm{end}(\mathrm{left}(D_i))$. The *density* of $\mathcal{F}$, written $d(\mathcal{F})$, is the maximum number of 2-intervals in $\mathcal{F}$ over a single point. Formally,

$$d(\mathcal{F}) = \max_{x \in X(\mathcal{F})} \{D \in \mathcal{F} : \mathrm{end}(\mathrm{left}(D) \leq x < \mathrm{start}(\mathrm{right}(D))\}.$$

The structure of the set of all (simple) intervals involved in a family of 2-intervals $\mathcal{F}$ turns out to be of particular importance for algorithmic analysis of the 2-INTERVAL PATTERN problem. The *interval ground set* of $\mathcal{F}$, denoted $\mathcal{I}(\mathcal{F})$, is the set of all intervals involved in $\mathcal{F}$, *i.e.*, $\mathcal{I}(\mathcal{F}) = \{\mathrm{left}(D_i) : D_i \in \mathcal{F}\} \cup \{\mathrm{right} < (D_i) : D_i \in \mathcal{F}\}$. In [Vialette, 2004; Crochemore et al., 2008], we have introduced four types of interval ground sets:

- **UNLIMITED**: no restriction on the structure,
- **BALANCED**: each 2-interval $D_i \in \mathcal{F}$ is composed of two intervals having the same length, *i.e.*, $|\mathrm{left}(D_i)| = |\mathrm{right}(D_i)|$,
- **UNIT**: the interval ground set $\mathcal{I}(\mathcal{F})$ is solely composed of unit length intervals,
- **DISJOINT**: no two distinct intervals in the interval ground set $\mathcal{I}(\mathcal{F})$ intersect.

Recall that family of unit 2-intervals is balanced, while the converse is not necessarily true. Furthermore, for most applications, one may assume that a family of pairwise disjoint 2-intervals is unit. Observe that in this latter case, a family of 2-intervals reduces to a graph G equipped with a numbering of its vertices from 1 to $|V|$ (see Chapter 2 for a complete treatment of this restriction). Considering additional restrictions such as (i) bounding the width, the height or the depth of either the input family of 2-intervals or the solution subset, or (ii) bounding the interleaving distances are also of interest for practical applications.

Before going to algorithms, I would like to take the opportunity of this manuscript to clarify one point about 2-intervals I never explicitly stated. This is the right place I guess. Indeed, I was asked many times whether 2-intervals (and actually the same question would make sense for linear graphs) would ever yield any competitive algorithm for RNA structure prediction. The answer is No, and actually I did not ever think the answer could be Yes. Biology is of course too complicated for such a simplistic solution ("*Biology easily has 500 years of exciting problems to work on, it's at that level*", D. Knuth, Computer Literacy Bookshops Interview, 1993). Nobody would think that 2-intervals and linear graphs are accurate models for RNA structure, ever! There are so many parameters here, so many exceptions, so many exceptions to exceptions, ... "*Parfois, mais pas toujours, oui, non, enfin parfois, a dpend, pas toujours, non, voil, pas toujours*"would say Jean-Pierre Rousset. But this is precisely for this reason that I believe simple enough combinatorial objects are needed to deal with specific issues involved in the big picture (I have for example in mind the algorithmic impact of crossing structures). This was my guideline for introducing and studying 2-intervals in computational biology.

1.4.3 Algorithms and complexity

Let us start with some easy observations and statements. For one, the 2-INTERVAL PATTERN problem for $\mathcal{M} = \{<, \sqsubset, \between\}$ is related to the MAXIMUM INDEPENDENT SET problem in 2-interval graphs with a given 2-interval representation (recall that, as we have seen, recognizing 2-interval graphs, and hence computing a 2-interval representation, is **NP**-complete). For another, graphs of maximum degree Δ are $\lceil(\Delta+1)/2\rceil$-interval graphs [Griggs and West, 1979], and hence any graph with maximum degree 3 is a 2-interval graph. Therefore, since the MAXIMUM INDEPENDENT SET problem in its natural decision setting is **NP**-complete for planar graphs with maximum degree 3 [Garey et al., 1976] (we note in passing that any planar graph is a 3-interval graph [Scheinerman and West, 1983]), it follows that the 2-INTERVAL PATTERN problem is **NP**-complete in its whole generality. This is actually not very surprising (but we know what one is letting oneself in for).

The best complexity results for the 2-INTERVAL PATTERN problem are given in Table 1.3 for various models and interval ground sets, and we shall only discuss some very specific points in the sequel.

First, the $O(n\log(n)+\mathcal{L})$ time algorithm of [Chen et al., 2007a] for $\mathcal{M} = \{\sqsubset, \between\}$ and disjoint interval ground set now supersedes our $O(n^2\log(n))$ time algorithm [Blin et al., 2007c]. However, the techniques are quite comparable and are based on the following property. For a 2-interval D, define its *covering interval* to be $c(D) = [\text{start}(\text{left}(D)), \text{end}(\text{right}(D))]$, *i.e.*, the least interval that covers D. Let $\mathcal{F}$ be a family of 2-intervals and let G be the intersection graph of the intervals $\{c(D) : D \in \mathcal{F}\}$; G is certainly an interval graph. Moreover, any subfamily $\mathcal{F}' \subseteq \mathcal{F}$ type $\{\sqsubset, \between\}$ induces a clique in G. It is thus enough to focus on the maximal cliques of G. But an interval graph G is a chordal graph and as such has at most $|\mathbf{V}(G)|$ maximal cliques [Fulkerson and Gross, 1965]. Furthermore, all the maximal cliques of a chordal graph can be found in $O(n+m)$ time, where $n = |\mathbf{V}(G)|$ and $m = |\mathbf{E}(G)|$, by a modification of Maximum Cardinality Search (MCS) [Tarjan and Yannakakis, 1984; Blair and Peyton, 1993]. The $O(n^2\log(n))$ time algorithm follows.

Second, if the 2-INTERVAL PATTERN problem is solvable in $O(n\log(n))$ time for both $\mathcal{M} = \{<\}$ (the algorithm is trivial) and $\mathcal{M} = \{\sqsubset\}$, the two algorithms actually use different geometrical objects: intervals for the former and trapezoids [Felsner et al., 1997] for the latter (a structure which will prove extremely useful in the sequel).

Model $\mathcal{M}$	Interval Ground Set $\mathcal{I}(\mathcal{F})$	
	Unlimited, Balanced, Unit	Disjoint
$\{<, \sqsubset, \between\}$	**APX**-hard [Bar-Yehuda et al., 2002]	$O(n\sqrt{n})$ [Micali and Vazirani, 1980]
$\{<, \between\}$	**NP**-complete [Blin et al., 2007c]	**NP**-complete [Li and Li, 2009a]
$\{\sqsubset, \between\}$	**APX**-hard [Vialette, 2004]	$O(n \log(n) + \mathcal{L})$ [Chen et al., 2007a]
$\{<, \sqsubset\}$	$O(n \log(n) + nd)$ [Chen et al., 2007a]	
$\{<\}$	$O(n \log(n))$ [Vialette, 2004]	
$\{\sqsubset\}$	$O(n \log(n))$ [Blin et al., 2007c]	
$\{\between\}$	$O(n \log(n) + \mathcal{L})$ [Chen et al., 2007a]	

Figure 1.3: Best complexity results for the 2-INTERVAL PATTERN problem for all combinations of models and interval ground sets. For the polynomial-time cases, $n = |\mathcal{F}|$, $\mathcal{L} = \mathcal{L}(\mathcal{F})$ and $d = d(\mathcal{F})$.

Model $\mathcal{M}$	Interval Ground Set $\mathcal{I}(\mathcal{F})$			
	Unlimited	Balanced	Unit	Disjoint
$\{<, \sqsubset, \between\}$	4 [1] [Bar-Yehuda et al., 2002]	4 [2] [Crochemore et al., 2008]	3 [2] [Bar-Yehuda et al., 2002]	N/A
$\{\sqsubset, \between\}$	4 [1] [Bar-Yehuda et al., 2002]	4 [3] [Crochemore et al., 2008]	3 [3] [Crochemore et al., 2008]	N/A
$\{<, \between\}$	PTAS [Jiang, 2007b] (or effective 2 [4] [Jiang, 2007a])			

Figure 1.4: Performance ratios for hard instances of the 2-INTERVAL PATTERN problem (the 2-INTERVAL PATTERN problem for disjoint interval ground set and models $\mathcal{M} = \{<, \sqsubset, \between\}$ and $\mathcal{M} = \{\sqsubset, \between\}$ is polynomial-time solvable).

1.4.4 Approximation

The best approximation ratio for the 2-INTERVAL PATTERN problem are given Figure 1.4 for various models and interval ground sets, and, once again, we shall only discuss specific points.

First, we paid special attention to efficient approximation algorithms, and most of the results presented Figure 1.4 support implementation. However, the 4-approximation for unlimited 2-intervals is by linear programming and we are still not able to design a simple and practical approximation algorithm with the very same performance ratio.

Second, the PTAS of Jiang [Jiang, 2007b] for $\mathcal{M} = \{<, \between\}$ supersedes our results [Crochemore et al., 2008], *i.e.*, an approximation ratio (i) 6 for general 2-intervals, (ii) 4 for balanced 2-intervals, (iii) 3 for unit 2-intervals, and (iv) 2 when the 2-intervals reduce to a linear graph (see next chapter). It is worth noticing that, like most PTAS, Jiang's algorithm does not support implementation (however, Jiang has proposed an approximation algorithm with performance ratio 2 well-suited for practical applications).

Third, we considered in [Crochemore et al., 2008] a weighted variant of the 2-INTERVAL PATTERN problem: each 2-interval is associated to a weight and the goal is to find a maximum weight subfamily of pairwise disjoint 2-intervals with respect to a prespecified model $\mathcal{M}$. Here, one can for instance weight a 2-interval by the total sum of the lengths of its intervals, thereby allowing more refined solutions in the biological application of the problem. We have shown in [Crochemore et al., 2008] that our results can be extended to the weighted variant, while still maintaining the same approximation factors.

1.4.5 Parameterized complexity

We have considered in [Hermelin et al., 2009] the parameterized issues of some standard combinatorial problems restricted to d-intervals. It is understood here that, even if the considered problems are defined for graphs, we consider these problems on families of 2-intervals in terms of the associated 2-interval graphs. For example, the MINIMUM DOMINATING SET problem (given a graph G, find a minimum cardinality set of

vertices $V' \subseteq \mathbf{V}(G)$ such that any vertex in $\mathbf{V}(G) \setminus V'$ has at least one neighbor in V') reduces for a family of 2-intervals $\mathcal{F}$ to finding a subfamily of 2-intervals $\mathcal{F}' \subseteq \mathcal{F}$ such that any 2-interval in $\mathcal{F} \setminus \mathcal{F}'$ intersects at least one 2-interval in $\mathcal{F}'$. Our results (negative and positive) can be summarized as follows.

According to Proposition 1.4.3, the 2-INTERVAL PATTERN problem for $\mathcal{M} = \{\sqsubset, \between\}$ is **W[1]**-hard for its standard parameterization The parameterized complexity (standard parameterization) of the 2-INTERVAL PATTERN problem for $\mathcal{M} = \{<, \between\}$ is more intriguing. So far, we are still not able to determine the parameterized complexity of this problem. Recall that the 2-INTERVAL PATTERN problem for $\mathcal{M} = \{<, \between\}$ has a polynomial-time approximation scheme (PTAS) [Jiang, 2007b], and hence proving that it is **W[1]**-hard would show that, in some sense, a PTAS is the best approximation one can obtain (*i.e.*, no *efficient PTAS*) for this problem.

Conjecture 1. *The* 2-INTERVAL PATTERN *problem for* $\mathcal{M} = \{<, \between\}$ *is* **W[1]**-*hard for its standard parameterization.*

Proposition 1.4.1 ([Hermelin et al., 2009])**.** *The following problems are* **W[1]**-*hard for 2-interval graphs (assuming a 2-interval representation is given along with the graph):*

- *the* MAXIMUM INDEPENDENT SET *problem parameterized by the size of the solution,*
- *the* MINIMUM DOMINATING SET *problem parameterized by the size of the solution, and*
- *the* INDEPENDENT MINIMUM DOMINATING SET *problem parameterized by the size of the solution.*

It is worth noticing that [Hermelin et al., 2009] has rapidly become a well-cited paper, not in reason of Proposition 1.4.1 but for the *multicolored clique technique* we have introduced. In a nutshell, the multicolored clique technique allows for an almost systematic gadget-construction and helps in eliminating several technical details (see [Hermelin et al., 2009] where a large portion is devoted to presenting this general technique).

Proposition 1.4.2 ([Hermelin et al., 2009])**.** *The* MAXIMUM CLIQUE *problem for* d-*intervals is fixed-parameter tractable when parameterized by* d *and* k *(*k *is the size of the clique we are looking for).*

Central in the proof of Proposition 1.4.2 is the fact a d-interval graphs with no clique of size k has a vertex of degree less than $2k$. However, the algorithm is of limited practical interest due to the huge exponential term in the running time, *i.e.*, $O\left(k^2 \binom{2dk}{k}\right)$. Notice, however, that Jiang has recently proposed a better a $\max\{d^{O(k)}, 2^{O(klogk)}\} \cdot \text{poly}(n)$ time algorithm for MAXIMUM CLIQUE problem for d-intervals, where n is the number of vertices in the graph Jiang [2010]. Designing a *practical* fixed-parameter algorithm for the CLIQUE problem for d-intervals remains a challenging problem.

We conclude this chapter by discussing briefly parameterized issues of the 2-INTERVAL PATTERN problem. First, according to Proposition 1.4.1 for $\mathcal{M} = \{<, \sqsubset, \between\}$ is **W[1]**-hard for its standard parameterization. To complement this result, the following proposition is announced with proof.

Proposition 1.4.3. *The* 2-INTERVAL PATTERN *problem for* $\mathcal{M} = \{\sqsubset, \between\}$ *is* **W[1]**-*hard for its standard parameterization*

2 From linear graphs to permutations

Contents

2.1 Introduction

This chapter is devoted to algorithmic aspects of linear graphs and is a natural follow-up of Chapter 1 as linear graphs can be viewed as families of 2-intervals over a disjoint ground set (this was actually our initial motivation for studying linear graphs). In Section 2.2 we set up notation and terminology. Section 2.3 is devoted to introducing the relationship between permutations and linear matchings. The three following sections are devoted to algorithmic considerations: In Section 2.4 we consider the pattern matching for permutations problem, Section 2.5 is concerned with finding large common patterns in linear graphs whereas Section 2.6 aims at bringing together common patterns and permutations.

2.2 Definitions

We follow standard notations in graph theory (see for example [Diestel, 2000]). The *order* (resp. *size*) of a graph G is defined as the number of vertices (resp. edges) of G.

Definition 2.2.1 (Linear graph)**.** *A* linear graph *of order* n *is a vertex-labeled graph where each vertex is labeled by a distinct label from* $\{1, 2, \ldots, n\}$.

It is worth mentioning that we shall always assume in this chapter that a linear graph has no degree 0 vertices (see also note Page 30).

A linear graph can be thus viewed as a graph with vertices embedded on the integral line, yielding a total order amongst them. In case of linear graphs, we write an edge between vertices i and j, $i < j$, as the pair (i, j). By convention, if G is a linear graph, we let $G[i \ldots j]$, $1 \leq i \leq j \leq |\mathbf{V}(G)|$, denote the subgraph induced by all vertices labeled k with $i \leq k \leq j$. Two edges of a linear graph are *disjoint* if they do not share a common vertex. Of particular interest in our context are edge-disjoint linear graphs.

Definition 2.2.2 (Linear matching)**.** *A* linear matching *is an edge-disjoint linear graph.*

A linear matching with $2n$ vertices (should be indeed even) has thus n edges. Similarly to 2-intervals, relative positioning of disjoint edges is of particular interest. Let $e = (i, j)$ and $e' = (i', j')$ be two disjoint edges in a linear graph or a linear matching G. We write:

- $e < e'$ (e *precedes* e') if $i < j < i' < j'$,
- $e \sqsubset e'$ (e is *nested* in e') if $i' < i < j < j'$, and
- $e \between e'$ (e and e' *cross*) if $i < i' < j < j'$.

Two edges e and e' are R-*comparable*, for some $R \in \{<, \sqsubset, \between\}$, if eRe' or $e'Re$. For a subset $\mathcal{M} \subseteq \{<, \sqsubset, \between\}$, $\mathcal{M} \neq \emptyset$, edges e and e' are said to be $\mathcal{M}$-*comparable* if e and e' are R-comparable for some $R \in \mathcal{M}$. A set of edges E is $\mathcal{M}$-*comparable* if any pair of distinct edges $e, e' \in E$ are $\mathcal{M}$-comparable. A linear matching whose edge set is $\mathcal{M}$-comparable is said to be *type* $\mathcal{M}$. A *subgraph* of a linear graph G is a linear graph H which can be obtained from G by a series of vertex and edge deletions, where the deletion of vertex i results in removing vertex i and all edges incident to it from the graph, and then relabeling all vertices j with $j > i$ to $j - 1$. In our context, an edge-disjoint subgraph of a linear graph is also called a *structured pattern*.

2.3 From linear graphs to permutations . . . and back

It is folklore that linear matchings type $\{<, \sqsubset, \between\}$ of order $2n$ are in bijection with fixed-point free (fpf) involutions, *i.e.*, permutations of $\mathfrak{S}_{2n}$ with n cycles, each of length 2. The number of fpf involutions of $\mathfrak{S}_{2n}$ is the double factorial number $(2n-1)!! = 1 \cdot 3 \cdots (2n-1)$ (see the *On-Line Encyclopedia of Integer Sequences* for references).

It is also a simple observation that linear matchings type $\{\sqsubset, \between\}$ of size n are in bijection with permutations of $\mathfrak{S}_n$. To see this, let us consider a linear matching G type $\{\sqsubset, \between\}$ of size n. Then the vertices in G which are left endpoints of edges are labeled $\{1, 2, \ldots, n\}$ and the right endpoints are labeled $\{n+1, n+2, \ldots, 2n\}$. The permutation π_G corresponding to G is defined by $\pi_G(j - n) = i$ if and only if $(i, j) \in \mathbf{E}(G)$. Clearly, all linear matchings type $\{\sqsubset, \between\}$ have corresponding permutations, and vice versa. It follows from this bijective correspondence that the number of different linear matchings type $\{\sqsubset, \between\}$ of G of size n is $n!$. Interestingly enough, notice that increasing subsequences in π_G correspond to subgraphs type $\{\between\}$ of G, while decreasing subsequences correspond to subgraphs type $\{\sqsubset\}$. See Figure 2.1 for an illustration.

Both linear matchings type $\{<, \sqsubset\}$ and $\{<, \between\}$ of order $2n$ are in bijection with Dyck words of length $2n$ (should we call a linear matching type $\{<, \between\}$ an *anti-Dyck* pattern?). Recall that a Dyck word of length $2n$ is a string consisting of n a's and n b's such that no initial segment of the string has more b's than a's (for example, the following are the Dyck words of length 6: $aaabbb$, $abaabba$, $ababab$, $aabbab$, $aababb$). The number of Dyck words of length $2n$ is n-th Catalan number $C_n = \binom{2n}{n}/(n+1)$ (the best general reference here is [Stanley, 1999]).

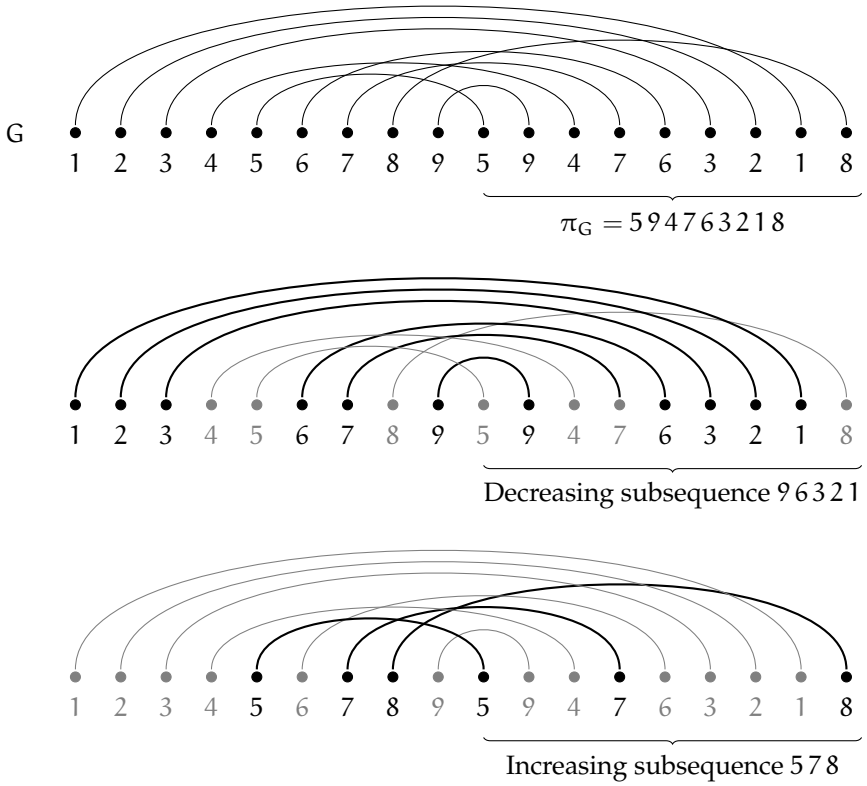


Figure 2.1: A linear matching G type $\{\sqsubset, \not\wr\}$ and the corresponding permutation $\pi_G = 594763218$. Also illustrated is the bijective correspondence between decreasing subsequences (resp. increasing subsequences) of π_G and patterns of G type $\{\sqsubset\}$ (resp. type $\{\not\wr\}$).

2.4 Pattern matching

We consider in this section the pattern matching problem for linear matchings: Given a pattern (in the form of a linear matching) and a target linear matching, decide whether there is an occurrence of the motif in the target. We refer to this problem as the PATTERN MATCHING FOR LINEAR MATCHINGS problem. According to the preceeding section, if both the pattern and the target are linear matching type $\{\sqsubset, \not\wr\}$ we are left with the pattern matching problem for permutations (the bijection is indeed pattern-preserving). We refer to this later problem as the PERMUTATION PATTERN problem. As we shall see, most of the difficulties in trying to solve the PATTERN MATCHING FOR LINEAR MATCHINGS problem originate from the PERMUTATION PATTERN problem.

Let us embed the PATTERN MATCHING FOR LINEAR MATCHINGS problem into permutations. A permutation π is said to *contain* the pattern (shorter permutation) σ, in symbols $\sigma \preceq \pi$, if there exists a subsequence of entries of π that has the same relative order as σ (alternatively, σ is involved in π). Otherwise, π avoids σ. For example, 3215674 contains the pattern 132 since the subsequence 154 is ordered in the same way as 132. Pattern involvement in permutations has become a very active area of research. For one, pattern containment restrictions are often used to describe classes of permutations that are sortable under various conditions [Knuth, 1973]. For another, a great deal of study has been devoted to counting pattern-avoiding permutations [Bóna, 2004], probably culminating in the proof of the Stanley-Wilf conjecture [Marcus and

Tardos, 2004].

Does there exist a fixed-parameter algorithm (standard parameterization) for finding an occurrence of a permutation $\sigma \in \mathfrak{S}_k$ in a permutation $\pi \in \mathfrak{S}_n$? In other words, does there exist an algorithm for finding an occurrence of σ in π is $f(k)n^{O(1)}$ time, where f is an arbitrary function depending only on k?
We thought for ages that the answer should be just No, unless **FPT** = **W**[**1**]. We changed our mind radically about this issue. Every attempt to design a parameterized reduction leaded us to – more or less – the same cul-de-sac.

Conjecture 2. *The pattern matching for permutations problem is fixed-parameter tractable for its standard parameterization.*

We, however, do believe that proving this conjecture is quite a difficult difficult task . . . far beyond the reach of our arms for the time being.

Given two permutations σ and π, the PERMUTATION PATTERN problem is thus to decide whether $\sigma \preceq \pi$ (this problem is ascribed to H. Wilf in [Bose et al., 1998]). The PERMUTATION PATTERN problem is **NP**-hard [Bose et al., 1998] (see [Vialette, 2004] for an alternate proof), but is clearly polynomial-time solvable if σ has bounded size. Indeed, if σ has size k and π has size n, a straightforward brute-force algorithm solves the problem in $O(n^k)$ time. Improvements to this algorithm were presented in [Albert et al., 2001] and [**?**], the latter describing a $O(n^{0.47k+o(k)})$ time algorithm. Also, the problem is known to be polynomial-time solvable (in k and n) if σ is *separable*, *i.e.*, σ contains neither the pattern 2413 nor 3142 [Bose et al., 1998; Ibarra, 1997]. In case σ is monotone, *i.e.*, $\sigma = 1 \dots k$ or $\sigma = k \dots 1$, a nice $O(n \log \log(n))$ time algorithm is known [Hunt and Szymanski, 1977a].

Is the PERMUTATION PATTERN problem fixed-parameter tractable for its standard parameterization? If only one question were to be asked in this manuscript this has to be this one. We still don't have any answer here. Could it be the case that the PERMUTATION PATTERN problem is a special case of a more general problem in **FPT**? Or in other words, does there exist a **FPT** proof for free? Since the PERMUTATION PATTERN problem is a special case of the general PATTERN MATCHING FOR LINEAR MATCHINGS problem, one can naturally reduces this question (in our context) to: is the PATTERN MATCHING FOR LINEAR MATCHINGS fixed-parameter tractable for its standard parameterization? We answer this latter question in the negative (unless **FPT** = **W**[**1**], a fairly unexpected event) by proving the following new result (observe that this does not, however, rule out the existence of another simpler problem in **FPT** containing the PERMUTATION PATTERN problem . . . hence the main question remains asked).

Proposition 2.4.1. *The* PATTERN MATCHING FOR LINEAR MATCHINGS *problem is* **W[1]***-hard for its standard parameterization,* i.e, *the size of the pattern we are looking for.*

Here is a sketch of the proof.

Proof. We propose a parameterized reduction from the CLIQUE problem which is known to be **W[1]**-hard when parameterized by the size of the clique we are looking for [Downey and Fellows, 1999].

Let (G, k) be an arbitrary instance of the CLIQUE problem. Write $\mathbf{V}(G) = \{u_1, u_2, \dots, u_n\}$ and $\mathbf{E}(G) = \{e_1, e_2, \dots, e_m\}$. Furthermore, let us write d_i for the degree of $u_i \in \mathbf{V}(G)$, and for convenience let $d_0 = 0$. For each $1 \leq i \leq n$, write D_i for $d_1 + d_2 + \ldots + d_{i-1}$. Finally, for $1 \leq i, j \leq n$, write $l_{i,j}$ for the number of

The following items might be useful tools towards proving fixed-parameter tractability of the PERMUTATION PATTERN problem (Join work with S. Guillemot).
The all 1's $k \times k$ binary matrix is denoted J_k. Let $A = [a_{i,j}]$ be a $m \times n$ binary matrix. It is said to be *pruned* if it contains neither a all 0's row nor a all 0's column. For now on, we assume A is pruned. A $(p/m, q/n)$-partition P is (i) a partition of $\{1, 2, \ldots, m\}$ into p intervals $R_1, R_2, \ldots, R_p$, and (ii) a partition of $\{1, 2, \ldots, n\}$ into q intervals $C_1, C_2, \ldots, c_q$. The *quotient* of A by P, in symbols A/P, is the $p \times q$ binary matrix $A/P = [a/p_{i,j}]$ defined by $a/p_{i,j} = 1$ if and only if $a_{k,l} = 1$ for some $k \in R_i$ and $l \in C_j$. Given two pruned binary matrices A and B of size $m \times n$ and $p \times q$, respectively, we say that B is *contained* in A, denoted by $B \preceq A$, if there exists a $(p/m, q/n)$-partition P such that $B \leq A/P$ (this latter notation means that $a/p_{i,j} = 1$ whenever $b_{i,j} = 1$ for $1 \leq i \leq p$ and $A \leq j \leq q$) .
For a $m \times n$ pruned binary matrix $A = [a_{i,j}]$, it will be convenient to define ones(A) as follows

$$\text{ones}(A) = \{(i, j) \in \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} : a_{i,j} = 1\}.$$

For any $e = (i, j) \in \text{ones}(A)$ and $e' = (i', j') \in \text{ones}(A)$, define the *distance* between e and e', denoted $d_A(e, e')$, by $d_A(e, e') = \max\{|i - i'|, |j - j'|\}$. The matrix A is k-*locally-dense* if there exists distinct $e, e' \in \text{ones}(A)$ such that $d_A(e, e') \leq k$ (by convention A is k-dense if $|\text{ones}(A)| \leq 1$). Define the *local-density* of A, simply denoted $d(A)$, to be the minimum k for which A is k-locally-dense.

Conjecture 3. *For any permutation matrix π, if $d(\pi) \geq k$ then $J_k \preceq \pi$.*

Although at first odd, no counter-example has yet been found. Notice that the above conjecture holds for $k = 2$ since non-separable permutations contain 3142 or 3142 and hence does contain J_2. It also holds for $|\pi| = k^2$ (details omitted). Finally, observe that, if True, the above conjecture would imply $D(\pi) \leq \Delta(\pi)$, where $D(\pi) = \max\{d(\sigma) : \sigma \preceq pi\}$ and $\Delta(\pi)$ is the maximum k for which $J_k \preceq \pi$ holds.

neighbors u_x of u_i such that $x < j$, *i.e.*,

$$l_{i,j} = |\{u_x : \{u_i, u_x\} \in \mathbf{E}(G) \ \wedge \ x < j\}|.$$

We construct the corresponding instance $(G_{\text{target}}, G_{\text{pattern}})$ of the PATTERN MATCHING FOR LINEAR MATCHINGS problem as follows. The linear matching G_{target} has order $4n + 8m + 4$ and its edge set $\mathbf{E}(G_{\text{target}})$ is defined by

$$\mathbf{E}(G_{\text{target}}) = E^1_{\text{target}} \cup E^2_{\text{target}} \cup E^3_{\text{target}} \cup E^4_{\text{target}} \cup E^5_{\text{target}} \cup E^6_{\text{target}}$$

where

$$E^1_{target} = \{(i, 4m + 4n + 4i + 1) : 1 \leq i \leq m\}$$
$$E^2_{target} = \{(i + 1, 4m + 4n + 4i + 4) : 1 \leq i \leq m\}$$
$$E^3_{target} = \{(2m + 2i - 1, 2m + 2n + 2i + D_i + 1) : 1 \leq i \leq n\}$$
$$E^4_{target} = \{(2m + 2i, 2m + 2n + 2i + D_{i+1} + 2) : 1 \leq i \leq n\}$$
$$E^5_{target} = \{(2m + 2n + 1, 2m + 2n + 2), (4m + 4n + 3, 4m + 4n + 4)\}.$$

Let us now describe E^6_{target} which is the only part of G_{target} that depends on the input (apart from m and n of course). We add two edges to E^6_{target} for each edge of G. More precisely, if $e_i = \{u_p, u_q\}$, $p < q$, is an edge of G, we add to E^6_{target} the two edges $(2m + 2n + 2i + D_p + l_{p,q} + 2, 4m + 4n + 4i + 2)$ and $(2m + 2n + 2i + D_p + l_{q,p} + 2, 4m + 4n + 4i + 2)$. This completes the construction of G_{target}.

We now turn to constructing $G_{pattern}$. The linear matching $G_{pattern}$ has order $4k^2 + 4$ (depending solely of k, good!) and its edge set $\mathbf{E}(G_{target})$ is defined by

$$\mathbf{E}(G_{pattern}) = E^1_{pattern} \cup E^2_{pattern} \cup E^3_{pattern} \cup E^4_{pattern} \cup E^5_{pattern} \cup E^6_{pattern}$$

using the very same construction as for G_{target} but considering as input the complete graph K_k on k vertices (and $\frac{1}{2}k(k-1)$ edges) instead of G.

It can be proved that there is an occurrence of $G_{pattern}$ in G_{target} if and only if G has a clique of size k. One direction is trivial (by construction). For the other direction, we only mention that the following observation is crucial for correctness (and for reducing the proof to a sequence of easy steps): the two edges of $E^5_{pattern}$ must match the two edges of E^5_{target}. Indeed, a careful observation of $G_{pattern}$ shows that, for any two edges $e_i, e_j \in \mathbf{E}(G_{pattern})$, $e_i < e_j$ if and only if $e_i = (2m+2n+1, 2m+2n+2)$ and $e_j = (4m+4n+3, 4m+4n+4)$. Similarly, for any two edges $e_i, e_j \in \mathbf{E}(G_{target})$, $e_i < e_j$ if and only if $e_i = (2m + 2n + 1, 2m + 2n + 2)$ and $e_j = (4m+4n+3, 4m+4n+4)$. This property allows us to draw the following crucial property: for $1 \leq i \leq 6$, all edges of $E^i_{pattern}$ are matched to edges in E^i_{target}. From this point, the rest of the proof is just a sequence of easy readings of the construction. □

In the light of the present situation (the parameterized complexity of the PERMUTATION PATTERN for its natural parameterization is still open), we have considered in [Guillemot and Vialette, 2009] the PERMUTATION PATTERN problem in case σ (possibly σ and π) avoids a pattern of length 3. Recall that Knuth proved in [Knuth, 1998] that for all six of the patterns of length 3 it is true that the number of permutations of size n that avoid the pattern is the Catalan number $C_n = \binom{2n}{n}/(n+1)$. First, it is easy to see that the PERMUTATION PATTERN problem is polynomial-time solvable if the pattern σ avoids 132, 312, 213 or 231 since σ is clearly separable in this case. Monotone patterns, *i.e.*, 123 and 321, however, deserve separate consideration (we focus here on 321-avoiding permutations but if a permutation avoids 123 then its reverse avoids 321). The rest of this section is devoted to presenting our results.

First, combining ordered forest embeddings with labeled DAG morphisms, we have shown that the PERMUTATION PATTERN problem is polynomial-time solvable if both π and σ are 321-avoiding.

Proposition 2.4.2 ([Guillemot and Vialette, 2009]). *In case both π and σ are 321-avoiding, the* PERMUTATION PATTERN *problem is solvable in $O(k^2 n^6)$ time, where $k = |\sigma|$ and $n = |\pi|$.*

Second, if we relax the problem to only one 321-avoiding permutation (and it has to be σ of course), we have obtained the following result.

Proposition 2.4.3 ([Guillemot and Vialette, 2009]). *In case only σ (the pattern) is 321-avoiding, the* PERMUTATION PATTERN *problem is solvable in $O(kn^{4\sqrt{k}+12})$ time, where $k = |\sigma|$ and $n = |\pi|$.*

Notice that the above proposition does not settle the complexity of the PERMUTATION PATTERN problem in case only σ is 321-avoiding. In [Guillemot and Vialette, 2009], we have conjectured this problem to be **NP**-complete. Unfortunately (or fortunately, we were not wrong!) this conjecture is true as shown in the following proposition (definitively too long proof omitted).

Proposition 2.4.4. *The* PERMUTATION PATTERN *problem is* **NP***-complete even if* σ *is* 321*-avoiding.*

We close this section by mentioning a generalization of the PERMUTATION PATTERN problem that may be of independent interest. The c-COLORED PERMUTATION PATTERN problem is defined as follows: given two permutations σ and π, and a *stair decomposition* D of σ (see [Atkinson et al., 2005] and [Guillemot and Vialette, 2009] for details), where σ and π are c-colored permutations (*i.e.*, a color in $\{1, 2, \ldots, c\}$ is associated to each point of the permutation), find a color-preserving embedding of σ into π. We have obtained the following result (recall that the Exponential-Time Hypothesis (ETH) is the assumption that the 3-SAT problem cannot be solved in $2^{o(n)}$ time, where n is the number of variables).

Proposition 2.4.5 ([Guillemot and Vialette, 2009])**.** *The* 2-COLORED PERMUTATION PATTERN *problem parameterized by* k *is* **W[1]***-hard and cannot be solved in* $n^{o(\sqrt{k})}$ *time assuming ETH.*

We also refer the reader to [Guillemot and Vialette, 2009] for **WNL**-hardness issues of Proposition 2.4.5 (the parameterized class **WNL** was introduced in [Guillemot, 2008] to capture the parameterized complexity of problems solvable by k-dimensional dynamic programming).

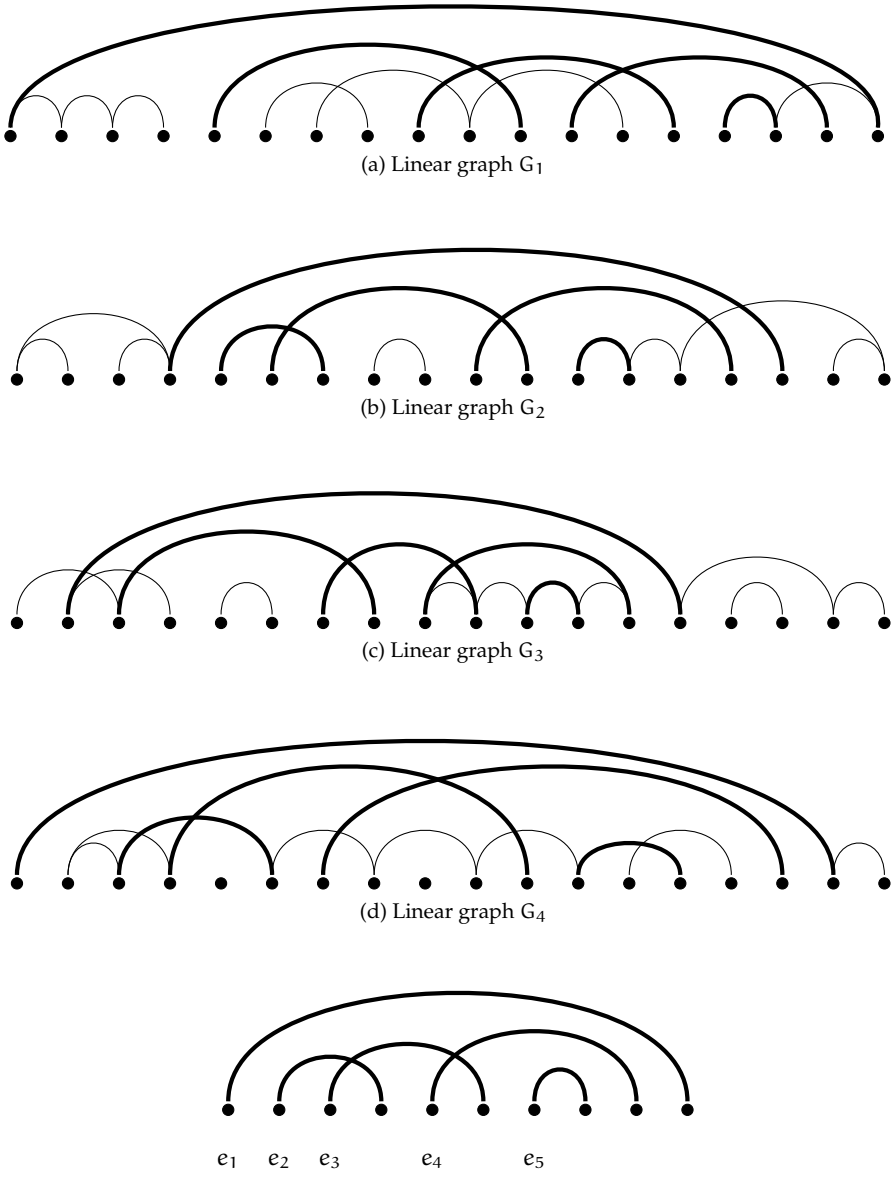
2.5 Finding common structures

2.5.1 Introduction

This section is devoted to finding common structures in linear graphs and linear matchings. We begin by presenting this problem in its original setting. RNA and proteins exhibit a three-dimensional structure that determines most of their functionality. This three dimensional structure can be modeled (at the price of simplifications!) in two dimensions by a linear graphs. The corresponding structure-similarity or structure-prediction problems that arise in such contexts usually translate to finding common linear matching subgraphs, or common *structured patterns*, that occur in a family of general linear graphs. Examples of such problems are the LONGEST COMMON SUBSEQUENCE problem [Hirschberg, 1977; Hunt and Szymanski, 1977b], the MAXIMUM COMMON ORDERED TREE INCLUSION problem [Alonso and Schott, 1993; Chung, 1998; Kilpeläinen and Mannila, 1995], the ARC-PRESERVING SUBSEQUENCE problem [Blin et al., 2005b; Evans, 1999c; Gramm et al., 2002], and the MAXIMUM CONTACT MAP OVERLAP problem [Goldman et al., 1999] problem (more on this in the perspective note Page 30). A general framework for such problems is known as the e MAXIMUM COMMON STRUCTURED PATTERN (MCSP) problem.

The MCSP problem was originally introduced (under a different name) by Davydov and Batzoglou [Davydov and Batzoglou, 2006] in the context of non-coding RNA secondary structure prediction via multiple structural alignment. There, an RNA sequence of n nucleotides is represented by a linear graph with n vertices, and an edge connects two vertices if and only if their corresponding nucleotides are complementary. A family of linear graphs is then used to represent a family of functionally-related RNAs, and a common structured pattern in such a family is considered to be a putative common secondary structure element of the family.

The MAXIMUM COMMON STRUCTURED PATTERN (MCSP) problem is formally defined as follows (see Figure 2.2 for an illustrative example).



(a) Linear graph G_1

(b) Linear graph G_2

(c) Linear graph G_3

(d) Linear graph G_4

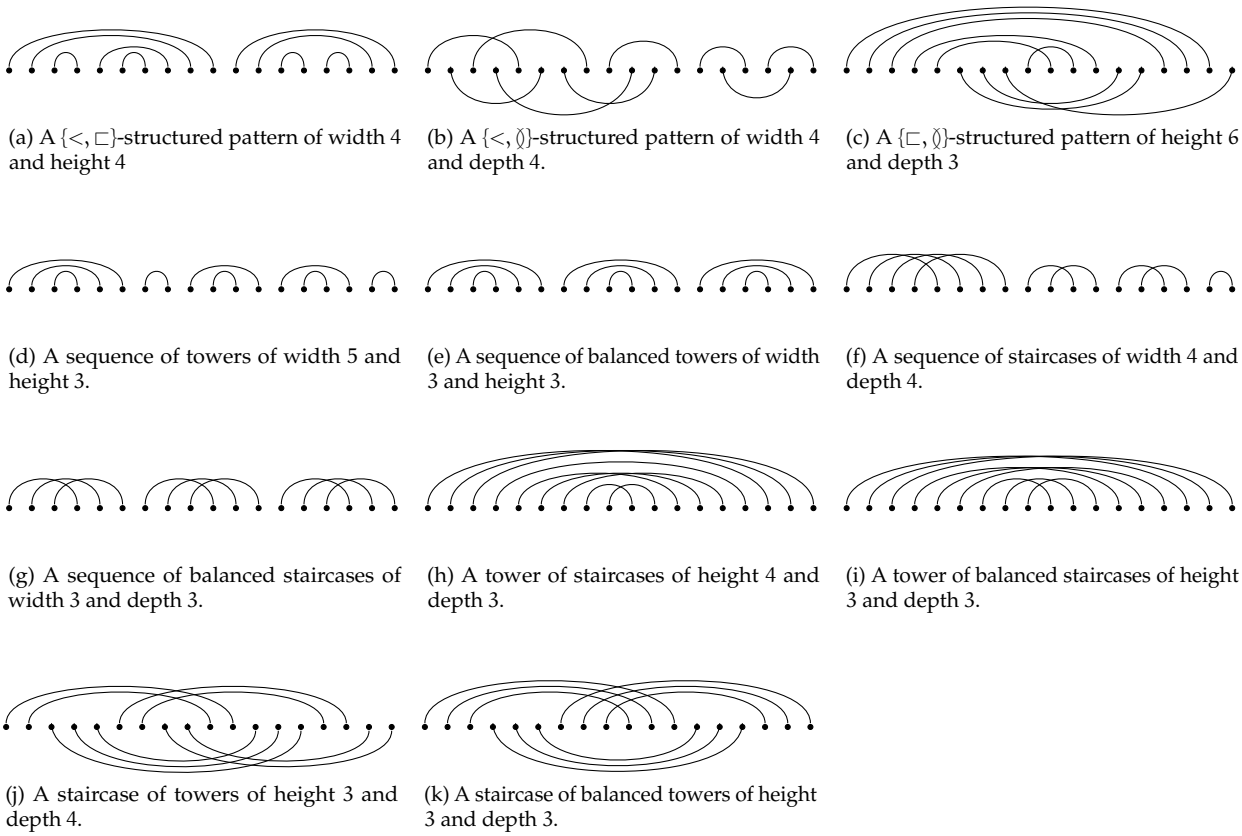
(e) Linear matching G_{sol}

Figure 2.2: Four linear graphs G_1, G_2, G_3 and G_4 and a common structured pattern (depicted as G_{sol}. The occurrence of the structured pattern G_{sol} in each graph is emphasized in bold. Edges e_2, e_3, e_4 and e_5 are nested in edge e_1 ; edges e_2 and e_3 precede edge e_5 ; edge e_2 precedes edge e_4 and crosses edge e_3, while edge e_3 crosses both edges e_2 and e_4.

MCSP

• **Input** : A family of linear graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ and a non-empty subset $\mathcal{M} \subseteq \{<, \sqsubset, \between\}$.
• **Solution** : A common structured pattern G_{sol} type $\mathcal{M}$ of $\mathcal{G}$, *i.e.*, a linear matching type $\mathcal{M}$ that occurs in each input linear graph of $\mathcal{G}$.
• **Measure** : The size of G_{sol}, *i.e.*, $|\mathbf{E}(G_{sol})|$.

It will be convenient to introduce some special linear matching. A linear matching type $\{<\}$ (resp. $\{\sqsubset\}$, $\{\between\}$) is called a *sequence* (resp. *tower*, *staircase*). Define the *width* (resp. *height*, *depth*) of a linear graph to be the size of a maximum cardinality sequence (resp. tower, staircase) subgraph of the graph. A linear matching type $\{<, \sqsubset\}$ with the additional property that any two maximal towers in it do not share an edge is called a *sequence of towers* . Similarly, a linear matching type $\{<, \between\}$ is a *sequence of staircases* if any two maximal staircases do not share an edge. A *tower of staircases* is a linear matching type $\{\sqsubset, \between\}$ where any pair of maximal staircases do not share an edge, and a *staircase of towers* is a comparable linear matching type $\{\sqsubset, \between\}$ where any pair of maximal towers do not share an edge. A sequence of towers (resp. sequence of staircases, tower of staircases, staircase of towers) is *balanced* if all of its maximal towers (resp. staircases, staircases, towers) are of equal size. Figure 2.3 illustrates an example of the above types of linear graphs.



(a) A $\{<, \sqsubset\}$-structured pattern of width 4 and height 4

(b) A $\{<, \between\}$-structured pattern of width 4 and depth 4.

(c) A $\{\sqsubset, \between\}$-structured pattern of height 6 and depth 3

(d) A sequence of towers of width 5 and height 3.

(e) A sequence of balanced towers of width 3 and height 3.

(f) A sequence of staircases of width 4 and depth 4.

(g) A sequence of balanced staircases of width 3 and depth 3.

(h) A tower of staircases of height 4 and depth 3.

(i) A tower of balanced staircases of height 3 and depth 3.

(j) A staircase of towers of height 3 and depth 4.

(k) A staircase of balanced towers of height 3 and depth 3.

Figure 2.3: Some restricted structured patterns. Edges are drawn above or below the vertices with no particular signification.

The MCSP problem is relatively easy for simple $\mathcal{M}$. Indeed, it is solvable in $O(nm)$ time for $\mathcal{M} = \{<\}$ [Gupta et al., 1982], in $O(nm \log\log(m))$ time for $\mathcal{M} = \{\sqsubset\}$ [Whang and Wang, 1992], and in $O(nm^{1.5})$ time for $\mathcal{M} = \{\between\}$ [Tiskin, 2006], where $n = |\mathcal{G}|$ and $m = \max_{G\in\mathcal{G}} |\mathbf{E}(G)|$.

We briefly review some related results. Valiente gave a dynamic programming algorithm for finding a largest nested linear graph that occurs in two nested linear graphs [Lozano and Valiente, 2004] (see also [Zhang and Shasha, 1989]). In a totally different context, Felsner *et al.* considered the matching problem regardless of precise pattern definition and proved that given a linear graph G of size m, a maximum size

nested subgraph of G can be found in $O(m^2)$ time [Felsner et al., 1997]. The general problem of finding a maximum size edge-independent subgraph of G is the well-known maximum matching problem [Diestel, 2000].

2.5.2 Structured patterns type $\{<, \sqsubset\}$

Of particular importance in the context of computational molecular biology, is the fact that the MCSP problem for structured patterns type $\{<, \sqsubset\}$ has been shown to be **NP**-complete [Davydov and Batzoglou, 2006]. We have strengthen this result in a drastic way.

Proposition 2.5.1 ([Kubica et al., 2006]). *The* MCSP *problem for structured patterns type* $\{<, \sqsubset\}$ *is* **NP***-hard even if each input linear matching is a sequence of towers of height at most* 2*.*

Notice, however, that we have proved the MCSP problem to be polynomial-time solvable in case the number of input linear graphs is a fixed integer [Kubica et al., 2006]. As for the approximation, the MCSP problem for structured patterns type $\{<, \sqsubset\}$ was proved to be approximable with ratio $O(\log^2(k))$ [Davydov and Batzoglou, 2006], where k is the size of an optimal solution. We have improved this result in [Kubica et al., 2006]

Proposition 2.5.2 ([Kubica et al., 2006]). *The* MCSP *problem for structured patterns type* $\{<, \sqsubset\}$ *is approximable within ratio* $O(\log k)$ *in* $O(nm^2)$ *time, where* k *is the size of an optimal solution,* $n = |\mathcal{G}|$*, and* m *is the maximum size of any linear graph in* $\mathcal{G}$*.*

2.5.3 Structured patterns type $\{<, \between\}$

Focusing on $\mathcal{M} = \{<, \between\}$, we have obtained the following results (the first one is a straightforward consequence of Proposition 2.5.1 whereas the second one requires two ingredients: (i) any structured pattern type $\{<, \between\}$ contains a sequence of staircases of substantial size and (ii) any sequence of staircases contains a balanced subgraph of substantial size, details omitted).

Proposition 2.5.3 ([Fertin et al., 2007]). *The* MCSP *problem for structured patterns type* $\{<, \between\}$ *is* **NP***-hard even if each input linear graph is a sequence of staircases of depth at most* 2*.*

Notice that a recent result in [Li and Li, 2009b] implies that the MCSP problem for structured patterns type $\{<, \between\}$ is hard even if $\mathcal{G}$ consists of only two linear graphs. However, the input linear graphs used in [Li and Li, 2009b] are of unlimited structure, unlike Proposition 2.5.3. Interestingly enough, the case $|\mathcal{G}| = 1$ has been recently proved to be **NP**-hard [Li and Li, 2009b].

Proposition 2.5.4 ([Fertin et al., 2007]). *The* MCSP *problem for structured patterns type* $\{<, \between\}$ *is approximable within ratio* $2H(k)$ *in* $O(nm^{3.5}\log(m))$ *time, where* k *is the size of an optimal solution,* $n = |\mathcal{G}|$*,* $m = \max_{G\in\mathcal{G}} |\mathbf{E}(G)|$*, and* $H(k) = \sum_{i=1}^{k} 1/i$ *is the* k*-th harmonic number.*

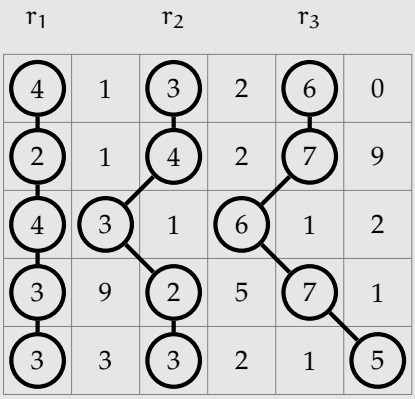
2.5.4 Structured patterns type $\{\sqsubset, \between\}$

Not surprisingly, the MCSP problem for structured patterns type $\{\sqsubset, \between\}$ is hard even for quite simple instance.

Proposition 2.5.5 ([Fertin et al., 2007]). *The* MCSP *problem for structured patterns type* $\{\sqsubset, \between\}$ *is* **NP***-hard even if each input linear graph is a tower of staircases of depth at most* 2*. The same result applies for staircases of towers*

As the reader might have guessed, the above result is an easy consequence of Proposition 2.5.1. As observed before, this case in strongly related to pattern matching for permutations. The well-known Erdős-Szekeres Theorem [Erdős and Szekeres, 1935] states that any permutation of $\mathfrak{S}_k$ contains either an increasing

Proposition 2.5.1 raises the following entertaining problem. Let $A = [a_{i,j}]$ be a $m \times n$ matrix with non-negative entries. Define a *run* r in A to be a mapping $r : m \to n$. The *weight* of the run r is defined by $\omega(r) = \min\{a_{i,r(i)} : 1 \leq i \leq m\}$. Let r_1 and r_2 be two runs of A. The run r_1 *precedes* the run r_2, in symbols $r_1 < r_2$, if $r_1(i) < r_2(i)$ for all $1 \leq i \leq m$. We consider the problem defined as follows: Given a $m \times n$ matrix with non-negative entries, find a sequence of runs $r_1 < r_2 < \ldots < r_k$ with maximum total weight $\sum_{i=1}^{k} \omega(r_i)$. Observe that the number of runs in the solution is not part of the input, one is only interested in maximizing the total weight of the solution. Below is an illustration for a 5×6 matrix with three runs $r_1 < r_2 < r_3$ of total weight $\omega(r_1) + \omega(r_2) + \omega(r_3) = 2 + 2 + 5 = 9$.



According to Proposition 2.5.1, the above problem is **NP**-complete even if every entry of the matrix is one of the integers 0, 1 and 2 (each row denotes a sequence of tower and each entry denotes the height of a tower). Moreover, it is easy to show that the problem is polynomial-time solvable if every entry of the matrix is one of the integers 0 and 1 (Proposition 2.5.1 is tight). How approximable is the general problem? Is it **APX**-hard? We would be very surprised if the answer to this latter question for a fixed number of distinct non-negative integers was Yes. In other words, we believe a PTAS exists for this special case. Notice that the straighforward greedy algorithm (repeatedly select a maximum weight run) does not yield any approximation result. To see this, consider the $m \times m$ matrix $A = [a_{i,j}]$ defined by $a_{i,j} = 2$ if $i = j$ and $a_{i,j} = 1$ otherwise.

According to [Li and Li, 2009b], the PATTERN MATCHING FOR LINEAR MATCHINGS problem is **NP**-complete is the pattern is type $\{<, \emptyset\}$. What about the complexity if both the target graph and the pattern are linear matchings type $\{<, \emptyset\}$?

Conjecture 4. *Finding an occurrence of a linear matching type $\{<, \emptyset\}$ in another linear matching type $\{<, \emptyset\}$ is polynomial-time solvable.*

or a decreasing subsequence of size at least $\sqrt{k}$. It is worth noticing that extremal Erdős-Szekeres (EES) permutations, *i.e.*, permutations that do not contain monotone subsequences longer than $\sqrt{k}$, are known to exist (for example, there are 4 EES permutations of length 4: 2 1 4 3, 2 4 1 3, 3 1 4 2 and 3 4 1 2). Combining this with algorithms for finding a largest common structured pattern type $\{\sqsubset\}$ or $\{\emptyset\}$, we have obtained the following result.

Proposition 2.5.6 ([Fertin et al., 2007]). *The* MCSP *problem for model* $\mathcal{M} = \{\sqsubset, \emptyset\}$ *is approximable within ratio* $\sqrt{k}$ *in* $O(nm^{1.5})$ *time, where* k *is the size of an optimal solution* $n = |\mathcal{G}|$, *and* $m = \max_{G \in \mathcal{G}} |\mathbf{E}(G)|$.

How tight Proposition 2.5.6 is? Central in Proposition 2.5.6 is the use of family of patterns to probe the input. Can we use more complicated families of patterns to improve the approximation (notice that there is tradeoff to be made here, the family should be large enough, but only polynomially large if we want to possibly consider each member)? Unfortunately, the answer is negative. For $k \in \mathbb{N}$, let $\Pi_k \subseteq \mathfrak{S}_k$ be a set of $|\Pi_k|$ permutations on k elements. Each permutation in Π_k can be equivalently regarded as a linear matching type $\{\sqsubset, \emptyset\}$. Alon recently communicated us a proof of essentially the following lemma.

Lemma 2.5.7 (N. Alon, Private communication). *For every family of permutations* $\Pi_k \subseteq \mathfrak{S}_k$, $k \in \mathbb{N}$ *and* $|\Pi_k| \leq 2^k$, *there exists a permutation* $\pi \in \mathfrak{S}_K$, $K = \Omega(k^2)$, *which avoids all permutations in* Π_k.

Notice that Alon's lemma shows that there exists a linear matching type $\{\sqsubset, \emptyset\}$ of size $K = \Omega(k^2)$ which does not contain any linear matching type $\{\sqsubset, \emptyset\}$ out of a family of at most 2^k such graphs. Hence, even using more involved or interesting families of linear matchings type $\{\sqsubset, \emptyset\}$ to be used to probe our input graphs, no approximation guarantee better than $O(\sqrt{k})$ for maximum common structured patterns type $\{\sqsubset, \emptyset\}$ can be possibly achieved.

2.5.5 Putting everything together

We now consider structured patterns type $\{<, \sqsubset, \emptyset\}$. We gave in [Fertin et al., 2007] three approximation algorithms with increasing time complexities but decreasing approximation ratios. Roughly speaking, these three algorithms rely on sufficiently large sub-patterns that occur in any structured pattern type $\{<, \sqsubset, \emptyset\}$, and the fact that finding maximum common structured patterns of these types is polynomial-time solvable.

Our first approximation algorithm is based on Dilworth's Theorem [Dilworth, 1950] and uses the following simple structure lemma.

Lemma 2.5.8 ([Fertin et al., 2007]). *Let* G *be a linear matching type* $\{<, \sqsubset, \emptyset\}$ *of size* k. *Then* H *contains a simple* (i.e., $\mathcal{M} = \{<\}$, $\mathcal{M} = \{\sqsubset\}$ *or* $\mathcal{M} = \{\emptyset\}$) *structured pattern of size at least* $k^{1/3}$.

It is easily seen, however, that Lemma 2.5.8 is tight. One way to obtain an extremal example of this is as follows: take $k^{1/3}$ balanced towers of staircases, each one of depth $k^{1/3}$ and height $k^{1/3}$, and concatenate them one next to the other into one supergraph of size k, reassigning labels accordingly. Combining the above lemma with the fact that a maximum common simple structured pattern of $\mathcal{G}$ can be found in $O(nm^{1.5})$ time, we have obtained the following approximation algorithm for general structured patterns.

Proposition 2.5.9 ([Fertin et al., 2007]). *The* MCSP *problem for patterns type* $\{<, \sqsubset, \emptyset\}$ *is approximable within ratio* $O(k^{2/3})$ *in* $O(nm^{1.5})$ *time, where* k *is the size of an optimal solution,* $n = |\mathcal{G}|$, *and* $m = \max_{G \in \mathcal{G}} |E(G)|$.

Our second approximation algorithm is based on [Kostochka, 1988] (see also [Kostochka and Kratocvil, 1997]). It uses the following structure lemma.

Lemma 2.5.10 ([Fertin et al., 2007]). *Let* G *be a linear matching type* $\{<, \sqsubset, \emptyset\}$ *of size* k. *Then* G *contains a subgraph of size* $\Omega\left(\sqrt{k/\log(k)}\right)$ *which is either type* $\{<, \sqsubset\}$ *or type* $\{\emptyset\}$.

Combining the above lemma with an algorithm for finding a maximum structured pattern type $\{\between\}$ and the $O(\log(n))$-approximation algorithm for structured patterns type $\{<, \sqsubset\}$, we have obtained the following result.

Proposition 2.5.11 ([Fertin et al., 2007]). *The* MCSP *problem for structured patterns type* $\{<, \sqsubset, \between\}$ *is approximable within ratio* $O(\sqrt{k \log^2(k)})$ *in* $O(nm^2)$ *time, where* k *is the size of an optimal solution,* $n = |\mathcal{G}|$, *and* $m = \max_{G \in \mathcal{G}} |\mathbf{E}(G)|$.

We now turn to our third approximation algorithm. It uses the following structure lemma.

Lemma 2.5.12 ([Fertin et al., 2007]). *Let* G *be a linear matching type* $\{<, \sqsubset, \between\}$ *of size* k. *Then* G *contains either a tower or a balanced sequence of staircases of size* $\Omega\left(\sqrt{k/\log(k)}\right)$.

Combining the above lemma with algorithms for finding a maximum common tower and a balanced sequence of staircases in $\mathcal{G}$ (see [Fertin et al., 2007] for details) we have obtained the following results.

Proposition 2.5.13 ([Fertin et al., 2007]). *The* MCSP *problem for structured patterns type* $\{<, \sqsubset, \between\}$ *is approximable within ratio* $O(\sqrt{k \log(k)})$ *in* $O(nm^{3.5} \log m)$ *time, where* k *is the size of an optimal solution,* $n = |\mathcal{G}|$, *and* $m = \max_{G \in \mathcal{G}} |\mathbf{E}(G)|$.

It remains a challenging problem to improve the approximation ratio for the MCSP problem for structured patterns type $\{<, \sqsubset, \between\}$.

Let us consider here subgraphs of linear matchings type $\mathcal{M}$ for some $\mathcal{M} \subseteq \{<, \sqsubset, \between\}$ with $|\mathcal{M}| = 2$. Prove or disprove the following (we would go for prove): *Any linear matching type* $\{<, \sqsubset, \between\}$ *of size* k *contains either a type* $\{<, \sqsubset\}$ *subgraph of size* $k^{2/3}$, *a type* $\{<, \between\}$ *subgraph of size* $k^{2/3}$, *or a type* $\{\sqsubset, \between\}$ *subgraph of size* $k^{2/3}$. Notice that, unfortunately, true or false, this cannot be applied for approximation purposes (approximating the MCSP problem for patterns type $\{\sqsubset, \between\}$ definitively remains the bottleneck).
Let us put this problem in perspective. For one, we have shown in [Fertin et al., 2007] that any linear matching type $\{<, \sqsubset, \between\}$ of size k contains a subgraph of size $\varepsilon\, k^{2/3}$, where $\varepsilon = (\sqrt{17}-1)/8 \approx 0.39$, which is either type $\{<, \sqsubset\}$, type $\{<, \between\}$, or type $\{\sqsubset, \between\}$. For another, let k be an integer such that $k^{1/3}$ is an integer. A simple construction shows that there exists a linear matching type $\{<, \sqsubset, \between\}$ of size k that contains neither a subgraph type $\{<, \sqsubset\}$, nor a subgraph type $\{<, \between\}$, nor a subgraph type $\{\sqsubset, \between\}$ of size least $\varepsilon\, k^{2/3}$ for any $\varepsilon > 1$. Indeed, assuming the contrary, then any linear matching type $\{<, \sqsubset, \between\}$ of size k contains a subgraph with at least $\sqrt{\varepsilon\, k^{2/3}} = \varepsilon^{1/2}\, k^{1/3}$ edges which is either type $\{<\}$, type $\{\sqsubset\}$, or type $\{\between\}$. A patent contradiction since it can be proved (see [Fertin et al., 2007]) that there exists a linear matching type $\{<, \sqsubset, \between\}$ of size k that does not contain a simple structured pattern of size $\varepsilon\, k^{1/3}$ for any $\varepsilon > 1$.

2.5.6 Towards biologically sounding models

As we have observed in [Herrbach and Vialette, 2005], the MCSP problem does not completely succeed in accurately modeling RNA structures. To this end, we have proposed in [Herrbach and Vialette, 2005] to consider the MCSP problem together with a simple RNA stacking-pair scoring scheme.

A *contact map* is a useful graph-theoretic abstraction (and two-dimensional depiction) of the structure of a protein. For a protein of size n, and a given threshold ε, the contact map $M_\varepsilon = [m_\epsilon(i,j)]$ is an $n \times n$ 0–1 matrix whose entry $m_\varepsilon(i,j)$ equals 1 if the distance between amino acids i and j is less than or equal to ε, and 0 otherwise [Goldman et al., 1999]. Of particular importance in our context, the contact map can also be viewed as a Hamiltonian path (usually depicted horizontally) with nodes representing the amin acids and with edges added that join pairs of nodes whose centers of gravity have been found to be closer to each other than a fixed threshold ε. Contact maps are thus linear graphs. Contact maps have been used for secondary structure prediction, fold assignment, protein structure alignment, and threading (see [Goldman et al., 1999] and references therein). A *Contact map overlap* is a measure of similarity of protein structures based on maximum size common subgraphs in contact maps) [Goldman et al., 1999]. In their pioneered work, Goldman *et al.* [Goldman et al., 1999] have laid the foundations of the algorithmic issues of contact map overlap: the general problem is **NP**-complete and approximable within a constant ratio for some restricted but interesting special cases. Also, they have introduces special structures (**stacks**, **queues** and **staircases**) that are of particular importance for proteins. A notable breakthrough in algorithmic aspects of contact map overlap is a recent paper of Xu *et al.* [Xu et al., 2007] where some preliminary fixed-parameter algorithms are presented. Below are some lines of research we plan to explore.

- Of particular importance, there exists a $O(n^6)$ time algorithm for finding the maximum overlap of two degree 2 contact maps, one of which is either a stack or a staircase [Goldman et al., 1999]. However, due to the high degree of complexity, it is not practical. Improving the time complexity of this problem is a challenging and important problem.
- It is intuitively clear – and widely accepted – that contact maps of real proteins are far from being arbitrary collections of edges since they have a specialized structure reflecting the geometry of proteins. To this end, Goldman *et al.* [Goldman et al., 1999] have introduced a special class of contact maps (**self-avoiding walk on the 2D grid**) that seem to be a long way toward capturing this structure (the problem of computing the maximum overlap remains, however, hard for self-avoiding walks on the 2D grid ... not a real surprise as self-avoiding walks do capture the complexity of real instances). Interesting questions include:
  - ◇ Computing the maximum overlap between two self-avoiding walks on the 2D grid is known to be approximable within ratio 4. Improving this ratio is crucial to bridge the theory–practice gap. A promising line of research could be to improve the **decomposition** of self-avoiding walks on the 2D grid (it is only known that a self-avoiding walk can be decomposed into 2 stacks and 1 queue). Notice that it is not even known whether this problem is **APX**-hard (self-avoiding walks on the 2D grid enjoy some degree of planarity).
  - ◇ Parameterized issues of self-avoiding walks on the 2D grid are completely unexplored. Obtaining efficient fixed-parameter algorithms would be of particular interest for practical perspectives.
  - ◇ It would be of particular interest to discover favorable properties of 3**-dimensional self-avoid walks**. Current approaches seem inherently 2-dimensional in that they exploit topological properties of the plane. These aspects (properties, algorithmic issues, ...) are completely unexplored yet.

If we assume that the secondary structure of an RNA contains no pseudoknots, the secondary structure can be decomposed into a few types of loops: stacking pairs, hairpins, bulges, internal loops and multiple loops [Waterman, 1995]. A *stacking-pair* is a loop formed by two pairs of consecutive bases (i, j) and $(i+1, j-1)$. By definition, a stacking-pair contains no unpaired bases, and any other kinds of loops contain one or more unpaired bases. Since unpaired bases are destabilizing and have positive free energy, staking pairs are the only type of loops that have negative free energy and stabilize the secondary structure, see [Ieong et al., 2003] for a nice application of the stacking-pair scoring scheme to RNA secondary structures. We need some new easy definitions.

Definition 2.5.14 (SP scoring scheme)**.** *Let* G *be a linear matching type* $\{<, \sqsubset\}$*. The* SP-score *of* G*, denoted* $SP(G)$*, is defined by*

$$SP(G) = |\{(i, j) : i + 1 < j - 1 \ \wedge \ (i, j) \in \mathbf{E}(G) \ \wedge \ (i + 1, j - 1) \in \mathbf{E}(G)\}|.$$

Definition 2.5.15 (SP-Trim linear matching)**.** *A linear matching graph* G *type* $\{<, \sqsubset\}$ *is called a* SP-trim nested linear graph *provided that* $SP(G) > SP(G[\mathbf{E}(G) - e])$ *for all* $e \in \mathbf{E}(G)$.

The MCSP for pseudoknot-free RNA under the stacking-pair scoring scheme can be rephrased as follows.

MCSP-SP

- **Input** : A family of linear graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$.
- **Solution** : A common SP-Trim pattern G_{sol} type $\{<, \sqsubset\}$ of $\mathcal{G}$, *i.e.*, a SP-Trim pattern G_{sol} type $\{<, \sqsubset\}$ that occurs in each input linear graph of $\mathcal{G}$.
- **Measure** : The SP-score of G_{sol}, *i.e.*, $SP(G_{sol})$.

We briefly review the results we have obtained for the MCSP-SP problem. Not surprisingly (in the light of Proposition 2.5.1), the MCSP-SP problem is computationally hard even for quite simple instances (notice that Proposition 2.5.1 does not apply here as it is concerned with towers of height 1 or 2).

Proposition 2.5.16 ([Herrbach and Vialette, 2005])**.** *The* MCSP-SP *problem – in its natural decision form – is* **NP***-complete even if* $\mathcal{G}$ *is composed of SP-trim linear matchings type* $\{<, \sqsubset\}$.

The above proposition may be contrasted with the following positive result.

Proposition 2.5.17 ([Herrbach and Vialette, 2005])**.** *The* MCSP-SP *problem is solvable in* $O(4^k k\sqrt{k}\, n\, m^4 \log(m))$ *time, where* $n = |\mathcal{G}|$, $m = \max\{|\mathbf{E}(G_i)| : G_i \in \mathcal{G}\}$, *and* k *is the SP-score of the sought common SP-Trim structured pattern type* $\{<, \sqsubset\}$.

The proof of Proposition 2.5.17 is by enumeration and dynamic programming. The following result, well-suited for fixed $|\mathcal{G}|$, complements Proposition 2.5.17.

Proposition 2.5.18 ([Herrbach and Vialette, 2005])**.** *The* MCSP-SP *problem is solvable in* $O(m^{2n} \log^{n-1}(m^n))$ *time, where* $n = |\mathcal{G}|$, $m = \max\{|\mathbf{E}(G_i)| : G_i \in \mathcal{G}\}$, *and* k *is the SP-score of the sought common SP-Trim structured pattern type* $\{<, \sqsubset\}$.

Interestingly enough, the proof of Proposition 2.5.18 is by a combination of high-dimensional trapezoids diagrams and high-dimensional trapezoids graphs [Felsner et al., 1997]. Crucial in our algorithm is a procedure to compute a maximum weighted disjoint subset of high-dimensional trapezoids (in terms of disjoint induced closed polygons).

According to Proposition 2.5.18, the MCSP-SP problem is polynomial-time solvable for fixed $|\mathcal{G}|$. Furthermore, as we have seen, the MCSP-SP problem is **NP**-complete even if $\mathcal{G}$ is composed of linear matchings type $\{<, \sqsubset\}$. Therefore, there is very little hope that a polynomial-time algorithm exists for this restricted case. However, this raises the important question of whether there exists a more efficient algorithm, although exponential in n, for fixed $|\mathcal{G}|$ in case $\mathcal{G}$ is composed of linear matchings type $\{<, \sqsubset\}$. The answer is positive.

Proposition 2.5.19 ([Herrbach and Vialette, 2005]). *The* MCSP-SP *problem for linear matchings type* $\{<, \sqsubset\}$ *is solvable in* $O(nm^{2n})$ *time, where* $n = |\mathcal{G}|$ *and* $m = \max\{|\mathbf{E}(G_i) : G_i \in \mathcal{G}\}$.

As the reader might have guessed, there is strong relationships between tree alignment problems and the MCSP-SP problem in case $\mathcal{G}$ is composed of linear matchings type $\{<, \sqsubset\}$, and this similarity is indeed at the heart of Proposition 2.5.19. First, observe that a linear matching type $\{<, \sqsubset\}$ can easily be mapped to an ordered tree structure. Second, grouping together stacking edges, we can assume that the ordered trees are vertex weighted by the number of stacking edges, *i.e.*, thickness. The goal is thus clearly to find a common homeomorphic ordered subtree with additional constraints on the weights (see [Herrbach and Vialette, 2005] for details).

2.6 Separable patterns

The preceding sections were concerned with two problems: (i) searching for an occurrence of a permutation in another one and (ii) finding a maximum size common pattern in a collection of linear graphs. But, as we have observed, a permutation is nothing but a special linear graph, and hence there is a natural interest in combining the two above-mentioned problems, *i.e.*, finding a maximum size common permutation in a collection of permutation. But the general problem of finding an occurrence of a permutation in another one is **NP**-complete [Bose et al., 1998], and hence, for algorithmic purposes, we need to restrict ourselves to classes of permutations for which the PERMUTATION PROBLEM problem is polynomial-time solvable. We focus in this section on *separable permutations* (this is actually not really a choice as, as far as we are aware of, separable permutations constitute the only non-trivial class of permutations for which the PERMUTATION PATTERN problem is polynomial-time solvable).

Recall that a permutation is separable if it contains neither the subpattern 3142 nor 2413 [Bose et al., 1998; Ibarra, 1997] and that the PERMUTATION PATTERN problem is polynomial-time solvable for separable patterns. There are actually numerous characterizations of separable permutations, for example in terms of permutation graphs [Bose et al., 1998], of interval decomposition [Bui-Xuan et al., 2005; Bergeron et al., 2005; Rossin and Bouvel, 2006], or with ad-hoc structures [Bose et al., 1998]. Our results can be stated as follows.

We are not aware of any *finitely based* pattern-avoiding permutation classes $\mathcal{C}$ for which the recognition problem (*i.e.*, $\pi \in \mathcal{C}$?) is **NP**-hard. This remarks motivates the following problem: Is the problem of finding a maximum size $\mathcal{C}$-pattern in a collection of n permutations polynomial-time solvable for any finitely based $\mathcal{C}$? If not, exhibit such a class $\mathcal{C}$ for which this problem is **NP**-hard. As far as we know, this problem is completely open.

Proposition 2.6.1 ([Bouvel et al., 2007]). *The problem of finding a common maximum size separable permutation in a collection of* m *permutations, each of size at most* n*, is solvable in* $O(n^{6m+1})$ *time.*

Being exponential in the number of permutation is roughly the best one can obtain as shown in the following proposition (we cannot exclude, however, a $O(n^{O(\sqrt{m})}$ time algorithm).

Proposition 2.6.2 ([Bouvel et al., 2007]). *The problem of finding a common maximum size separable permutation in a collection of permutations is* **NP***-complete, even if each input permutation is separable.*

We end this section with a first attempt to address the following question: How much finding a common maximum size separable permutation does help when looking for a common maximum size (general) pattern in a collection of permutations. Put it in our context, is a common maximum size separable permutation a good approximation of a common maximum size permutation? We have obtained the following – somewhat negative – general result.

Proposition 2.6.3 ([Bouvel et al., 2007])**.** *Let* Π *be a set of permutations and* $\mathcal{C}$ *be any pattern avoiding permutation class. Furthermore, let* k *(resp.* $k_{\mathcal{C}}$*) be the maximum size of a common pattern (resp. maximum size of a common* $\mathcal{C}$ *pattern) in* Π*. Then,* $k/k_{\mathcal{C}} \leq \sqrt{k}$*, and the inequality is tight.*

In other words, the problem of finding a common maximum size pattern in a collection of permutations cannot be approximated within a performance ratio better than $\sqrt{\mathbf{opt}}$ by the problem of finding a common maximum size $\mathcal{C}$-pattern, where **opt** is the size of an optimal solution and $\mathcal{C}$ is any pattern-avoiding permutation class.

3 Arc-annotated sequences

Contents

3.1 Introduction

Structure comparison for RNA has become a central computational problem bearing many computer science challenging questions. Indeed, RNA secondary structure comparison is essential for (i) identification of highly conserved structures during evolution (which cannot always be detected in the primary sequence, since it is often unpreserved) which suggest a significant common function for the studied RNA molecules, (ii) RNA classification of various species (phylogeny), (iii) RNA folding prediction by considering a set of already known secondary structures and (iv) identification of a consensus structure and consequently of a common role for molecules.

From an algorithmic point of view, RNA structure comparison was first considered in the framework of ordered trees [Shasha and Zhang, 1989]. More recently, it has also been considered in the framework of *arc-annotated sequences* [Evans, 1999c]. An arc-annotated sequence is a pair (u, P) where u is a sequence of RNA bases and P represents hydrogen bonds between pairs of elements of u. From a purely combinatorial point of view, arc-annotated sequences are a natural extension of simple sequences. However, using arcs for modeling non-sequential information together with restrictions on the relative positioning of arcs allow for varying restrictions on the structure of arc-annotated sequences.

Different pattern matching and motif search problems have been considered in the context of arc-annotated sequences among which we can mention the Longest Arc-Annotated Subsequence (LAPCS) problem, the Arc Preserving Subsequence (APS) problem, the Maximum Arc-Preserving Common Subsequence (MAPCS) problem, and the Edit-distance for arc-annotated sequence (EDIT) problem [Evans, 1999a; Jiang et al., 2000b; Alber et al., 2004; Gramm et al., 2006]. For an up-to-date survey of this area we refer the reader to our chapter [Blin et al., 2010a].

This chapter is devoted to presenting our algorithmic results for arc-annotated based problems. It is organized as follows. Section 3.2 presents some preliminaries. Section 3.3 deals with the problem of finding a common arc-annotated sequence (the LAPCS problem) whereas Section 3.4 is concerned with pattern matching issues in arc-annotated sequences. In Section 3.5, we extend the standard model with RNA applications in mind.

3.2 Definitions

Definition 3.2.1 (Arc-annotated sequence)**.** *An* arc-annotated sequence *over alphabet $\mathcal{A}$ is a pair (u, P), where u (the* sequence*) is a string over $\mathcal{A}^*$ and P (the* annotation*) is a set of arcs $\{(i, j) : 1 \leq i < j \leq |u|\}$.*

Notice that even if these objects are described in terms of arcs, the orientation is not relevant and we are actually concerned with edges (but we follow the standard terminology here). In the context of RNA structures, we have $\mathcal{A} = \{A, C, G, U\}$, and u and P represent the nucleotide sequence and the hydrogen bonds of the RNA structure, respectively. Characters in u are thus often referred to as *bases*. A letter of u is said to be *free* if it is no incident to an arc of P (this point is crucial if one compare arc-annotated sequences with linear graphs as the latters do not allow for free vertices). Two arcs of P are *independent* if they do not share a vertex.

Definition 3.2.2 (Occurrence)**.** *Let (u, P) and (v, Q) be two arc-annotated sequences. The arc-annotated sequence (v, Q)* occurs *in (u, P) if (v, Q) can be obtained from (u, P) by letter deletions.*

Notice that the above definition does not allow for edge deletion, *i.e.*, a a does not occurs in a b a. The definition 3.2.2 is illustrated Figure 3.1.

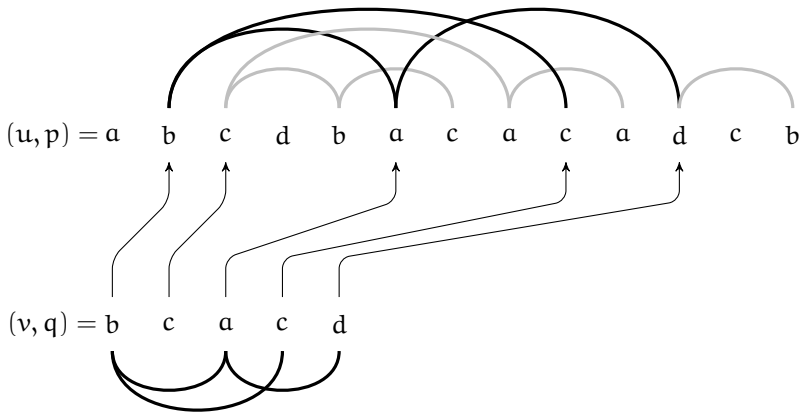


Figure 3.1: Occurrence of an arc-annotated sequence in another arc-annotated sequence.

Again, the relative positioning of arcs is of particular importance for arc-annotated sequences. Following the example of 2-intervals and linear graphs, this relative positioning is completely described by three binary relations: (i) the *precedence*, denoted $<$, (ii) the *inclusion*, denoted $\sqsubset$, and (iii) the *crossing*, denoted $\between$. For the sake of consistency, we choose to adopt this general framework for presenting our results.

Definition 3.2.3. *Let (u, P) be an arc-annotated sequence, and (i, j) and (k, l) be two independent arcs of P. The binary relations $<$, $\sqsubset$ and $\between$ are defined as follows:*

- ***precedence***: $(i,j) < (k,l)$ *if and only if* $j < k$,
- ***inclusion***: $(k,l) \sqsubset (i,j)$ *if and only if* $i < k$ *and* $l < j$, *and*
- ***crossing***: $(i,j) \between (k,l)$ *if and only if* $i < k < j < l$.

The notations $(k,l) \sqsubset (i,j)$ and $(i,j) \sqsupset (k,l)$, and $(i,j) < (k,l)$ and $(k,l) > (i,j)$ are of course equivalent (note, however, that there does not exist an equivalent for the relation $\between$). The binary relations $<$, $\sqsubset$ et $\between$ are illustrated Figure 3.2.

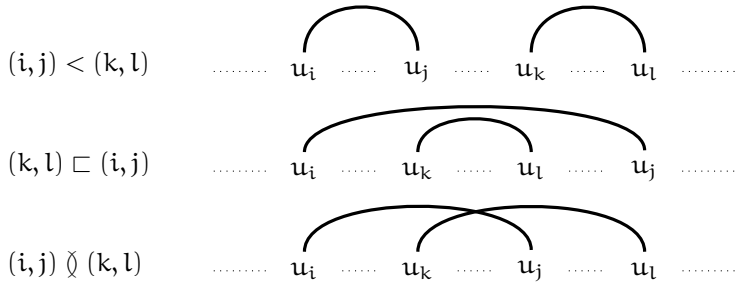


Figure 3.2: The binary relations $<$, $\between$, $\between$.

For the sake of presentation, for two arcs $(i,j), (k,l) \in p$, we write

- $(i,j) \sim_< (k,l)$ if and only if $(i,j) < (k,l)$ or $(i,j) > (k,l)$,
- $(i,j) \sim_\sqsubset (k,l)$ if and only if $(i,j) \sqsubset (k,l)$ or $(i,j) \sqsupset (k,l)$, and
- $(i,j) \sim_\between (k,l)$ if and only if $(i,j) \between (k,l)$ or $(k,l) \between (i,j)$.

In her pioneering work [Evans, 1999a], Evans has introduced a five level hierarchy for arc-annotated sequences that is described as follows

- **UNLIMITED**: No restriction.
- **CROSSING**: for any two arcs $(i,j), (k,l) \in p$, either $(i,j) \sim_< (k,l)$, $(i,j) \sim_\sqsubset (k,l)$, or $(i,j) \sim_\between (k,l)$.
- **NESTED**: for any two arcs $(i,j), (k,l) \in p$, either $(i,j) \sim_< (k,l)$ or $(i,j) \sim_\sqsubset (k,l)$.
- **CHAIN**: for any two arcs $(i,j), (k,l) \in p$, $(i,j) \sim_< (k,l)$.
- **PLAIN**: No arc, *i.e.*, $p = \emptyset$.

The **PLAIN** level thus corresponds to sequences in the usual sense. This hierarchy is clearly organized according to the following chain of inclusions:

$$\textbf{PLAIN} \subset \textbf{CHAIN} \subset \textbf{NESTED} \subset \textbf{CROSSING} \subset \textbf{UNLIMITED}.$$

The hierarchy introduced by Evans is clearly incomplete (with respect to the combinatorics induced by the three binary relations $<$, $\sqsubset$ and $\between$). In particular, in the context of RNA secondary structures, the above hierarchy does not allow to precisely describe stems. To this end, a first refinement of the **NESTED** level has been proposed [Guignon et al., 2005]: for any two arcs $(i,j), (k,l) \in p$, $(i,j) \sim_\sqsubset (k,l)$.

Extending our works on 2-intervals and linear graphs, we have proposed in [Blin et al., 2005a] a clear unified framework for arc-annotated sequences. While we claim no novelty at all, we do believe this general

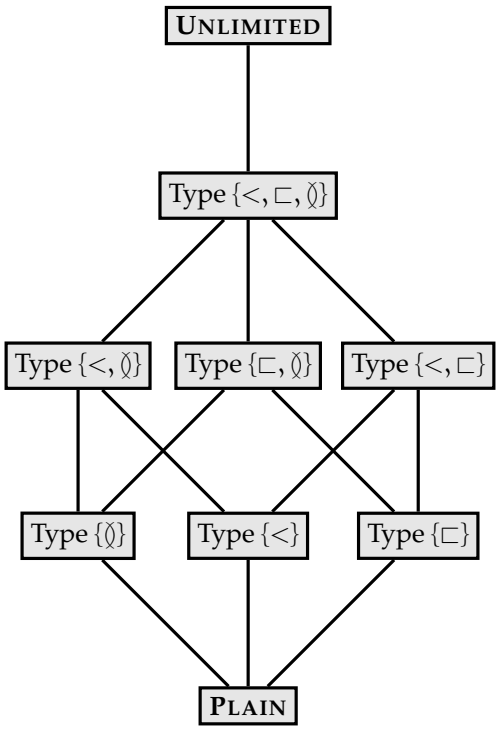


Figure 3.3: The refined arc-annotated sequences hierarchy.

framework allows us for varying restrictions in a clear and precise way. We give an outline of this approach. Let $\mathcal{M} \subseteq \{<, \sqsubset, \Diamond\}$, $\mathcal{M} \neq \emptyset$. An arc-annotated sequence (u, P) is type $\mathcal{M}$ if for any two arcs $(i, j), (k, l) \in p$, there exists a binary relation $R \in \mathcal{M}$ such that $(i, j) \sim_R (k, l)$. According to this definition, **CROSSING** corresponds to type $\{<, \sqsubset, \Diamond\}$, **NESTED** correspond to type $\{<, \sqsubset\}$, and **PLAIN** is type $\{<\}$. If we define the **PLAIN** level as the class of all arc-annotated sequences with at most one arc, we are left with the refined hierarchy given Figure 3.3. It is this hierarchy we have chosen to adopt in the sequel.

3.3 Maximum common patterns

Evans has introduced in [Evans, 1999b] the natural extension of longest common subsequences [Bergroth et al., 2000; Crochemore et al., 2007] to arc-annotated sequences. This problem is known as the LAPCS (LONGEST ARC-PRESERVING COMMON SUBSEQUENCE) problem.

LAPCS

- **Input** : Two arc-annotated sequences (u, p) and (v, Q).
- **Solution** : An arc-annotated sequence (w, R) that occurs in both (u, P) and (v, Q).
- **Measure** : The number of letters of (w, R), *i.e.*, $|w|$.

For two subsets $\mathcal{M}, \mathcal{M}' \in \{<, \sqsubset, \Diamond\}$, $\mathcal{M} \neq \emptyset$, $\mathcal{M}' \neq \emptyset$, we let LAPCS$(\mathcal{M}, \mathcal{M}')$ stand for the LAPCS problem where (u, P) (resp. (v, Q)) is an arc-annotated sequence type $\mathcal{M}$ (resp. $\mathcal{M}'$).

The complexity (standard, approximation and parameterized) has been studied in numerous papers and manuscripts [Evans, 1999b,c; Jiang et al., 2000b; Lin et al., 2002; Guo, 2002; Hamel et al., 2010]. Evans

[Evans, 1999b] was the first to prove that the LAPCS($\{<, \sqsubset, \between\}$, PLAIN) problem is **NP**-complete. On the positive side, she proved that the LAPCS($\{<\}, \{<\}$) problem is polynomial-time solvable. Jiang *et al.* [Jiang et al., 2000b] have proposed a $O(nm^3)$ time algorithm for both the LAPCS($\{<, \sqsubset\}, \{<\}$) and LAPCS($\{<\}, \{<\}$) problems. Of particular importance for pseudoknot-free RNA secondary structures, Lin *et al.* [Lin et al., 2002] have proved (such a clever reduction!) that the LAPCS($\{<, \sqsubset\}, \{<, \sqsubset\}$) problem is **NP**-complete even if (u, P) and (v, Q) are built over a one letter alphabet. This latter result leaves the possibility of a polynomial-time algorithm for the LAPCS($\{\sqsubset\}, \{\sqsubset\}$) problem computational biologists are particularly interested in (see also [Guignon et al., 2005]). Unfortunately (and surprisingly I would say ... I used to think that the problem was polynomial-time solvable), our recent result rules out such a positive issue.

Proposition 3.3.1 ([Hamel et al., 2010]). *The* LAPCS($\{\sqsubset\}, \{\sqsubset\}$) *problem is* **NP***-complete.*

The complexity of the LAPCS($\{\sqsubset\}, \{\sqsubset\}$) problem remains, however, open if the two arc-annotated sequences are built over a fixed-size alphabet (quite a natural restriction for practical applications). Nevertheless, we conjecture that the LAPCS($\{\sqsubset\}, \{\sqsubset\}$) problem remains **NP**-complete for a fixed-size alphabet (we also conjecture that it would not be a piece of cake to prove hardness of this restriction).

It is shown in [Alber et al., 2002] that a simple enumerative brute-force algorithm solves the LAPCS($\{<, \sqsubset\}, \{<, \sqsubset\}$) in $O((3|\mathcal{A}|)^l\, n\, l)$ time, where l is the length of the common subsequence and $|\mathcal{A}|$ is the size of the underlying alphabet. Central in this approach is a dynamic programming algorithm [Gramm et al., 2006] that determines, given two arc-annotated sequences (u, P) and (v, q)), in $O(|u|\,|v|)$ time whether (v, Q) is an arc-preserving subsequence of (u, v).
Clearly, the above presented result only leads to an efficient exact algorithm if parameter l (subsequence length) is small. To get round this limitation, a more involved – and probably more practical – algorithm in presented in [Alber et al., 2002] to determine in $O(3.31^{k_1+k_2}\, n)$ time whether an arc-preserving common subsequence can be obtained by deleting (together with incident arcs) k_1 letters from (u, v) and k_2 from (v, Q). It is a challenging problem to adapt this search tree based algorithm or to develop a new approach for the **NP**-complete LAPCS($\{\sqsubset\}, \{\sqsubset\}$) problem. This problem deserves deep consideration.

3.4 Pattern matching

The APS (ARC-PRESERVING SUBSEQUENCE) problem is the natural extension of the usual pattern matching [Crochemore et al., 2007] to arc-annotated sequences.

APS

- **Input** : Two arc-annotated sequences (u, p) and (v, Q).
- **Question** : Does there exist an occurrence of (u, P) in (v, Q)?

Notice that, oppositely to the LAPCS problem, the APS problem is a pure decision problem. Again, for two sets $\mathcal{M}, \mathcal{M}' \in \{<, \sqsubset, \between\}$, $\mathcal{M} \neq \emptyset$, $\mathcal{M}' \neq \emptyset$, we let APS($\mathcal{M}, \mathcal{M}'$) stand for the APS problem where (u, P) (resp. (v, Q)) is an arc-annotated sequence type $\mathcal{M}$ (resp. $\mathcal{M}'$).

Guo [Guo, 2002] has shown that the APS$(\{<, \sqsubset, \between\}, \{<\})$ problem is **NP**-complete. He has also observed in [Gramm et al., 2002, 2006] that the hardness of both the APS$(\{<, \sqsubset, \between\}, \{<, \sqsubset, \between\})$ and APS(**UNLIMITED**, **PLAIN**) problems are direct consequences of Evans' works on the LAPCS problem [Evans, 1999b]. Algorithms with $O(nm)$ and $O(n + m)$ running times, $n = |u|$ et $m = |v|$, are described in [Gramm et al., 2002, 2006] for the APS$(\{<, \sqsubset\}, \{<, \sqsubset\})$ and APS$(\{<\},$ **PLAIN**$)$ problem, respectively.

Trying to precisely confine the intractability of the APS problem, quite arduous polynomial reductions have allowed us to refine and complete the results of Guo [Guo, 2002].

Proposition 3.4.1 ([Blin et al., 2005a]). *The* APS$(\{\sqsubset, \between\},$ **PLAIN**$)$ *and the* APS$(\{<, \between\},$ **PLAIN**$)$ *problems are* **NP***-complete.*

In other words, using the binary relation $\between$ in conjunction with $<$ or $\sqsubset$ is enough to get intractability. This negative result is confirmed by the following easy proposition that shows that the binary relation $\between$ alone does not result in intractability.

Proposition 3.4.2 ([Blin et al., 2005a]). *The* APS$(\{\between\}, \{\between\})$ *problem is solvable in* $O(nm^2)$ *time, where* $n = |u|$ *and* $m = |v|$.

3.5 Extending the standard model

Whereas clearly defined, one may naturally argues that the objective function of the LAPCS problem considers letters only. If this definition makes sense, undoubtedly, for the standard pattern matching framework, one may reasonably doubt about the adequacy and the accuracy of this model for RNA where the weight of the arcs cannot be neglected. On that account, we have proposed in [Blin et al., 2007a] a simple extension of the LAPCS problem, referred hereafter as the MAPCS (MAXIMUM ARC-PRESERVING COMMON SUBSEQUENCE) problem, where the objective function is concerned with both the number of letters and the number of arcs in a solution.

MAPCS

- **Input** : Two arc-annotated sequences (u, p) and (v, Q) built over alphabet $\mathcal{A}$, and functions $f : \mathcal{A} \to \mathbb{N}^*$ and $g : \mathcal{A}^2 \to \mathbb{N}^*$.
- **Solution** : An arc-annotated sequence (w, R) that occurs both in (u, P) and in (v, Q).
- **Measure** : The score $s((w, r)) = \sum_{a \in w} f(a) + \sum_{(i,j) \in r} g(w[i], w[j])$ of the arc-annotated sequence (w, R).

Once again, for any two sets $\mathcal{M}, \mathcal{M}' \in \{<, \sqsubset, \between\}$, $\mathcal{M} \neq \emptyset$, $\mathcal{M}' \neq \emptyset$, we simply let MAPS$(\mathcal{M}, \mathcal{M}')$ stand for the MAPS problem where (u, P) (resp. (v, Q)) is an arc-annotated sequence type $\mathcal{M}$ (resp. $\mathcal{M}'$).

Observe that the LAPCS problem is nothing but the MAPCS problem for zero everywhere g, *i.e.*, $g((x, y)) = 0$ for all $(x, y) \in \mathcal{A}^2$, and hence all negative results of the LAPCS problem directly propagate to the MAPCS problem. Focusing on zero everywhere f, *i.e.*, $f(x) = 0$ for all $x \in \mathcal{A}$, results in a more interesting problem that deserves separate consideration. Our positive and negative contributions are given in the following propositions (the first two are concerned with zero everywhere f).

Not surprisingly, the MAPCS problem is hard for simple instance.

Proposition 3.5.1 ([Blin et al., 2007a]). *The* MAPCS$(\{<, \sqsubset\}, \{<, \sqsubset\})$ *and* MAPCS$(\{<, \sqsubset, \between\},$ **PLAIN**$)$ *problems are* **NP***-complete.*

Proposition 3.5.2 ([Blin et al., 2007a]). *The* MAPCS$(\{<, \sqsubset, \between\}, \{<, \sqsubset, \between\})$ *problem is* **NP***-complete even for zero everywhere* f.

The two following propositions give some positive results.

Proposition 3.5.3 ([Blin et al., 2007a]). *For zero everywhere* f, *the* $\mathsf{MAPCS}(\{<, \sqsubset\}, \{<, \sqsubset\})$, $\mathsf{MAPCS}(\{<, \sqsubset\}, \{<\})$, *and* $\mathsf{MAPCS}(\{<\}, \{<\})$ *problems are solvable in* $O(n^2m^2)$, $O(m^2, n)$ *and* $O(nm)$ *time, respectively, where* $n = |u|$ *and* $m = |v|$.

Proposition 3.5.4 ([Blin et al., 2007a]). *The* $\mathsf{MAPCS}(\{<, \sqsubset\}, \{<\})$ *and* MAPCS $(\{<\}, \{<\})$ *problems are solvable in* $O(nm^3)$ *and* $O(nm)$ *time, respectively, where* $n = |u|$ *and* $m = |v|$.

Part II

Pattern Matching in Graphs

Introduction

High-throughput analysis makes possible the study of protein-protein interactions at a genome-wise scale [Gavin et al., 2002; Ho et al., 2002; Uetz et al., 2000], and comparative analysis tries to determine the extent to which protein networks are conserved among species. Indeed, mounting evidence suggests that proteins that function together in a pathway or a structural complex are likely to evolve in a correlated fashion, and, during evolution, all such functionally linked proteins tend to be either preserved or eliminated in a new species [Pellegrini et al., 1999].

Protein interactions identified on a genome-wide scale are commonly visualized as protein interaction graphs, where proteins are vertices and interactions are edges [Titz et al., 2004]. Experimentally derived interaction networks can be extremely complex, so that it is a challenging problem to extract biological functions or pathways from them. However, biological systems are hierarchically organized into functional modules. Several methods have been proposed for identifying functional modules in protein-protein interaction graphs. As observed in [Pereira-Leal et al., 2004], cluster analysis is an obvious choice of methodology for the extraction of functional modules from protein interaction networks. Comparative analysis of protein-protein interaction graphs aims at finding complexes that are common to different species. Kelley *et al.* [Kelley et al., 2003] developed the program PathBlast, which aligns two protein-protein interaction graphs combining topology and sequence similarity. Sharan *et al.* [Sharan et al., 2004] studied the conservation of complexes (they focused on dense, clique-like interaction patterns) that are conserved in *Saccharomyces cerevisae* (a species of budding yeast) and *Helicobacter pylori* (a gram-negative, microaerophilic bacterium that infects various areas of the stomach and duodenum), and found 11 significantly conserved complexes (several of these complexes match very well with prior experimental knowledge on complexes in yeast only). They actually recasted the problem of searching for conserved complexes as a problem of searching for heavy subgraphs in an edge- and node-weighted graph, whose vertices are orthologous protein pairs. A promising computational framework for alignment and comparison of more than one protein network together with a three-way alignment of the protein-protein interaction networks of *Caenorhabditis elegans*, *Drosophila melanogaster* and *Saccharomyces cerevisae* is presented in [Sharan et al., 2005].

This part is devoted to graph-based algorithmic aspects of this topic. We have divided our presentation into three chapters. Chapter 4 is devoted to graph homomorphisms-like aspects. The rationale for this research is that graph-homomorphisms do preserve adjacencies and hence are a natural choice for pattern matching problems in biological networks (as long as injectivity is also required!). Chapter 5 is concerned with a more recent view of graph motifs in biological networks. Here, topology is of lesser importance but the functionalities of network nodes (expressed by colors) form the governing principle. Finally, we present in Chapter 6 our contribution to a somewhat more classical view of pattern matching in biological networks.

4 Pattern matching in graphs

Contents

4.1 Introduction

We consider in this chapter two edge-preserving pattern matching problems in graphs (one being a restriction of the other). Common to these two problems are the fact that each vertex of the motif (given in the form of a graph) is allowed to match to only few vertices of the target graph. First, we shall consider the case where each vertex of the motif is associated with the list of vertices of the target graph it is allowed to match. Notice that we shall only discuss about "*lists*"whereas we actually mean "*sets*"as order is not relevant here . . . but most – not to say all – references in this area use the term list. Our interest in this problem will be for fixed cardinality lists. Second, we shall consider a natural restriction on lists: any two intersecting lists are equal. It will be more convenient to use colors instead of lists for this particular problem.

This chapter is organized as follows. Section 4.2 presents some preliminaries. Section 4.3 is devoted to list graph matching and we consider in Section 4.4 approximate occurrences. Section 4.5 is concerned with the relaxation to colors.

4.2 Definitions

A *graph homomorphism* θ from a graph G to a graph H, written $\theta : G \to H$, is a mapping $\theta : \mathbf{V}(G) \to \mathbf{V}(H)$ from the vertex set of G to the vertex set of H such that $\{u, v\} \in \mathbf{E}(G)$ implies $\{\theta(u), \theta(v)\} \in \mathbf{E}(H)$. A graph homomorphism is thus a mapping between two graphs that respects their structure; more concretely it maps adjacent vertices to adjacent vertices. If $\theta : G \to H$, G is said to be homomorphic to H or H-colorable. Indeed,

in terms of graph coloring, k-colorings of G are precisely homomorphisms $\theta : G \to K_k$, where K_k is the complete graph with k vertices. As a consequence if $G \to H$, the chromatic number of G is at most that of H. The best general reference is [Hell and Nešetřil, 2004]. If the homomorphism $\theta : G \to H$ is a bijection whose inverse function is also a graph homomorphism, then θ is a *graph isomorphism*. Determining whether there is an isomorphism between two graphs is an important (but hard!) problem in computational complexity theory (see [Garey and Johnson, 1979]).

Given graphs G and H, and lists $\mathcal{L}(u) \subseteq \mathbf{V}(H)$, $u \in \mathbf{V}(G)$, a *list homomorphism* of G to H with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, is an homomorphism $\theta : G \to H$ such that $\theta(u) \in \mathcal{L}(u)$ for all $u \in \mathbf{V}(G)$. By abuse of notation, for any $v \in \mathbf{V}(H)$, we let $\mathcal{L}^{-1}(v)$ stand for $\{u \in \mathbf{V}(G) : v \in \mathcal{L}(u)\}$. Recall that the *degree* of a vertex $\delta(u)$ is the number of vertices adjacent to u and that the degree of G is $\Delta(G) = \max\{\delta(u) : u \in \mathbf{V}(G)\}$. A graph is *regular* of degree Δ or Δ-regular if the degree of all vertices equal Δ.

4.3 Finding exact occurrences

4.3.1 Polynomial cases, hardness and coping with hardness

The problem we are interested in is formally defined as follows.

Exact-(μ_G, μ_H)-Matching

- **Input** : Two graphs G and H, and the lists $\mathcal{L}(u) \subseteq \mathbf{V}(H)$, $u \in \mathbf{V}(G)$, such that (i) $\max\{|\mathcal{L}(u)| : u \in \mathbf{V}(G)\} \leq \mu_G$ and (ii) $\max\{|\mathcal{L}^{-1}(v)| : v \in \mathbf{V}(H)\} \leq \mu_H$.
- **Question** : Does there exist an injective list homomorphism $\theta : G \to H$ with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$?

Clearly, we may assume $|\mathcal{L}(u)| > 0$ for all $u \in \mathbf{V}(G)$, and $|\mathcal{L}^{-1}(v)| > 0$ for all $\in \mathbf{V}(H)$ (a trivial clean-up procedure would apply otherwise). For now on, unless explicitly stated, we assume μ_G and μ_H to be fixed constant. Indeed, observe that for unbounded μ_G and μ_H, the EXACT-(μ_H, μ_G)-MATCHING problem trivially contains the CLIQUE problem, and hence is **NP**-complete [Garey and Johnson, 1979]. Furthermore, to avoid heavy notations, we will let n and m stand for the number of vertices and the number of edges of G, respectively, and p and q stand for the number of vertices and the number of edges of H, respectively.

As sketched above, most on our interest in the EXACT-(μ_H, μ_G)-MATCHING problem is concerned with small (and actually fixed) μ_G and μ_H. We have proved in [Fagnot et al., 2008] that the problem we are interested in is polynomial-time for $\mu_G \leq 2$.

Proposition 4.3.1 ([Fagnot et al., 2008]). *The* EXACT-$(2, \mu_H)$-MATCHING *is solvable in* $O(n^3 + q)$ *time. This reduces to* $O(n + q)$ *time if* $\mu_H = O(1)$.

Notice that the counting problem associated to EXACT-$(2, \mu_H)$-MATCHING is #**P**-complete and is solvable in $O(1.3247^n)$ time [Fagnot et al., 2008]. We have completed the above proposition in [Fertin et al., 2009b].

Proposition 4.3.2 ([Fertin et al., 2009b]). *The* EXACT-$(\mu_G, 1)$-MATCHING *is solvable in linear time for* $\Delta(G) \leq 2$, *for any constant* μ_G.

One may argue, however, that the above proposition is too constrained to be of interest (each vertex $u \in \mathbf{V}(G)$ has private vertices in H and G is collection of paths and cycles). Unfortunately, despite the simplicity of Proposition 4.3.2, the result is quite tight - taking into consideration both $\Delta(G)$ and $\Delta(H)$ - as shown in the following proposition that summarize our negative results.

Proposition 4.3.3 ([Fagnot et al., 2008; Fertin et al., 2009b]). *The following problems are* **NP***-completes:*

- *the* EXACT-$(3,2)$-MATCHING *problem for* $\Delta(G) \leq 1$ *and* $\Delta(H) \leq 2$,
- *the* EXACT-$(3,1)$-MATCHING *problem for bipartite* G *and* H, *and*
- *the* EXACT-$(3,1)$-MATCHING *for* $\Delta(G) \leq 3$ *and* $\Delta(H) \leq 4$.

In the light of the negative results presented in Proposition 4.3.3, a substantial part of our work presented [Fagnot et al., 2008] was devoted to coping with hardness by means of exponential-time algorithms.

Proposition 4.3.4 ([Fagnot et al., 2008]). *The* EXACT-(μ_G, μ_H)-MATCHING *problem is solvable*

- *in* $O(1.1889^n)$ *time and exponential space,*
- *in* $O(1.2025^n)$ *time and polynomial-space,*
- *in* $O(1.2388^{n+m})$ *time, and*
- *in* $(2 - 2/(\mu_G + 1))^n$ *time within a polynomial factor.*

Parameterized complexity issues of the EXACT-(μ_G, μ_H)-MATCHING problem have been initiated in [Fagnot et al., 2004] and further investigated in [Fagnot et al., 2008]. We have considered two natural parameters: (i) the number of ambiguous vertices in G (those vertices that can match different vertices in H), and (ii) an objective with respect to a weight function. It turns out that the first parameterization yields to fixed-parameter tractability whereas the second yields to parameterized intractability.

Proposition 4.3.5 ([Fagnot et al., 2008]). *The* EXACT-(μ_G, μ_H)-MATCHING *problem is solvable in* $O(k\,(\mu_G)^k\,(n+m))$ *time, where* $k = |\{u \in \mathbf{V}(G) : |\mathcal{L}(u)| > 1\}|$.

Proposition 4.3.6 ([Fagnot et al., 2008]). *Let* G *and* H *be two graphs,* $\mathcal{L}(u) \subseteq \mathbf{V}(H)$, $u \in \mathbf{V}(G)$ *be lists, and* $\omega : (\mathbf{V}(G) \times \mathbf{V}(H)) \to \mathbb{N}^+$ *be a scoring such that* $\omega(u,v) > 0$ *only if* $v \in \mathcal{L}(u)$. *Deciding whether there exists an injective homomorphism* $\theta : G \to H$ *with respect to the lists* $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, *such that* $\sum_{u \in \mathbf{V}(G)} \omega(u, \theta(u)) \geq k$ *is a* **W[1]***-hard problem with respect to parameter* k.

In other words, under a reasonable and commonly accepted complexity hypothesis Downey and Fellows [1999], there does not exist an algorithm exponential in k only to determine whether there exists an injective homomorphism $\theta : G \to H$ with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, such that $\sum_{u \in \mathbf{V}(G)} \omega(u, \theta(u)) \geq k$ It is worth noticing that Proposition 4.3.6 holds even if $\omega(u,v) \in \{0,1\}$ for all $(u,v) \in (\mathbf{V}(G) \times \mathbf{V}(H))$.

4.3.2 The corresponding number

Aiming at separating Yes instances from possibly No instances (and hence speeding-up our algorithms for some special instances), we have introduced in [Fertin et al., 2009b] the *correspondence number* $C(G,H,\mathcal{L})$ of any instance of the EXACT-$(\mu_G, 1)$-MATCHING problem. It is defined as follow:

$$C(G,H,\mathcal{L}) = \min\left\{ \frac{|\{u',v'\} : u' \in \mathcal{L}(u) \wedge v' \in \mathcal{L}(v) \wedge \{u',v'\} \in \mathbf{E}(H)\}|}{|\mathcal{L}(u)|\,|\mathcal{L}(v)|} : \{u,v\} \in \mathbf{E}(G) \right\}.$$

For now on, we assume that, for each edge $\{u,v\} \in \mathbf{E}(G)$, there exists an edge $\{u',v'\} \in \mathbf{E}(H)$ such that $u' \in \mathcal{L}(u)$ and $v' \in \mathcal{L}(v)$ (see [Fertin et al., 2009b] for details). The rationale for introducing the corresponding number $C(G,H,\mathcal{L})$ stems from the following observations. For one $(\mu_G)^{-2} \leq C(G,H,\mathcal{L}) \leq 1$. For another, if $C(G,H,\mathcal{L}) = 1$, then there exists an injective homomorphism $\theta : G \to H$ with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$. Indeed, any injective mapping of G to H with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, is a solution

(recall that $\mu_H = 1$). Ideally, one would like to determine a bound c^* as small as possible, ${\mu_G}^{-2} < c^* < 1$, such that if $C(G, H, \mathcal{L}) > c^*$ then $(G, H, \mathcal{L})$ is a Yes instance and if $C(G, H, \mathcal{L}) \leq c^*$ then $(G, H, \mathcal{L})$ is possibly a No instance. Unfortunately, we did not succeed in obtaining such a precise bound and we have thus focused in [Fertin et al., 2009b] on the determination of two bounds c_{low} and c_{up}, $c_{low} \leq c_{up}$, such that if $C(G, H, \mathcal{L}) > c_{up}$ then $(G, H, \mathcal{L})$ is a Yes instance, and if $C(G, H, \mathcal{L}) \leq c_{low}$ then $(G, H, \mathcal{L})$ is possibly a No instance. Of course, the smaller c_{up} and $c_{up} - c_{low}$ are, the better our estimation is.

Proposition 4.3.7 ([Fertin et al., 2009b]). *Let $(G, H, \mathcal{L})$ be any instance of the* EXACT-$(\mu_G, 1)$-MATCHING *problem. If $C(G, H, \mathcal{L}) > \frac{2\Delta(G)-1-e^{-1}}{2\Delta(G)-1}$ then there exists an injective homomorphism θ of G to H with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$. If $C(G, H, \mathcal{L}) \leq \frac{\Delta(G)-1}{\Delta(G)}$ then an injective homomorphism θ of G to H with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, might not exist.*

The upper-bound is by the Lovász local lemma [Erdös and Lovász, 1975]. According to this bound, if $\Delta(G) = 1$ (resp. $\Delta(G) = 2$, $\Delta(G) = 3$) and $C(G, H, \mathcal{L}) > 0.633$ (resp. $C(G, H, \mathcal{L}) > 0.878$, $C(G, H, \mathcal{L}) > 0.927$) then there exists an injective homomorphism θ of G to H with respect to the lists $\mathcal{L}(u)$. As for the lower-bound, for any $d > 1$, we provided a generic construction of an instance $(G, H, \mathcal{L})$ of the EXACT-$(\mu_G, 1)$-MATCHING problem with $\Delta(G) = d$ and $C(G, H, \mathcal{L}) \leq \frac{\Delta(G)-1}{\Delta(G)}$ for which there does not exist an injective homomorphism θ of G to H with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$.

Subsection 4.3.2 is concerned exclusively with the EXACT-$(\mu_G, 1)$-MATCHING problem. The rationale for considering $\mu_h = 1$ is that the problem of finding an injective homomorphism $\theta : G \to H$ with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, enjoys some "*degree of independence*". Indeed, for any distinct $u, v \in \mathbf{V}(G)$, we have $\theta(u) \neq \theta(v)$ in any solution θ since $\mathcal{L}(u) \cap \mathcal{L}(v) = \emptyset$. Extending Proposition 4.3.7 to any instance of the EXACT-(μ_G, μ_H)-MATCHING problem would be of particular interest. In particular, does there exist a constant c^* (possibly depending on μ_G and μ_H) such that if $C(G, H, \mathcal{L}) > c^*$ then there exists an injective homomorphism θ of G to H with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$? Most of these issues are completely unexplored.

4.4 Approximate occurrences

Requiring an injective homomorphism, *i.e.*, an injective mapping that preserves *all* edges of G, might result in an over-constrained problem, though it may exist good approximate solutions, *i.e.*, solutions that match *many but not all* edges of G. This remark is just common sense for practical considerations. We considered in [Fertin et al., 2009b] one possible approach to deal with approximate occurrences (see also upcoming Section 4.5 for a – from our point of view – more practical approach). We refer to this optimization problem as the MAX-(μ_G, μ_H)-MATCHING problem.

Max-(μ_G, μ_H)-Matching

• **Input** : Two graphs G and H, and the lists $\mathcal{L}(u) \subseteq \mathbf{V}(H)$, $u \in \mathbf{V}(G)$, such that (i) $\max\{|\mathcal{L}(u)| : u \in \mathbf{V}(G)\} \leq \mu_G$ and (ii) $\max\{|\mathcal{L}^{-1}(v)| : v \in \mathbf{V}(H)\} \leq \mu_H$.
• **Solution** : A mapping $\theta : \mathbf{V}(G) \to \mathbf{V}(H)$ with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, *i.e.*, $\theta(u) \in \mathcal{L}(u)$ for all $u \in \mathbf{V}(G)$.
• **Measure** : The number of edges conserved by θ, *i.e.*, $|\{\{u, v\} \in \mathbf{E}(G) : \{\theta(u), \theta(v)\} \in \mathbf{E}(H)\}|$.

Notice that for the MAX-(μ_G, μ_H)-MATCHING problem the solution mapping θ may not be (and in general is not) an injective graph homomorphism as it is not required to preserve *all* edges. Being a natural but mere restriction of the EXACT-(μ_G, μ_H)-MATCHING problem, the MAX-(μ_G, μ_H)-MATCHING problem inherits of all the negative results of it (see Section 4.3). Therefore, we focus on approximation and (unfortunately) on hardness of approximation. Indeed (and not surprisingly, I admit it), as we have shown in [Fertin et al., 2009b], turning the pure decision EXACT-(μ_G, μ_H)-MATCHING problem into an optimization one results in a harder problem (considering parameters μ_G and μ_H).

Proposition 4.4.1 ([Fertin et al., 2009b]). *The* MAX-$(1, 2)$-MATCHING *is* **APX***-hard even if* G *and* H *are bipartite graphs with* $\Delta(G) \leq 3$ *and* $\Delta(H) \leq 2$.

The above proposition gains in interest if we compare it with Proposition 4.3.1 and Proposition 4.3.2. Actually, it is an immediate consequence of Proposition 4.4.1 that the MAX-$(1, 2)$-MATCHING problem for $\Delta(G) = 3$ and $\Delta(H) = 2$ (resp. $\Delta(G) = 6$ and $\Delta(H) = 5$) is not approximable within ratio 1.0005 (resp. 1.0014), unless $\mathbf{P} = \mathbf{NP}$.

Proposition 4.4.2 ([Fertin et al., 2009b]). *The* MAX-$(\mu_G, 1)$-MATCHING *problem is approximable within ratio* $2\lceil 3\Delta(G)/5\rceil$ *if* $\Delta(G)$ *is even and within ratio* $2\lceil (3\Delta(G)+2)/5\rceil$ *if* $\Delta(G)$ *is odd.*

Actually, we have shown a somewhat stronger result in [Fertin et al., 2009b]: if the linear arboricity conjecture (see [Akiyama et al., 1981]) is true, then the MAX-$(\mu_G, 1)$-MATCHING is approximable within ratio $\Delta(G) + 2$ if $\Delta(G)$ is even and within ratio $\Delta(G) + 1$ if $\Delta(G)$ is odd, for any $\Delta(H)$ and any fixed μ_G. Notice that the linear arboricity conjecture has been shown to be asymptotically correct as $d \to \infty$ [Alon, 1988]

Using a straightforward application of the *probabilistic method* [Alon and Spencer, 1992] – a powerful tool for demonstrating the existence of combinatorial objects – we gave in [Fertin et al., 2005] a linear-time randomized $(\mu_G)^2$-approximation algorithm for the MAX-$(\mu_G, 1)$-MATCHING problem. We have improved this result in [Fertin et al., 2009b].

Proposition 4.4.3 ([Fertin et al., 2009b]). *There exists a randomized* $2\mu_G$*-approximation algorithm for the* MAX-$(\mu_G, 1)$-MATCHING *problem, for any* μ_G.

We close this section by discussing exponential issues of the MAX-$(\mu_G, 1)$-MATCHING problem. For any instance $(G, H, \mathcal{L})$ of the MAX-$(\mu_G, 1)$-MATCHING problem, we have shown in [Fertin et al., 2009b] that one may construct in polynomial-time a (unfortunately complicated) graph $I[G, H, \mathcal{L}]$, called the *incompatibility graph* of the instance, that satisfies the following properties:

1. there exists an injective mapping $\theta : \mathbf{V}(G) \to \mathbf{V}(H)$ with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, such that $|\{\{u, v\} \in \mathbf{E}(G) : \{\theta(u), \theta(v)\} \in \mathbf{E}(H)\}| \geq k$ if and only if the stability number of $I[G, H, \mathcal{L}]$ is at least k, and

2. $\Delta(I[G, H, \mathcal{L}]) \leq (\mu_G - 1)(2\,\mu_G\,\Delta(G) - \mu_G + 1)$.

Combining these properties, we have obtained the following result.

Proposition 4.4.4 ([Fertin et al., 2009b]). *The* MAX-$(\mu_G, 1)$-MATCHING *problem is solvable in* $O(m\,(D+1)^k)$ *time, where* m *is the number of edges of* G *and* $D = \Delta(I[G, H, \mathcal{L}])$.

Notice that D is fixed as long as $\Delta(G)$, μ_G and μ_H are fixed. Therefore, the MAX-$(\mu_G, 1)$-MATCHING problem is fixed-parameter tractable for parameter "*number of conserved edges*".

As the reader may have noticed, the approximation of the general MAX-(μ_G, μ_H)-MATCHING problem is almost completely unexplored. Indeed, we are still not be able to tackle the case $\mu_H > 1$. As noticed in the headache note Page 46, in case $\mu_H = 1$, any injective mapping $\theta : G \to H$ with respect to the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, enjoys some "*degree of independence*" that we do use for approximation design. Overcoming this difficulty remains a totally open but challenging problem.

4.5 Replacing lists by colors

We consider in this section a restriction of the MAX-(μ_G, μ_H)-MATCHING problem well-suited for better modeling specific applications. Indeed, one may argue that using lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, to represent the putative correspondences is not restrictive enough for most practical applications (although allowing a large degree of freedom in the design!). One very important objection is that one may reasonably asks for $\mathcal{L}(u) = \mathcal{L}(v)$ as soon as $\mathcal{L}(u) \cap \mathcal{L}(v) \neq \emptyset$ as a golden rule. This objection becomes evident in the context of protein-protein interaction networks where it is folklore to construct the putative correspondences (the lists) by (i) (BLAST) comparing the sequences two by two, (ii) adjusting a cutoff to construct a correspondence graph, and finally (iii) computing the connected components of the correspondence graph. See [Brevier et al., 2007] and [Brevier et al., 2009]. This additional constraint is better taken into account by using colored-vertices instead of the lists $\mathcal{L}(u)$, $u \in \mathbf{V}(G)$, in the MAX-(μ_G, μ_H)-MATCHING problem.

Let col be a set of colors and G equipped with a coloring mapping $\lambda : \mathbf{V}(G) \to \text{col}$. For any color $c_i \in \text{col}$, we denote by $\text{col}_G(c_i)$ the set of vertices of G that are colored with color c_i, *i.e.*, $\text{col}_G(c_i) = \{u \in \mathbf{V}(G) : \lambda(u) = c_i\}$. The *multiplicity* of λ in G, written $\text{mult}(G, \lambda)$, is the maximum number of occurrences of a color in G, *i.e.*, $\text{mult}(G, \lambda) = \max\{|\,\text{col}_G(c_i)| : c_i \in \text{col}\}$. Let G and H be two graphs and let $\theta : \mathbf{V}(G) \to \mathbf{V}(H)$ be an injective mapping. The set of edges of G that are preserved in H by θ is denoted by $\text{match}(G, H, \theta)$, *i.e.*, $\text{match}(G, H, \theta) = \{\{u, v\} \in \mathbf{E}(G) : \{\theta(u), \theta(v)\} \in \mathbf{E}(H)\}$. If both G and H are equipped with some colorings $\lambda_G : \mathbf{V}(G) \to \text{col}$ and $\lambda_H : \mathbf{V}(H) \to \text{col}$ of their vertices, a mapping $\theta : \mathbf{V}(G) \to \mathbf{V}(H)$ is said to be *with respect to* λ_G *and* λ_H if $\lambda_G(u) = \lambda_H(\theta(u))$ for every $u \in \mathbf{V}(G)$, *i.e.*, θ is a color-preserving mapping. For simplicity, we shall usually abbreviate such a mapping as $\theta : \mathbf{V}(G) \xrightarrow{\lambda_G, \lambda_H} \mathbf{V}(H)$.

Max-(ρ, σ)-Matching-Colors

- **Input** : Two graphs G and H together with the coloring mappings $\lambda_G : \mathbf{V}(G) \to \text{col}$ and $\lambda_H : \mathbf{V}(H) \to \text{col}$ with $\text{mult}(G, \lambda_G) = \rho$ and $\text{mult}(H, \lambda_H) = \sigma$.
- **Solution** : An injective mapping $\theta : \mathbf{V}(G) \xrightarrow{\lambda_G, \lambda_H} \mathbf{V}(H)$.
- **Measure** : The number of edges of G matched by the injective mapping θ, *i.e.*, $|\text{match}(G, H, \theta)|$.

We let EXACT-(ρ,σ)-MATCHING-COLORS stand for the extremal problem of finding an injective mapping $\theta : \mathbf{V}(G) \xrightarrow{\lambda_G,\lambda_H} \mathbf{V}(H)$ that matches all the edges of G, *i.e.*, θ is required to be an injective graph homomorphism as we have considered in Section 4.3. Also, we call an instance of both MAX-(ρ,σ)-MATCHING-COLORS and EXACT-(ρ,σ)-MATCHING-COLORS *colorful* if $\rho = 1$, *i.e.*, each color occurs once in the motif graph G.

Clearly, MAX-$(1,\sigma)$-MATCHING–COLORS and MAX–$(\mu_G,1)$–MATCHING are equivalent problems (colorful instances and disjoint lists do represent the same configuration). Then it follows that the MAX-$(1,\sigma)$-MATCHING-COLORS is approximable within ratio $2\lceil 3\Delta(G)/5\rceil$ if $\Delta(G)$ is even and within ratio $2\lceil (3\Delta(G)+2)/5\rceil$ if $\Delta(G)$ is odd, for any $\Delta(H)$ and any fixed σ_H (see Proposition 4.4.2).

We have proposed in [Brevier et al., 2009] a random walk algorithm to deal with exact colorful instances (recall that the O^* notation suppresses polynomial terms) Observe that the EXACT-$(1,\sigma)$-MATCHING-COLORS problem is easily solvable in $O^*(\sigma^n)$ time (n is the order of G): the easy brute-force algorithm tries all possible injective mappings $\theta : \mathbf{V}(G) \xrightarrow{\lambda_G,\lambda_H} \mathbf{V}(H)$ and returns the best one.

Proposition 4.5.1 ([Brevier et al., 2009]). *There exists a randomized algorithm that, given any instance (G,H) of the* EXACT-$(1,\sigma)$-MATCHING-COLORS *problem, returns an injective homomorphism $\theta : \mathbf{V}(G) \xrightarrow{\lambda_G,\lambda_H} \mathbf{V}(H)$ (if such a mapping exists) in $O(f(\sigma)^n)$ expected time (ignoring polynomial factors), where*

$$f(\sigma) = \frac{4\sigma(2\sigma-2)^3}{4(2\sigma-2)^3 + 27(2\sigma-3)}.$$

Recall that the MAX-$(1,2)$-MATCHING-COLORS problem for bipartite graphs G and H with $\Delta(G) = 3$ and $\Delta(H) = 2$ (resp. with $\Delta(G) = 6$ and $\Delta(H) = 5$) is **APX**-hard and is not approximable within ratio 1.0005 (resp. 1.0014), unless $\mathbf{P} = \mathbf{NP}$ [Fertin et al., 2009b]. Therefore, there is a natural interest to investigate the complexity issues of MAX-(ρ,σ)-MATCHING-COLORS for restricted graph classes. Our results are technical but quite negative.

Proposition 4.5.2 ([Brevier et al., 2009]). *The* MAX-$(3,3)$-MATCHING-COLORS *(resp.* MAX-$(2,2)$-MATCHING-COLORS*) problem is* **APX***-hard even if both G and H are linear forests (resp. trees with maximum degree 3).*

It remains open, however, whether the MAX-(ρ,σ)-MATCHING-COLORS problem for linear forests G and H is polynomial-time solvable in case $\rho < 3$. In the light of the negative results of Proposition 4.5.2, there is a natural interest on approximating the MAX-(ρ,σ)-MATCHING-COLORS problem for bounded-degree graphs. We have shown the following result.

Proposition 4.5.3 ([Brevier et al., 2009]). *For any ρ and σ, the* MAX-(ρ,σ)-MATCHING-COLORS *problem is approximable within ratio $3/2(\Delta_{min}+1)+\varepsilon$ for any $\varepsilon > 0$, where $\Delta_{min} = \min\{\Delta(G),\Delta(H)\}$.*

Central in the above result is a $(3/2+\varepsilon)$-approximation algorithm, for any $\varepsilon > 0$, for a new combinatorial problem that may be of independent interest [Brevier et al., 2009]: Given a graph G and a symmetric matrix $A = [a_{i,j}]$ of order m whose entries are natural integers, find a maximum cardinality matching $\mathcal{M} \subseteq \mathbf{E}(G)$ subject to the constraint that, for $1 \le i \le j \le m$, the number of edges in $\mathcal{M}$ having one end-vertex colored c_i and one end-vertex colored c_j is at most $a_{i,j}$.

Combining a random 2-labeling procedure (together with its induced cut) and a weighted bipartite matching algorithm, we have obtained in [Brevier et al., 2009] the following result.

Proposition 4.5.4 ([Brevier et al., 2009]). *There exists a randomized algorithm for the* MAX-(ρ,σ)-MATCHING-COLORS *problem with expected performance ratio 4σ.*

We have already mentioned that the MAX-$(3,3)$-MATCHING-COLORS problem is **APX**-hard even if both G and H are linear forests. Furthermore, according to Proposition 4.5.3, the MAX-(ρ,σ)-MATCHING-COLORS problem for linear forests is approximable within ratio $2(\Delta_{\min}+1) = 6$. This is strengthened by the following proposition (it is worth mentioning that the technique is based on 2-interval patterns).

Proposition 4.5.5 ([Brevier et al., 2009]). *For any* ρ *and* σ*, the* MAX-(ρ, σ)-MATCHING-COLORS *problem is approximable within ratio* 4 *in case both* G *and* H *are linear forests.*

We close this chapter by mentioning that we have proposed a better approximation for the MAX-$(2, 2)$-MATCHING-COLORS problem.

Proposition 4.5.6 ([Brevier et al., 2009]). MAX-$(2, 2)$-MATCHING-COLORS *is approximable within ratio* 1.1442.

5 Searching for connected occurrences

Contents

5.1 Introduction

With the advent of network biology [Sharan and Ideker, 2006; Alm and Arkin, 2003] and complex network analysis in general, the study of pattern matching problems in graphs has become more and more important. In this context, the term "*graph motif*"plays a central role.

Roughly speaking, there are two views of graph (or network) motifs. The older is the topological view where one basically ends up with certain subgraph isomorphism problems. For instance, the term "*network motif*" has been used to represent patterns of interconnections that occur in a network at frequencies much higher than those found in random networks [Shen-Orr et al., 2002; Wernicke, 2006] (Chapter 4 was concerned with such a view). By way of contrast, the second and more recent view on graph motifs takes a more "*functional approach*". Here, topology is of lesser importance but the functionalities of network nodes (expressed by colors) form the governing principle. This approach has been propagated by Lacroix *et al.* [Lacroix et al., 2006].

This chapter is organized as follows. Section 5.2 presents some preliminaries. Section 5.3 is devoted to studying algorithmic aspects of the problem of finding a connected occurrence of a motif in a graph.

Section 5.4 and Section 5.5 are concerned with optimization issues of this topic, and we briefly present in Section 5.6 further variants of this problem we are particularly interested in.

5.2 Definitions

A *multiset* (or *bag*) is a pair (A, mult), where A is some set and $\text{mult} : A \to \mathbb{N}^*$ The set A is called the *underlying set* of elements. For each $a \in A$, the *multiplicity* (that is, the number of occurrences) of a is the number $\text{mult}(a)$. The *maximum multiplicity* of (A, mult) is defined to be $\max\{\text{mult}(a) : a \in A\}$. It is common to write the function mult as a set of ordered pairs $\{(a, \text{mult}(a)) : a \in A\}$. For example, $\{(a, 2), (b, 3), (c, 1)\}$ is the multiset $(\{a, b, c, d\}, \text{mult})$, where $\text{mult} : A \to \mathbb{N}^*$ is defined by $\text{mult}(a) = 2$, $\text{mult}(b) = 3$, and $\text{mult}(c) = 1$.

We shall consider here motifs given in the form of multisets of colors and we write $\mathcal{M} = (\text{col}, \text{mult})$. A motif $\mathcal{M}$ is called *colorful* if it has maximum multiplicity 1, *i.e.*, $\mathcal{M}$ reduces to a set.

The problem we are interested in is formally defined as follows.

Color-Matching

- **Input** : A set of colors col, a motif $\mathcal{M} = (\text{col}, \text{mult})$, and a vertex colored graph (G, λ), where $\lambda : \mathbf{V}(G) \to \text{col}$ is the coloring mapping.
- **Question** : Does there exist a connected induced subgraph of G colored by $\mathcal{M}$, *i.e.*, a subset $V' \subseteq \mathbf{V}(G)$ such that (i) $G[V']$ is connected, and (ii) $\lambda(V') = \mathcal{M}$?

In other words, we are asked to find a connected subgraph of G with $|\mathcal{M}|$ vertices which is exactly colored with the colors of $\mathcal{M}$ (including multiplicities, if any). See Figure 5.1 for an illustration.

The different vertex colors are used to model different functionalities. Although originally introduced in a biological context [Lacroix et al., 2006; Fellows et al., 2007], it is conceivable that the GRAPH MOTIF is an interesting problem not only for biological networks, but also may prove useful when studying complex social or technical networks (this remark is also in Betzler et al. [2008]).

5.3 Searching for exact connected occurrences

5.3.1 Polynomial-time and hardness results

The GRAPH MOTIF problem has been shown to be **NP**-complete even if the target graph G is a tree in [Lacroix et al., 2006]. The following proposition complete this result (one may easily notice that the GRAPH MOTIF problem is polynomial-time solvable if $\Delta(G) \leq 2$).

Proposition 5.3.1 ([Fellows et al., 2007]). *The two following variants of the* GRAPH MOTIF *problem are* **NP***-complete:*

1. *the target* G *is a bipartite graph,* $\Delta(G) = 4$*, and* λ *is a proper 2-coloring of* G*, and*
2. *the target* G *is a tree,* $\Delta(G) = 3$*, each color occurs at most three times in* G*, and* $\mathcal{M}$ *is a colorful motif.*

We did not succeed in proving that the GRAPH MOTIF problem is **NP**-complete if the target G is a bipartite graph, $\Delta(G) = 3$, and λ is a (not necessarily) proper 2-coloring of G. However, we conjecture this restriction to be **NP**-complete.

Defining the precise tractability landscape of the GRAPH MOTIF is of particular interest to strengthen hardness results. The following proposition shows that the jump in complexity is sudden and confirms that the second item of Proposition 5.3.1 is the best possible.

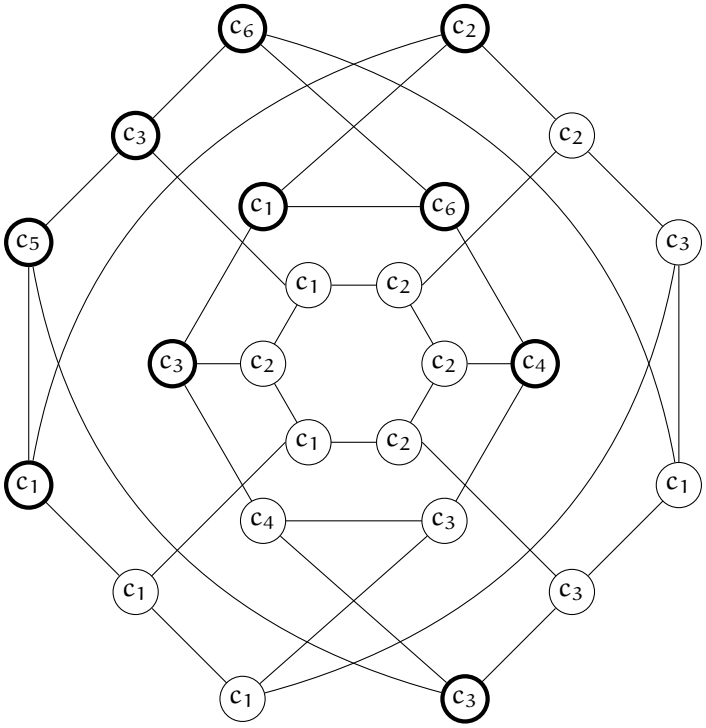


Figure 5.1: A vertex-colored graph (the pancake network of order 4) together with an occurrence (in bold) of the motif $\mathcal{M} = \{c_1, c_1, c_2, c_3, c_3, c_3, c_4, c_5, c_6, c_6\}$.

Proposition 5.3.2 ([Fellows et al., 2007]). *The* GRAPH MOTIF *problem is solvable in polynomial-time if the target* G *is a tree, each color occurs at most two times in* G*, and* $\mathcal{M}$ *is a colorful motif.*

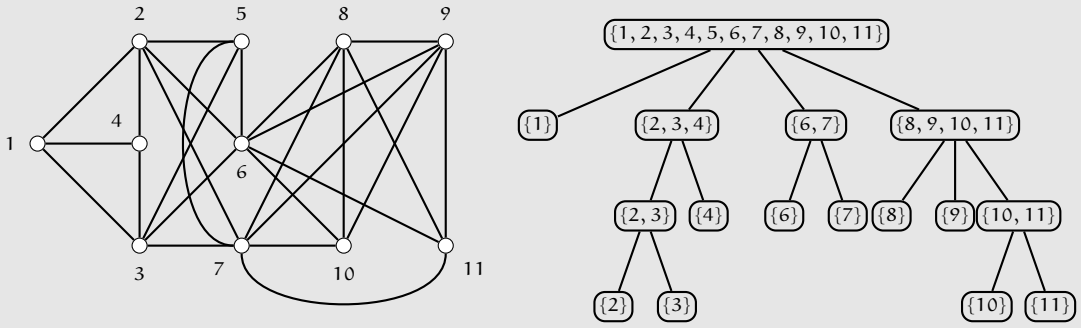
5.3.2 Parameterized complexity

In their pioneered work [Lacroix et al., 2006], the GRAPH MOTIF problem was proved to fixed-parameter tractable when parameterized by the size of the given motif (*i.e.*, $|\mathcal{M}|$), in case the target graph is a tree. However, as observed in [Lacroix et al., 2006], their fixed-parameter algorithm does not apply when the vertex-colored graph is a general graph. For this case, they only provided a heuristic algorithm which works well in practice. This motivates us to further investigate the tractability landscape of the GRAPH MOTIF problem. From our point of view, a notable breakthrough in the study of the GRAPH MOTIF problem for general graphs is that it is, as we shall see soon, fixed-parameter tractable when parameterized by the size of the motif $\mathcal{M}$ [Fellows et al., 2007]. This result is important in many ways. For one, it rests on a firm foundation and paves the way to further fixed-parameter algorithms (current approaches are still limited to motifs of size about 15 whereas practical applications do ask for motifs of size about 25–30 [Bruckner et al., 2009b], not an order of magnitude difference). For another, it motivates the investigation of the GRAPH MOTIF problem under different parameters which govern the structure of its input.

At the heart of our approach is the *color-coding technique* introduced by Alon *et al.* [Alon et al., 1995], whose derandomized version crucially relies on the notion of perfect hash families.

Definition 5.3.3 (Perfect Hash Family [Alon et al., 1995]). *Let* G *be a graph. A family* $\mathcal{F}$ *of functions from* $V(G)$ *to* $\{1, 2, \ldots, k\}$ *is* perfect *if for any subset* $V' \subseteq \mathbf{V}(G)$ *of* k *vertices there is a function* $f \in \mathcal{F}$ *which is one-to-one on* V'*.*

Aiming at accurate models, variants of the GRAPH MOTIF problem are greatly needed. To this aim, Betzler *et al.* [Betzler et al., 2008] replaced connectedness demand by more robust requirements, and proved the problem of finding a biconnected occurrence of $\mathcal{M}$ in G to be **W[1]**-complete when the parameter is the size of the motif. This result is of particular importance as it sheds light on the fact that a seemingly small step towards motif topology results in parameterized intractability. What about replacing the connectedness demand by modularity? Recall that a *module* in a graph G is a subset $V' \subseteq \mathbf{V}(G)$ such that the neighborhoods outside the module of the vertices within the module are all equal [McConnell and Spinrad, 1999]. The problem now becomes: Given a vertex-colored graph G and a motif $\mathcal{M}$, find a subset $V' \subseteq \mathbf{V}(G)$ such that (i) V' is colored by $\mathcal{M}$, and (ii) V' is a module in G? We do believe this direction is a promising line of research that we plan to expand in future works (this is actually a direction we are currently pursuing with F. Sikora). For one, considering modules makes sense in the general setting of graphs motifs. Indeed, a module is a set of vertices such that each vertex not in the module has a uniform relationship to all members of the module, *i.e.*, vertices of the module are indistinguishable from the outside. For another, the notion of modules is shipped with *modular decomposition trees*, *i.e.*, an organization in a tree of the *strong modules*. Below is an example of a graph together with its modular decomposition tree



Modular decomposition trees should certainly help for algorithm design. Sure enough, replacing connectedness by the notion of modules is not a strong enough relaxation (is it a relaxation?) to push the GRAPH MOTIF problem towards tractability (actually, definitively not!). However, we expect the modular tree decomposition to be a useful structure to design efficient fixed-parameter algorithms. At a more general level, there does not exist any precise definition of what a motif is (or should be) in a biological network, and hence we think that providing a parameterized toolbox incorporating several definitions of graph motifs could be of particular interest for practical applications.

Our result can be stated as follows.

Proposition 5.3.4 ([Fellows et al., 2007])**.** *The* GRAPH MOTIF *problem is solvable in* $2^{O(k)}\, n^2 \log(n)$ *time, where* $k = |\mathcal{M}|$ *and* n *is the number of vertices in the target graph* G.

The GRAPH MOTIF problem is thus fixed-parameter tractable when parameterized by $|\mathcal{M}|$. We only sketch the main ideas to prove Proposition 5.3.4. Suppose $\mathcal{M}$ has an occurrence V' in G, and suppose

we are provided with a perfect family $\mathcal{F}$ of functions from $\mathbf{V}(G)$ to $\{1, 2, \ldots, k\}$. Since $\mathcal{F}$ is perfect, we are guaranteed that at least one function in $\mathcal{F}$ assigns V' with k distinct labels. Let $f \in \mathcal{F}$ be such a function. For a given $L \subseteq \{1, 2, \ldots, k\}$, we define $\mathcal{M}_L(v)$ to be the family of all motifs $\mathcal{M}' \subseteq \mathcal{M}$, $|\mathcal{M}'| = |L|$, for which there exists an occurrence V'' with $v \in V'$, such that the set of (unique) labels that f assigns to V'' is exactly L. Since $\mathcal{M}$ occurs in G, we know that $\mathcal{M} \in \mathcal{M}_{\{1,2,\ldots,k\}}(v)$ for some $v \in \mathbf{V}(G)$. To determine whether $\mathcal{M}$ occurs in G, we apply a dynamic programming to compute $\mathcal{M}_L(v)$ for all $v \in \mathbf{V}(G)$ and $L \subseteq \{1, 2, \ldots, k\}$. Now, fix L to be some subset of $\{1, 2, \ldots, k\}$, and let v be any vertex of G. Our goal is thus to compute $\mathcal{M}_L(v)$ assuming $\mathcal{M}_{L'}(u)$ has been previously computed for every vertex $u \in \mathbf{V}(G)$ and any $L' \subseteq L \setminus \{f(v)\}$. The straightforward approach is to consider small motifs occurring at neighbors of v. However, a motif occurring at v might be the union of motifs occurring at any number of neighbors of v, and so this approach might require exponential running time in n. We have shown in [Fellows et al., 2007] that there exists an alternative method for computing $\mathcal{M}_L(v)$ that uses an even more naive approach, but one that requires exponential-time only with respect to k. Notice that while the motifs computed by our algorithm are in general multisets of colors, the procedure always considers sets of distinct labels.

The tree-width parameter of graphs [Robertson and Seymour, 1986] plays a central role in designing exact algorithms for many **NP**-hard graph problems [Arnborg and Proskurowski, 1989; Bodlaender, 1993]. Using tree decompositions and nice tree decompositions of arbitrary graphs (in a somewhat nonstandard way to tailor-fit our purposes), we have shown that the GRAPH MOTIF problem is polynomial-time solvable when the target graph G has constant tree-width and $\mathcal{M}$ consists of a constant number of colors (but arbitrary number of elements). This should be compared with the sharp hardness result of Proposition 5.3.1 which states that there are rather restricted classes of graphs, such as bounded degree bipartite graphs, where the GRAPH MOTIF problem is **NP**-complete even when $\mathcal{M}$ is built over only two colors.

Proposition 5.3.5 ([Fellows et al., 2007]). *The* GRAPH MOTIF *problem is solvable in* $O^*\left(2^\omega 2^{2c}\right)$ *time, where* ω *is the tree-width of the target graph* G*, and* c *is the number of distinct colors in the motif* $|\mathcal{M}|$*.*

Although Proposition 5.3.5 gives a nice complementary result to the sharp hardness result of Proposition 5.3.1, it still leaves a certain gap. The following proposition closes this gap (by the negative).

Proposition 5.3.6 ([Fellows et al., 2007]). *The* GRAPH MOTIF *problem, parameterized by the number of distinct colors* c *in the motif* $\mathcal{M}$*, is* **W[1]***-hard for trees.*

Even if we do believe that the GRAPH MOTIF problem introduced by Lacroix *et al.* [Lacroix et al., 2006] has shed new light on graph motifs, it suffers from much the same weaknesses as all pure decision problem: it does not allow for approximate occurrences. The rest of this chapter is devoted to analyzing natural variants of the GRAPH MOTIF problem that deal with approximate occurrences. As we shall see, the GRAPH MOTIF problem enjoys several variants (reflecting different points of view) that deserve separate considerations.

5.4 Minimizing the number of connected components

We consider in this section the problem of finding an occurrence of a motif $\mathcal{M}$ in a vertex-colored graph that results in a minimum number of connected components. This problem has been first considered in [Dondi et al., 2007], and [Betzler et al., 2008] presents some additional interesting results. We refer to this problem as the MINIMUM CC problem (MINIMIZING THE NUMBER OF CONNECTED COMPONENTS). It is formally defined as follows.

Minimum CC

- **Input** : A set of colors col, a motif $\mathcal{M} = (\text{col}, \text{mult})$, and a vertex colored graph (G, λ).
- **Solution** : A subset $V' \subseteq \mathbf{V}(G)$ such that $\lambda(V') = \mathcal{M}$.
- **Measure** : The number of connected components in the induced subgraph $G[V']$.

In other words, we are asked to find a subgraph of G with $|\mathcal{M}|$ vertices which is exactly colored with the colors of $\mathcal{M}$ (including multiplicities, if any) that induces a minimum number of connected components. See Figure 5.3 for an illustration.

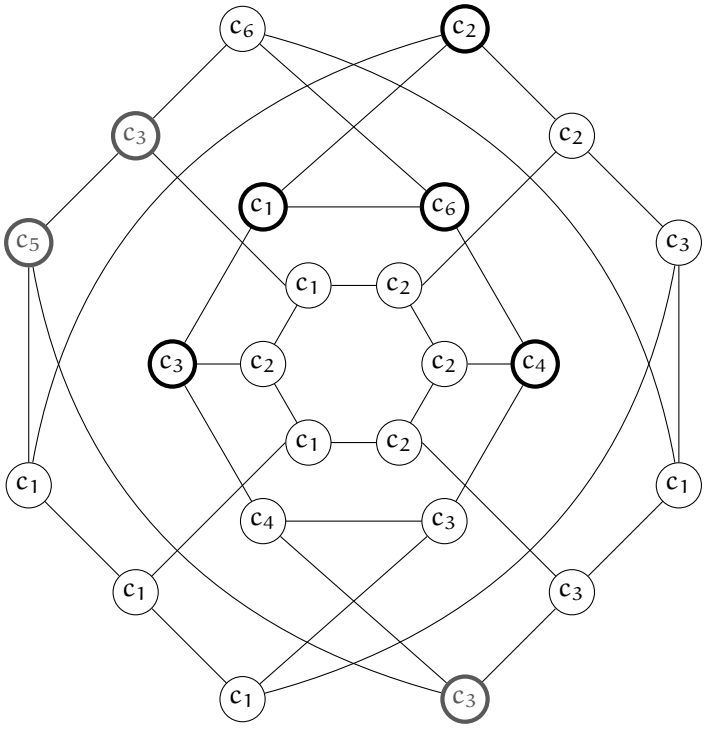


Figure 5.2: A vertex-colored graph together with an occurrence (in bold) of the motif $\mathcal{M} = \{c_1, c_1, c_2, c_3, c_3, c_4, c_5, c_6\}$ that results in two connected components, *i.e.*, $\{c_1, c_2, c_3, c_4, c_6\}$ and $\{c_3, c_3, c_5\}$.

5.4.1 Algorithms and hardness

It turns out that the MINIMUM CC problem is the most difficult variant of the GRAPH MOTIF problem if one focuses on graph classes. Let us explain this point. We need some new definitions. Define an *isogram* to be a word in which no letter is used more than once, and a *pair isogram* to be a word in which each letter occurs exactly twice. A *cover* of size k of a word u is an ordered collection of words $C = (v_1, v_2, \ldots, v_k)$ such that $u = w_1 v_1 w_2 v_2 \ldots w_k v_k w_{k+1}$ for some words $w_1, w_2, \ldots, w_{k+1}$ and $v = v_1 v_2 \ldots v_k$ is an isogram The cover is called *prefix* (resp. *suffix*) if w_1 (resp. w_{k+1}) is the empty word. Strongly related are proper 2-colorings. A *proper 2-coloring* of a pair isogram u is an assignment f of colors c_1 and c_2 to the letters of u such that every letter of u is colored with color c_1 once and colored with color c_2 once. If two adjacent letters x and y are colored with different colors we say that there is a *color change* between x and y.

***Example* 2** Consider the pair-isogram $u = a\,b\,b\,c\,d\,d\,e\,e\,c\,a\,f\,g\,g\,f\,h\,h$. A cover C of u of size 4 is given by $C = (ab, de, c, gfh)$ and the associated proper 2-coloration of u is given by

$$u = \underline{a\ \ b}\ \ b\ \ c\ \ d\ \ \underline{d\ \ e}\ \ e\ \ \underline{c}\ \ a\ \ f\ \ g\ \ \underline{g\ \ f\ \ h}\ \ h.$$

∎

A word u is said to be crossing-free (resp. inclusion-free) if there do not exist indices $1 \leq i_1 < i_2 < i_3 < i_4 \leq |u|$ such that $u[i_1] = u[i_3] \neq u[i_2] = u[i_4]$ (resp. $u[i_1] = u[i_4] \neq u[i_2] = u[i_3]$). Does there exist a polynomial-time for computing a minimum cardinality cover of a crossing-free pair isogram? What about inclusion-free pair isograms?

How approximable is the MINIMUM CC problem for paths?

The 1-REGULAR-2-COLORS-PAINT-SHOP problem is defined as follows (see [Bonsma et al., 2006; Bonsma, 2003; Epping and Oertel, 2004] for the general PAINTSHOP-FOR-WORDS problem): Given a pair isogram u, find a 2-coloring f of u that minimizes the number of color changes in (u, f). Bonsma [Bonsma, 2003] proved that the 1-REGULAR-2-COLORS-PAINT-SHOP problem is **APX**-hard. Combining this with the fact that a minimum cardinality cover of a pair isogram cannot be both prefix and suffix (see [Dondi et al., 2007]), we have obtained the following result.

Proposition 5.4.1 ([Dondi et al., 2007]). *The* MINIMUM CC *problem is* **APX***-hard even if $\mathcal{M}$ is colorful and the target graph G is a path in which each color appears at most twice.*

Quite a negative result! The following proposition moderates the above proposition.

Proposition 5.4.2 ([Dondi et al., 2007]). *The* MINIMUM CC *problem for paths is solvable in $O(n^{c+4})$ time, where n is the number of vertices in the path and c is the number of distinct colors in the motif $\mathcal{M}$.*

Focusing on trees, we have obtained the following positive and negative results (the positive results being actually exponential-time algorithms).

Proposition 5.4.3 ([Dondi et al., 2007]). *There exists a constant $c > 0$ such that the* MINIMUM CC *problem for trees cannot be approximated within performance ratio $c \log(n)$, where n is the number of vertices in the tree.*

Proposition 5.4.4. *The* MINIMUM CC *problem for trees is solvable in $O(n^2 k^{(c+1)^2+1})$ time, where n is the number of vertices in the tree, k is the size of the motif $\mathcal{M}$, and c is the number of distinct colors in $\mathcal{M}$.*

Proposition 5.4.5 ([Dondi et al., 2007]). *The* MINIMUM CC *problem for trees is solvable in $O(n^2 2^{\frac{2n}{3}})$ time, where n is the number of vertices in the tree.*

5.4.2 Parameterized complexity

Extending our result (Proposition 5.3.4), we have shown the MINIMUM CC problem for its standard parameterization to be fixed-parameter tractable as well.

Proposition 5.4.6 ([Dondi et al., 2007]). *The* MINIMUM CC *problem is fixed-parameter tractable when parameterized by the size of the motif.*

The $O(4.32^k k^2 m \log(\varepsilon)|)$ time algorithm of [Betzler et al., 2008] uses (to speed-up the dynamic programming procedure) the technique of *fast subset convolution*. This novel technique was developed by Björklund *et al.* [Björklund et al., 2007], who used it to speed-up several dynamic programming algorithms including the algorithm by Scott *et al.* [Ideker et al., 2006] for computing minimum weight size k trees in signaling networks.
From our point of view, fixed-parameter algorithmic results should support implementation and experimental work. It would be of particular interest to investigate whether the recently introduced subset convolution technique, which so far has been studied purely from a theoretical point of view, also yields a significant speed-up in practice.
A similar question may be asked as to how much color coding techniques [Alon et al., 1995] support implementation. (indeed, it turns out that the $O(4.32^k k^2 m \log(\varepsilon)|)$ time algorithm of [Betzler et al., 2008] increases the number of colors that are used for color-coding in order to increase the probability of an occurrence of $\mathcal{M}$ to receive a colorful recoloring, see [Huffner et al., 2007]). Is anybody aware of any implementation of perfect hash families to derandomize this approach? I suspect there isn't one, most approaches use a randomized color procedure and not perfect hash families. However, recent research on implementation on (randomized) color-coding based graph algorithms [Dost et al., 2007; Huffner et al., 2007; Ideker et al., 2006] are, undoubtly, positive experiences.

Our result is now superseded by [Betzler et al., 2008] where it is shown (in a clever way) that the MINIMUM CC problem can be solved with error probability ε in $O(4.32^k k^2 m \log(\varepsilon)|)$ time, where k is the size of the motif $\mathcal{M}$ and m is the number of edges in the target graph (see thinking note page 58).

The following proposition shows a sharp contrast in complexity if one considers the number of connected components as the parameter of interest.

Proposition 5.4.7 ([Dondi et al., 2007]). *The* MINIMUM CC *problem is* **W[2]***-hard when parameterized by the number of connected components, even if the target graph* G *is a tree.*

It is worth mentioning that, answering an question we have raised in [Dondi et al., 2007], N. Betzler *et al.* [Betzler et al., 2008] have recently proved the MINIMUM CC problem to be **W[1]**-hard only for paths (proof from the **W[1]**-hard PERFECT CODE problem).

5.5 Maximizing the size of the connected occurrence

We now turn to another variant of the GRAPH MOTIF problem where one is interested in obtaining a single connected component (as in the original GRAPH MOTIF problem) at the cost of *"loosing"*some colors. Several definitions actually would perfectly fit to this. We focus here on finding a connected occurrence that uses as much as possible colors from the motif (see next section for other possible definitions). We have introduced this variant of the GRAPH MOTIF in [Dondi et al., 2009] and we refer to it as the MAXIMUM MOTIF problem.

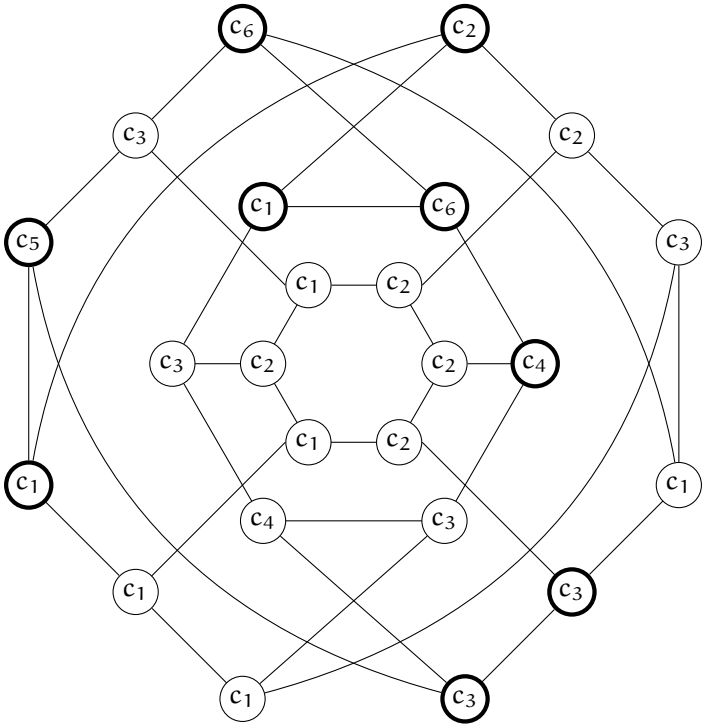


Figure 5.3: A vertex-colored graph together with an maximum cardinality occurrence (in bold) of a submotif $\mathcal{M}' = \{c_1, c_1, c_2, c_3, c_3, c_4, c_5, c_6, c_6, \}$ of the motif $\mathcal{M} = \{c_1, c_1, c_2, c_3, c_3, c_4, c_4, c_5, c_6, c_6, \}$.

Maximum Motif

- **Input** : A set of colors col, a motif $\mathcal{M} = (\text{col}, \text{mult})$, and a vertex colored graph (G, λ).
- **Solution** : A subset $V' \subseteq \mathbf{V}(G)$ such that (i) $G[V']$ is connected, and (ii) $\lambda(V') = \mathcal{M}'$ for some submotif $\mathcal{M}' \subseteq \mathcal{M}$.
- **Measure** : The size of $\mathcal{M}'$, *i.e.*, $|\mathcal{M}'|$.

Intuitively, the MAXIMUM MOTIF problem thus asks for the largest submotif $\mathcal{M}' \subseteq \mathcal{M}$ that occurs in G as a connected component. See Figure 5.3 for an illustration. Being a mere restriction of the GRAPH MOTIF problem, the MAXIMUM MOTIF problem is **NP**-complete as well [Lacroix et al., 2006].

5.5.1 Algorithms and hardness

Not surprisingly, the MAXIMUM MOTIF problem is hard to approximate. The following proposition prove that the MAXIMUM MOTIF problem does not enjoy a PTAS even for trees and colorful motifs.

Proposition 5.5.1 ([Dondi et al., 2009]). *The* MAXIMUM MOTIF *problem is* **APX***-hard even if the motif is colorful, the target graph is a tree with maximum degree* 3*, and each color occurs at most twice in the tree.*

It is worth mentioning that we do believe Proposition 5.5.1 not to be tight since we seriously doubt the MAXIMUM MOTIF problem for colorful motifs and trees with bounded number of occurrences of colors to be even in **APX**. The following proposition supports this sentiment (the rather technical proof is by the

self-improvement technique, see for example [Hein et al., 1996; Jiang and Li, 1995; Karger et al., 1995] to see this powerful technique in action).

Proposition 5.5.2 ([Dondi et al., 2009]). *For any constant $\delta < 1$, the* MAXIMUM MOTIF *problem for trees and colorful motifs cannot be approximated within performance ratio $2^{\log^{\delta} n}$, unless* **NP** $\subseteq$ **DTIME**$[2^{\text{poly}\log n}]$.

First, notice that **NP** $\not\subseteq$ **DTIME**$[2^{\text{poly}\log n}]$ is considered to be a reasonable complexity hypothesis (and is actually widely believed to be true). It is also worth noticing that, as we have shown in [Dondi et al., 2009], substituting the complexity hypothesis **NP** $\subseteq$ **DTIME**$[2^{\text{poly}\log n}]$ by the classical **P** $=$ **NP** yields inapproximability within a constant ratio. Second, the only difference in the instances between Proposition 5.5.1 and Proposition 5.5.2 is that the number of occurrences of each color is fixed in the former. Although at first odd, we believe that this restriction is not stronger enough to imply membership to **APX**.

5.5.2 Algorithms and parameterized complexity

In the light of the negative results for approximating the MAXIMUM MOTIF problem, we turn to exponential-time algorithms and parameterized complexity. We gave in [Dondi et al., 2009] two exact branch-and-bound algorithms for the MAXIMUM MOTIF problem in case the target graph is a tree. The two results are summarized in the following proposition.

Proposition 5.5.3 ([Dondi et al., 2009]). *The* MAXIMUM MOTIF *problem for trees of size n can be solved in $O(1.62^n \text{ poly}(n))$ time. In case the motif is colorful, the time complexity reduces to $O(1.33^n \text{ poly}(n))$.*

Based on the color-coding technique and perfect hash families [Alon et al., 1995], we have considered in [Dondi et al., 2009] parameterized issues of the MAXIMUM MOTIF problem. Our results can be stated as follows.

Proposition 5.5.4 ([Dondi et al., 2009]). *The* MAXIMUM MOTIF *problem for trees with n vertices is solvable in $O(k2^k n^3 \log n)\, 2^{O(k)}$ time, where k is the size of the submotif occurring in the tree. For general graphs of order n, the* MAXIMUM MOTIF *problem is solvable in $O(2^{5k} k n^2 \log^2 n)\, 4^{O(k)}$ time, where k is the size of the submotif occurring in the graph.*

One should admit that our parameterized results are still far from being able to support implementation. However, we believe that the color coding approach reaches its limits here and that improving the running-time of our algorithms requires different techniques.

5.6 Further variants

This short section is devoted to briefly presenting further variants of the GRAPH MOTIF problem we are interested in. Indeed, as stated in the preceding section, relaxing the GRAPH MOTIF problem to allow for approximate solutions may lead to distinct combinatorial problems (the MINIMUM CC and MAXIMUM MOTIFS problems are two such possibilities). We are especially interested in variants where one is searching for a single connected component at the cost of loosing/adding/modifying some colors as it seems that this requirement is well-suited for practical applications.

- The MINIMUM DELETE MOTIF problem: Find a submotif $\mathcal{M}' \subseteq \mathcal{M}$ that occurs as a connected component in the target graph. The optimization is concerned with deleting a minimum number of colors in $\mathcal{M}$.
- The MINIMUM ADD MOTIF problem: Find a supermotif $\mathcal{M}' \supseteq \mathcal{M}$ that occurs as a connected component in the target graph. The optimization is concerned with adding a minimum number of colors to $\mathcal{M}$, or equivalently minimizing $|\mathcal{M}'|$.

- The MINIMUM SUBSTITUTION MOTIF problem: Find a motif $\mathcal{M}'$ (related to $\mathcal{M}$) that occurs as a connected component in the target graph. The optimization is concerned with modifying a minimum number of colors in $\mathcal{M}$ to obtain $\mathcal{M}'$.

Being mere variants of the GRAPH MOTIF problem, all these variants are of course **NP**-complete (actually, it turns out that they are all **APX**-hard or not approximable). However, they correspond to different questions one may ask for connected motifs. Hereafter we mention some thoughts about these three problems.

- In terms of optimal solutions, the MAXIMUM MOTIF and the MINIMUM DELETE MOTIF problems are clearly equivalent. The comparison stops there. Indeed, considering approximation, dual combinatorial problems usually enjoys opposite properties (this is not a rule, I admit, only a trend). As for the parameterized complexity, the two problems seem to behave radically differently. Without going into the details, we just mention that, oppositely to the MAXIMUM MOTIF problem, the MINIMUM DELETE MOTIF problem for its standard parameterization is *not* fixed-parameter tractable.

- The MINIMUM ADD MOTIF problem seems to be more well-suited than the MINIMUM CC problem for most applications. Indeed, in the MINIMUM CC problem, the focus is on the number of connected components, regardless whether these connected components are arbitrary far in the graph. In some sense, the MINIMUM ADD MOTIF problem allows us to control this aspect by putting the focus on the number of colors to add to the motif to connect all those connected components into a single one.

- Although being a natural variant of the GRAPH MOTIF problem, the MINIMUM SUBSTITUTION MOTIF problem is quite intriguing from an algorithmic point of view as it seems to add one level of freedom. Interestingly enough (but unfortunately), the MINIMUM SUBSTITUTION MOTIF problem parameterized by the number of substitutions is **W[2]**-hard.

5.6.1 Practical issues

As we have seen in this chapter, we are still far from being able to provide a complete algorithmic toolbox to deal with the many flavors of the GRAPH MOTIF problem. Most fixed-parameter algorithms (including ours) do not support implementation yet. However, bridging the gap between theory and practice is greatly needed for practical applications.

A first step towards providing an integrated algorithmic solution is the TORQUE web server [Bruckner et al., 2009b] (it implements the algorithms in [Bruckner et al., 2009a] for querying protein sets across species). TORQUE combines three approaches: a dynamic programming method utilizing color coding, integer linear programming and a fast heuristic based on shortest paths. Quoting the authors [Bruckner et al., 2009a]: "*TORQUE automatically selects the best method to apply at each stage and outputs the highest scoring match*". Actually, there is no magical trick, TORQUE relies on color coding if the motif is small enough (about 15 elements) and switches to linear programming otherwise.

In collaboration with G. Blin and F. Sikora, we have also developed an integrated algorithmic toolbox, named GraMoFoNe, to deal with the many flavors of the GRAPH MOTIF problem [Blin et al., 2010b]. Notice that, oppositely to TORQUE, GraMoFoNe is not a web server but a plugin for the popular cytoscape open source platform (`http://www.cytoscape.org/`). Another notable difference with TORQUE is that GraMoFoNe does not combine two techniques (color coding and linear programming) but uses boolean linear programming. The rationale for this choice is twofold. For one, equipped with such a framework, TORQUE is superseded by GraMoFoNe in terms of modeling as it allows to consider most variant of the GRAPH MOTIF problem (TORQUE is indeed limited to colorful motifs). Without going into the details, it is worth noticing that TORQUE and GraMoFoNe notably differ in the consideration of the connectedness property: whereas TORQUE simulates a flow algorithm and hence does need linear programming, GraMoFoNe simulates a breadth first search (BFS) procedure that can be performed by boolean linear programming. For another,

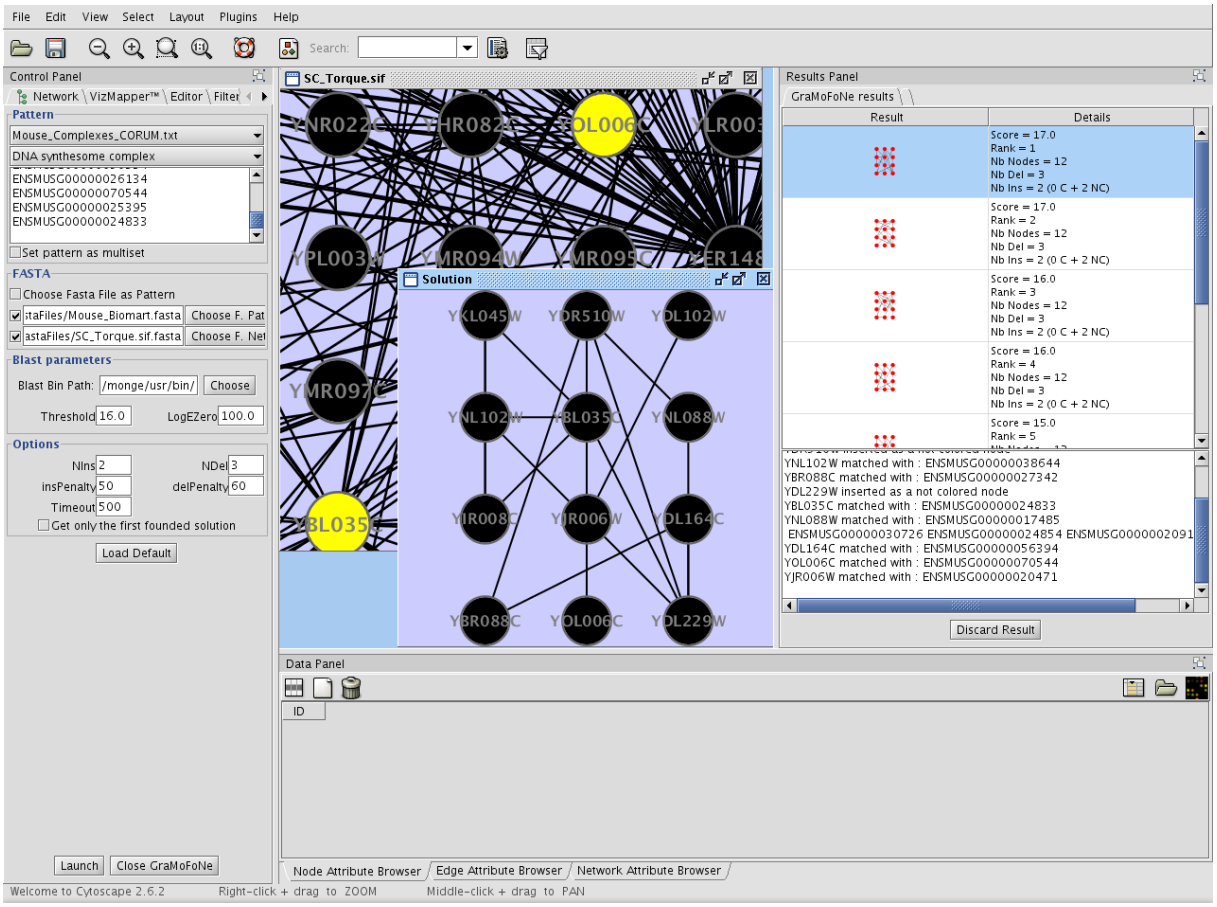


Figure 5.4: Screenshot of the GraMoFoNe software: Querying the mouse DNA synthesome complex in the yeast PPI network (see [Bruckner et al., 2009b]).

GraMoFoNe uses a pure pseudo-boolean programming engine together with some data reduction rules to speed-up the computations. See Figure 5.4 for a screenshot of GraMoFoNe in action.

TORQUE and GraMoFoNe perform more or less the same in terms of performances for moderate size tree motifs (GraMoFoNe is, however, not limited to trees). They also suffer from the same drawbacks: they are not able to deal with large motifs. However, GraMoFoNe is by far more scalable and is completely integrated in the cytoscape software (and hence can be easily used in combination with other cytoscape plugins). It is a challenging and important problem to improve GraMoFoNe so that it can tackle motifs of size about 30.

6 Querying PPI Networks

Contents

6.1 Introduction

This short chapter is devoted to a somewhat more classical view of pattern matching in protein-protein interaction (PPI) networks. Comparative analysis of PPI tries to determine the extent to which protein networks are conserved among species. Indeed, it was observed that proteins functioning together in a pathway (*i.e.*, a path in the interactions graph) or a structural complex (*i.e.*, an assembling of strongly connected proteins) are likely to evolve in a correlated fashion and during evolution, all such functionally linked proteins tend to be either preserved or eliminated in a new species [Pellegrini et al., 1999].

The classical view of PPI network querying is as follows: Given a PPI network and a pattern with a graph topology, find a subnetwork of the PPI network that is as similar as possible to the pattern, in respect to the initial topology. Similarity is measured both in terms of sequence similarity and graph topology conservation.

Unfortunately, this problem is clearly equivalent to the **NP**-complete SUBGRAPH HOMEOMORPHISM problem [Garey and Johnson, 1979]. Recently, several techniques have been proposed to overcome the difficulty of this problem. By restricting the query to a path of length less than five, Kelley *et al*. [Kelley et al., 2003] developped PathBlast, an exponential-time algorithm which allows one consecutive mismatch. Later on, Shlomi *et al*. [Shlomi et al., 2006] proposed an alternative, called QPath, for querying paths in a PPI network (the algorithm is based on the color coding technique [Alon et al., 1995]). By restricting the query to a tree, Pinter *et al*. [Pinter et al., 2005] proposed an algorithm that is restricted to forest PPI networks, *i.e.*, collection of trees. Finally, Dost *et al*. [Dost et al., 2007] developed QNet, a software to handle tree query in the general context of PPI networks. Of particular importance, [Dost et al., 2007] proposed an algorithm based on tree-decomposition for querying general graphs.

Indisputably, QNet is the state-of-the-art software to query PPI networks. Let us thus present it briefly (more precisely, let us present the implemented part of QNet since, as we shall see soon, this distinction is crucial in the present context). QNet is a fixed-parameter tractable randomized algorithm for querying

trees in (general) PPI networks. The complexity is $m2^{O(k)} \log(\epsilon^{-1})$ time, where k is the number of proteins in the query, m the number of edges of the PPI network and $1 - \epsilon$ is the probability of success (for any $\epsilon > 0$). Following the example of QPath, QNet combines (in a non-trivial way) a dynamic programming procedure together with the color-coding technique. This completely described the implemented part of QNet. However, QNet is shipped with an additional algorithm to query general graphs in PPI networks. At the heart of this algorithm is a procedure to transform the query graph into a tree (the technique is by tree-decomposition [Bodlaender, 1993]). The running time is $2^{O(k)} n^{\omega+1}$, where ω is the tree-width of the query graph (recall, however, that computing the tree-width of a graph is **NP**-complete [Arnborg et al., 1987]). A word of caution is necessary here. Indeed, the authors were not be able to implement this algorithm (this is probably due to its inherent difficulties at dealing with tree-decomposition). Nevertheless, even if they would succeed in implementing it, we highly suspected the huge constants hidden by the big-O notation to make it useless.

This chapter is devoted to presenting an *effective* alternative to QNet called PADA1 (I am not being held responsible for this Star Wars name!). It is organized as follows. Chapter 6.2 is intended to motivate our investigation. Chapter 6.3 is devoted to practical considerations of our contribution.

6.2 A feedback vertex set approach

PADA1 [Blin et al., 2009c] is an effective network querying algorithm that extends QNet to more general query graphs. Following the example of QNet, PADA1 is a two-step procedure: it first transforms the query graph into a tree and next uses that tree to effectively perform the query. Notice that it allows for insertions and deletions in the occurrence. While both QNet and PADA1 use a tree-like query, the two algorithms use totally different approaches. Indeed, whereas QNet is based on tree-decomposition, PADA1 focuses on the fact that most query graphs have relatively small feedback vertex set in practice (recall that a feedback vertex set is subset of vertices whose removal leaves us with a cycle-free graph).

Finding a smallest feedback vertex set is a well-known **NP**-complete problem [Garey and Johnson, 1979]. The current implementation of PADA1 transforms the query graph into a tree by iteratively finding a cycle, duplicating (and storing) a node on that cycle and finally breaking the cycle by edge deletion. More efficient approaches, including iterative compression [Guo et al., 2006] and reduction to kernel [Thomasse, 2009], may be used to identify a feedback vertex set and transform the query graph into a tree, but experimentations show that our "*brute-force*"algorithm turns out to be the fastest in practice. Indeed, (i) iteratively finding cycles relies on a fast BFS search (a $O(n + m)$ time procedure), (ii) the feedback vertex set of most real instances is very small, and finally (iii) finding an occurrence of the constructed tree into the PPI network is definitively the most time-consuming part of our approach.

The second step of PADA1 consists in finding an occurrence (allowing insertions and deletions) of the constructed tree into the PPI network. Our approach is by combining random coloring and dynamic programming (see [Blin et al., 2009c] for details). The main difficulty in this second step is to take into account all those duplicated vertices, and more precisely to group process all the copies of a same vertex (done by dynamic programming in the current implementation).

On the whole, the complexity of PADA1 is $O(mn^{|f|} N_{del} 2^{O(k+N_{ins})} \log(\epsilon^{-1}))$ time, where k is the number of proteins in the query, m the number of edges of the PPI network, $1 - \epsilon$ is the probability of success (for any $\epsilon > 0$), N_{ins} is the maximum number of insertions N_{del} is the maximum number of deletions, and f is the feedback vertex identifies in the very first part of the algorithm. Of particular importance, observe that the time complexity does not depend on the total number of duplicated nodes but on the size of the identified feedback vertex set (good, exactly what we were looking for).

6.3 Practical issues

We briefly discuss practical issues of PADA1. Since we wanted PADA1 to be an effective alternative to QNet, we have confronted our results in [Blin et al., 2009c] with thoses obtained by QNet. PADA1 has proved to perform as well as QNet (we refer the reader to [Blin et al., 2009c] for details). For example, our second experiment was performed across species. The Mitogen-Activated Protein Kinase (MAPK) is a collection of signal transduction queries. According to [Dent et al., 2003], they have a critical function in the cellular response to extracellular stimuli. They are known to be conserved through different species. We obtained the human MAPK from the KEGG database [Kanehisa et al., 2004] and tried to retrieve them in the fly network as done in QNet. While QNet uses only trees, we were able to query general graphs (See Figure 6.1 for an illustrative output of PADA1). The results were quite satisfying since we retrieved all of them, with only few modifications.

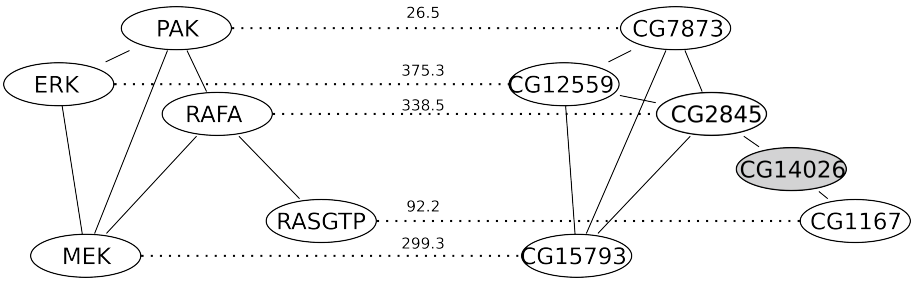


Figure 6.1: An automatic dot file generated by PADA1 (verbose output omitted). Left: A MAPK human query [Kanehisa et al., 2004]. Right: The alignment graph given provide by PADA1 in the fly PPI network. Dashed lines denotes the BLAST homology scores between the proteins. See [Dost et al., 2007] for a discussion on this particular query.

Part III

Genome Rearrangements

Introduction

The third part of this manuscript is concerned with comparative genomics. The combinatorial study of genome rearrangements started with permutations, but permutations lack the possibility of taking *duplications* into account. Duplications are however a major evolutionary event, believed to be one of the most important mechanisms for novel generations in evolution, and almost all datasets on eukaryotes contain duplicated genes (see e.g. [Ohno, 1970]). An appropriate tool for studying genomes with duplicated genes was therefore needed, and *strings* are a very natural generalization of permutations that fit this purpose well. It allows to add two possible rearrangement events: *duplications* and *deletions*. We shall see in this part that **NP**-completeness and even inapproximability results are very numerous. The subject was surveyed in 2005 by El-Mabrouk [2005]. The most up-to-date reference to this field is our recent monograph [Fertin et al., 2009a].

Biological motivations

Duplications can occur at several levels, ranging from the duplication of a single gene or small segment of DNA to the duplication of a whole chromosome, and even whole genome duplications are known to occur. These evolutionary events result in genomes in which where some markers are undifferentiable, and we call them duplicated *genes*.

Given a set of genomes, all copies of a given gene among those genomes are said to be *homologous*, which means that they originate from a common ancestral gene, and form a *gene family*. The presence of two copies of a gene in a set of genome may be explained by *speciation* events, that is, the apparition of two distinct species, each genome carrying the gene; it can also be explained by *duplication* events, which result in two copies of a gene in the same genome. The relationships of the copies of a gene in a gene family can thus be of several type. Two copies of a gene are said to be *orthologous* if they derive from a speciation event, and *Paralogous* if they derive from a duplication event. Given two genomes and a gene family, a distinction is made between *out-paralogs*, which are paralogous gene copies which derive from a duplication that occured *before* the last common ancestor of the two genomes, and *in-paralogs*, that derive from a duplication that occured *after* the last common ancestor. Note that the name *gene* is classically a bit ambiguous, as it refers either to a family (there are several copies of a same gene in the genomes), or to copies (two genes may derive from a duplication).

Those different situations motivate and justify the use of the models we will consider in this part. Indeed, when comparing two sequences under the assumption that all copies of a given element in a single string are

in-paralogs, the goal will be to identify the position of the unique ancestor. If there can be out-paralogs, then the goal will be to detect orthologs by matching some copies. The distances between two strings will vary according to which model is chosen. Every combinatorial problem we have seen so far can be reformulated in terms of strings, but the algorithmic treatment is usually completely different. For instance, the breakpoint graph, which is a ubiquitous objects when dealing with permutations, is not used on strings, in spite of some attempts to define them in the case of whole genome duplications by [Alekseyev and Pevzner, 2007].

How to deal with multiple copies?

Two strings on the same alphabet that contain the same number of occurrences of each gene family are said to be *balanced*. Two balanced strings obviously have same length. This property ensures that it is possible to transform one string into another without deletions and duplications as rearrangements.

It is not equally difficult to take into account multiple copies when considering two balanced or two general (that is, not necessarily balanced) strings. By convention, balanced strings are supposed to contain only out-paralogs, which means that each of the h members of some gene family present on each string S and T originates from one of the h members of the same gene family present on their last common ancestor. The difficulty is then to identify (that is, to match) the pairs of members, one on each string, which originate from the same member of the last common ancestor (*i.e.* are orthologous). On the contrary, general strings allow to assume the existence of both out-paralogs and in-paralogs on each string, so that deletion and insertion events have to be considered additionally to the rearrangement events when comparing general strings. The assignation of ortholog pairs of genes given two strings reduces to finding a matching between them.

While comparing two strings u and v, a matching between u and v is aimed at representing the common composition of the strings, as supported by their last common ancestor and regardless of (but without losing touch with) the order of the characters. Any pair of matched characters is then assumed to correspond to orthologous genes, while the unmatched characters are supposed to be in-paralogs. Here rearrangement studies meet the important problem of ortholog identifications. The members of the same gene family present on the same string and which are matched are out-paralogs. In order to distinguish out-paralogs from each other, a *relabeling* may be performed, which gives new and distinct names. to out-paralogs and renames the orthologous of each out-paralog accordingly. The last step of such a treatment of the strings is the obtention of a pruning. The good news at this stage is that if we assume the relabeling is done such that the characters in the pruned strings are integers, both strings are permutations and may be compared using the usual distances on permutations.

Now, going back to our initial question "*How to deal with multiple copies?*"two answers are available: either define a collection of possible rearrangement and compute the minimum number of operations needed to transform one genome into the other, or reduce genomes to permutations using matchings and pruning and then compute the distance (or (dis)similarity) between the permutations. We refer to these two approaches as the *block edit* model and the *match-an-prune* model , respectively. Part III of this manuscript is devoted to the match-and-prune model. For a thorough introduction to the block edit model we refer the reader to [Fertin et al., 2009a].

7 Genome rearrangements with duplicate genes

Contents

7.1 Introduction

This chapter is devoted to algorithmic aspects of the *match-an-prune* model for genome rearrangements. All problems follow the same guideline: start with two genomes with duplicate genes, *i.e.*, strings, and transform them into two permutations (by means of some special matching between the two genomes) so as to optimize a given distance or (dis)similarity measure. The rationale for this guideline is that most distances and (dis)similarity measures (more precisely those that are of interest in comparative genomics) are computable in polynomial-time for permutations. The difficulty of the problem is thus to compute a "*good*"transformation.

This chapter is organized as follows. Section 7.2 presents the relevant material thus making our exposition self-contained. Section 7.3 is concerned with complexity issues of genome comparisons and we discuss in Section 7.4 a family of fast general heuristics to cope with intractability.

7.2 From genomes to permutations ... and back

7.2.1 Genomes

A *signed genome* $\mathbf{G}$ is a string over the alphabet of integers (excluding 0, to avoid having to write $+0$ and -0). An *unsigned genome* is defined analogously by forbidding negative integers, this is a thus string over the alphabet of positive integers. In the context of comparative genomics, we refer to the letters of $\mathbf{G}$ as *genes* (the sign denotes the orientation of the gene). We follow standard string terminology and the size of a genome $\mathbf{G}$ is denoted $|\mathbf{G}|$. We write $\mathbf{G}[i]$ for the gene at position i in $\mathbf{G}$, $1 \leq i \leq |\mathbf{G}|$, and we denote its sign by $\text{sign}(\mathbf{G}[i])$. A *gene family* of $\mathbf{G}$ is a positive integer that occurs in $\mathbf{G}$ regardless of its sign (here are those famous duplications we are interested in). We will denote by $F(\mathbf{G})$ the set of gene families that occur in $\mathbf{G}$. For simplicity of notation, we write $g \in \mathbf{G}$ if $g \in F(\mathbf{G})$ (thus we may write $g \in \mathbf{G}$ even if the gene g occurs only negatively in $\mathbf{G}$). We will denote by $|\mathbf{G}|_g$ the number of occurrences of a gene family $g \in \mathbf{G}$, and we let $\deg(\mathbf{G})$ stand for the maximum number of occurrences of a gene family in $\mathbf{G}$, *i.e.*, $\deg(\mathbf{G}) = \max\{|\mathbf{G}|_g : g \in \mathbf{G}\}$. Of particular importance, notice that $\deg(\mathbf{G})$ is computed independently of the signs of the genes. A genome $\mathbf{G}$ is *duplication-free* if $|\mathbf{G}[i]| \neq |\mathbf{G}[j]|$ for all $1 \leq i < j \leq |\mathbf{G}|$. In other words, a genome is duplication-free if any gene occurs exactly once, regardless of its sign.

***Example* 3** For genome $\mathbf{G} = 1\ -4\ 2\ 3\ -1\ 2$, we have $F(\mathbf{G}) = \{1,2,3,4\}$, $|\mathbf{G}|_1 = 2$, $|\mathbf{G}|_2 = 2$, $|\mathbf{G}|_3 = 1$, $|\mathbf{G}|_4 = 1$, and hence $\deg(\mathbf{G}) = 2$. On the other hand, genome $\mathbf{H} = -2\ -1\ 3\ 5\ 4$ is duplication-free. ∎

Let $\mathbf{G}$ be a duplication-free genome of size n, and i and j, $1 \leq i < j \leq n$. If $g = \mathbf{G}[i]$ and $g' = \mathbf{G}[j]$, the *distance* between gene g and gene g' in $\mathbf{G}$, denoted $\text{dist}(\mathbf{G}, g, g')$, is defined by $\text{dist}(\mathbf{G}, g, g') = j - i$.

Definition 7.2.1 (Pegged genome)**.** *A genome* $\mathbf{G}$ *is* pegged *if each interval between two genes in the same gene family contains at least one singleton (a gene that occurs exactly once in* $\mathbf{G}$*).*

Pegged genomes have the interesting property that singletons act as markers helping to uniquely identify each occurrence of a non-singleton by its position with respect to these markers.

7.2.2 Permutations

The symmetric group on set $\{1, 2, \ldots, n\}$ is written $\mathfrak{S}_n = \mathfrak{S}(\{1, 2, \ldots, n\})$, and we let $\mathfrak{S}_n^0 = \mathfrak{S}(\{0, 1, \ldots, n\})$ stand for the symmetric groups on $\{0, 1, \ldots, n\}$. A *permutation* π of size n is a bijection $\pi : \mathfrak{S}_n \to \mathfrak{S}_n$. A classical notation used in combinatorics to denote a permutation π is the *two-row notation* , where one arranges the "*natural*"ordering of the elements being permuted on a row, and the new ordering on another row. For example,

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 3 & 1 \end{pmatrix}$$

stands for the permutation π of the set $\{1, 2, 3, 4, 5\}$ defined by $\pi(1) = 2$, $\pi(2) = 5$, $\pi(3) = 4$, $\pi(4) = 3$, and $\pi(5) = 1$. We will, however, adopt the more convenient – and standard – one-row notation that keeps only the second row. Going back to our example, $\pi = (2\ 5\ 4\ 3\ 1)$. The *identity permutation* $(1\ 2\ \ldots\ n)$ is denoted ι, regardless of n.

The *composition* of two permutations $\pi, \sigma \in \mathfrak{S}_n$, denoted $\pi \circ \sigma$, is defined by $\pi \circ \sigma = (\pi_{\sigma_1}\ \pi_{\sigma_2}\ \ldots\ \pi_{\sigma_n})$. For example for $\pi = (3\ 1\ 4\ 2)$ and $\sigma = (4\ 1\ 3\ 2)$, we have $\pi \circ \sigma = (2\ 3\ 4\ 1)$. The *inverse permutation* of $\pi \in \mathfrak{S}_n$ is the permutation π^{-1} defined by $\pi_i^{-1} = i$ for all $1 \leq i \leq n$.

Signed permutations model the organization of genomes better than unsigned permutations, because they take into account the double helix structure of DNA. A *signed permutation* on $\{1, 2, \ldots, n\}$ is a permutation π of the set $\{-n, \ldots, -2, -1, 1, 2, \ldots, n\}$ such that $\pi_{-i} = -\pi_i$ for all $1 \leq i \leq n$. The one-row notation is also used for signed permutations. For example, the permutation

$$\pi = \begin{pmatrix} -5 & -4 & -3 & -2 & -1 & 1 & 2 & 3 & 4 & 5 \\ -5 & 3 & -1 & 4 & 2 & -2 & -4 & 1 & -3 & 5 \end{pmatrix}$$

is simply written $\pi = (-2\ -4\ 1\ -3\ 5)$.

We recall here some basic definitions about permutations (we refer the reader to [Bóna, 2004] for general combinatorial aspects of permutations and to [Fertin et al., 2009a; Estivill-Castro and Wood, 1992] for applications to comparative genomics).

Definition 7.2.2 (linear extension)**.** *The* linear extension *of a (signed or unsigned) permutation $\pi \in \mathfrak{S}_n$ is the permutation $\pi^l \in \mathfrak{S}^0_{n+1}$ defined by $\pi^l = (0\ \pi_1\ \pi_2\ \dots\ \pi_n\ n+1)$.*

Definition 7.2.3. *Let π^l be the linear extension of a (signed or unsigned) permutation $\pi \in \mathfrak{S}_n$. A* point *is an ordered pair (π^l_i, π^l_{i+1}), $0 \leq i \leq n$. This point is called*

- *an* adjacency *if $\pi^l_{i+1} = \pi^l_i + 1$,*
- *a* reverse adjacency *if $\pi^l_{i+1} = \pi^l_i - 1$,*
- *a* breakpoint *if it is not an adjacency, and*
- *a* strong breakpoint *if it is neither an adjacency nor a reverse adjacency.*

Definition 7.2.4 (Interval (in a permutation))**.** *An* interval *in a (signed or unsigned) permutation $\pi \in \mathfrak{S}_n$ is a set $I = \{|\pi_i|, |\pi_{i+1}|, \dots, |\pi_j|\}$, $1 \leq i \leq j \leq n$. The elements π_i and π_j of I are called the* extremities *of the interval.*

Definition 7.2.5 (Common interval)**.** *An interval I is a* common interval *of permutations $\pi, \sigma \in \mathfrak{S}_n$ if it is an interval of both π and σ.*

In case $\sigma = \iota$, an interval $I = \{|\pi_i|, |\pi_{i+1}|, \dots, |\pi_j|\}$, $1 \leq i \leq j \leq n$, is a common interval of π and ι if I is a set of consecutive integers. The number of common intervals of two permutations π and σ is denoted $CI(\pi, \sigma)$. For $\pi, \sigma \in \mathfrak{S}_n$, it is easily seen that $n + 1 \leq CI(\pi, \sigma) \leq \binom{n}{2} + n$, The lower bound is attained, for example, if we take $\pi = (1\ 2\ 3\ 4)$ and $\sigma = (2\ 4\ 1\ 3)$ or $\sigma = (3\ 1\ 4\ 2)$ (the excluded patterns of separable permutations [Bose et al., 1998]!). The upper bound is attained for $\pi = \sigma$ or $\sigma = (\pi_n\ \dots\ \pi_2\ \pi_1)$.

7.2.3 Turning a genome into a permutation

How to deal with multiple copies? The match-and-prune model addresses the following question, which arises naturally when trying to discover the relationships between two genomes **G** and **H**: how can we take into account, when comparing genomes with duplicates, that the structure of the last common ancestor of **G** and **H** plays an important role in the evolutionary distance between the two genomes? As this structure is unknown, unless we have very good reasons to conclude we can afford to keep it unknown, the solution is to attempt to model it.

Three (sub)models are used to this aim. They have essential differences and essential common points. The differences come from different assumptions with respect to composition of the last common ancestor. The main common point is the method to compute measures (distances and similarities) between strings, which only takes into account their common composition, as identified by the composition of the last common ancestor. Speaking about the differences, the three models have the following features. In the *exemplar model* , the last common ancestor is assumed to contain exactly one member of each gene family which has members both on **G** and **H**. In the *intermediate model* , the last common ancestor is assumed to contain at least one members of each gene family which is common to **G** and **H**. Finally, in the *full model*, the last common ancestor is assumed to contain as many members as possible from any gene family.

We want to draw the attention of the reader on the fact that the intermediate introduce a level of difficulty. Indeed, observe that for the exemplar and full models, we know in advance the size of the resulting permutations as we keep exactly one gene or as many genes as possible from each gene family. The situation is different for the intermediate model as we do not know in advance how many genes of each gene family will be kept in an optimal solution.

The history of these three models starts with Sankoff's paper [Sankoff, 1999], who put the basis of the exemplar model and, in the same time, of the most general match-and-prune model. Besides its biological motivations, Sankoff's idea has two important features, that make it attractive. Reducing to one the cardinality of each gene family in each string implies that (1) the resulting strings are permutations, and computing distances on permutations is both already studied and often polynomial; and (2) the one-to-one correspondence of genes in the same family on the two strings is obvious, and thus may avoid further complications. The next model to be defined was the full model, whose first ideas are suggested in Sankoff's paper [Sankoff, 1999] and that was more precisely defined by [Tang and Moret, 2003] for balanced strings. The most recent one is the intermediate model we have introduced in [Angibaud et al., 2006].

Definition 7.2.6 (Matching). *Let* $\mathbf{G}$ *and* $\mathbf{H}$ *be two genomes. A* matching $\mathcal{M}$ *between* $\mathbf{G}$ *and* $\mathbf{H}$ *is a set of pairs*

$$\mathcal{M} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\} \subseteq \mathcal{P}(\{1, 2, \ldots, |\mathbf{G}|\} \times \{1, 2, \ldots, |\mathbf{H}|\})$$

such that

1. $|\mathbf{G}[i]| = |\mathbf{H}[j]|$ *for all pairs* $(i, j) \in \mathcal{M}$*, and*
2. *if* (i_l, j_l) *and* $(i_{l'}, j_{l'})$ *are two distinct pairs of* $\mathcal{M}$*, then* $l \neq l'$*.*

The $2k$ *genes* $\mathbf{G}[i_1], \mathbf{G}[i_2], \ldots \mathbf{G}[i_k], \mathbf{H}[i_1], \mathbf{H}[i_2], \ldots \mathbf{H}[i_k]$ *are said to be* saturated *by the matching* $\mathcal{M}$*.*

Notice here that we do allow $\mathbf{G}[i]$ and $\mathbf{H}[j]$, $(i, j) \in \mathcal{M}$, to have opposite signs. In the sequel, it will be enough to focus on compatible genomes as defined below.

Definition 7.2.7 (Compatible genomes). *Two genomes* $\mathbf{G}$ *and* $\mathbf{H}$ *are said to be* compatible *if* $F(\mathbf{G}) = F(\mathbf{H})$*.*

Definition 7.2.8 (Exemplar, intermediate and full matching). *A matching* $\mathcal{M}$ *between two compatible genomes* $\mathbf{G}$ *and* $\mathbf{H}$ *is an* intermediate matching *if* $\mathcal{M}$ *saturates at least one gene of each gene family of* $F(\mathbf{G}) = F(\mathbf{H})$*. An intermediate matching is called an* exemplar matching *(resp.* full matching*) if it is of minimum (resp. maximum) cardinality.*

Roughly speaking, intermediate matchings correspond to standard matchings in graphs (in bipartite graphs here) whereas exemplar and full matchings have additional constraints. In other words, exemplar and full matchings are in intermediate matchings.

***Example* 4** Consider the two following compatible genomes $\mathbf{G}$ and $\mathbf{H}$:

$$\mathbf{G} = 1\ 2\ -4\ -2\ 3\ 1\ 4\ -3\ 4$$
$$\mathbf{H} = 4\ 1\ -3\ -2\ 2\ 1\ 2\ 4.$$

with $F(\mathbf{G}) = \{1, 2, 3, 4\} = F(\mathbf{H})$.

1. The matching $\mathcal{M} = \{(1, 2), (2, 4), (5, 3), (6, 6), (7, 8)\}$ is an intermediate matching, and the pruned genomes $\mathbf{G}_{\mathcal{M}}$ and $\mathbf{H}_{\mathcal{M}}$ induced by $\mathcal{M}$ reduce to (indices $_a$ and $_b$ are used to disambiguate the induced mapping and clarify the presentation):

$$\mathbf{G}_{\mathcal{M}} = 1_a\ 2\ 3\ 1_b\ 4$$
$$\mathbf{H}_{\mathcal{M}} = 1_a\ -3\ -2\ 1_b\ 4.$$

The associated permutations (according to a relabeling so that $\pi_{\mathbf{G},\mathcal{M}} = \iota$) are thus given by:

$$\pi_{\mathbf{G},\mathcal{M}} = (1\ 2\ 3\ 4\ 5)$$
$$\pi_{\mathbf{H},\mathcal{M}} = (1\ -3\ -2\ 4\ 5).$$

2. The matching $\mathcal{M}' = \{(1,2),(2,4),(5,3),(9,8)\}$ is an exemplar matching. The pruned genomes $\mathbf{G}_{\mathcal{M}'}$ and $\mathbf{H}_{\mathcal{M}'}$ induced by $\mathcal{M}'$ reduce to:

$$\begin{aligned}\mathbf{G}_{\mathcal{M}'} &= 1\ 2\ 3\ 4\\ \mathbf{H}_{\mathcal{M}'} &= 1\ -3\ -2\ 4.\end{aligned}$$

The associated permutations (according to a relabeling so that $\pi_{\mathbf{G},\mathcal{M}'} = \iota$) are thus given by:

$$\begin{aligned}\pi_{\mathbf{G},\mathcal{M}'} &= (1\ 2\ 3\ 4)\\ \pi_{\mathbf{H},\mathcal{M}'} &= (1\ -3\ -2\ 4).\end{aligned}$$

3. The matching $\mathcal{M}'' = \{(1,6),(2,7),(3,1),(4,5),(5,3),(6,2),(9,8)\}$ is a full matching. The pruned genomes $\mathbf{G}_{\mathcal{M}''}$ and $\mathbf{H}_{\mathcal{M}''}$ induced by $\mathcal{M}''$ reduce to (once again, indices $_a$ and $_b$ are used to disambiguate the induced mapping):

$$\begin{aligned}\mathbf{G}_{\mathcal{M}''} &= 1_a\ 2_a\ -4_a\ -2_b\ 3\ 1_b\ 4_b\\ \mathbf{H}_{\mathcal{M}''} &= 4_b\ 1_b\ -3\ 2_b\ 1_a\ 2_a\ 4_a\end{aligned}$$

The associated permutations (according to a relabeling so that $\pi_{\mathbf{G},\mathcal{M}''} = \iota$) are thus given by:

$$\begin{aligned}\pi_{\mathbf{G},\mathcal{M}''} &= (1\ 2\ -3\ -4\ 5\ 6\ 7)\\ \pi_{\mathbf{H},\mathcal{M}''} &= (7\ 6\ 5\ 4\ 1\ 2\ 3).\end{aligned}$$ ∎

It is now a simple matter to see that exemplar, intermediate and full matchings coincide if $\deg(\mathbf{G}) = 1$ or $\deg(\mathbf{H}) = 1$, *i.e.*, $\mathbf{G}$ or $\mathbf{H}$ are duplication-free.

Definition 7.2.9 ($\mathcal{M}_{\text{expl}}(\mathbf{G},\mathbf{H})$, $\mathcal{M}_{\text{int}}(\mathbf{G},\mathbf{H})$, and $\mathcal{M}_{\text{full}}(\mathbf{G},\mathbf{H})$)**.** *For any two compatible genomes* $\mathbf{G}$ *and* $\mathbf{H}$*, we let* $\mathcal{M}_{expl}(\mathbf{G},\mathbf{H})$ *(resp.* $\mathcal{M}_{int}(\mathbf{G},\mathbf{H})$*,* $\mathcal{M}_{full}(\mathbf{G},\mathbf{H})$*) stand for the set of all exemplar (resp. intermediate, full) matchings between* $\mathbf{G}$ *and* $\mathbf{H}$*.*

The following definition will facilitate the exposition of subsequent sections.

Definition 7.2.10 ($\Pi_{\text{expl}}(\mathbf{G},\mathbf{H})$, $\Pi_{\text{int}}(\mathbf{G},\mathbf{H})$, and $\Pi_{\text{full}}(\mathbf{G},\mathbf{H})$)**.** *For any two compatible genomes* $\mathbf{G}$ *and* $\mathbf{H}$*, we define* $\Pi_{expl}(\mathbf{G},\mathbf{H})$*,* $\Pi_{int}(\mathbf{G},\mathbf{H})$ *and* $\Pi_{full}(\mathbf{G},\mathbf{H})$ *by:*

$$\begin{aligned}\Pi_{expl}(\mathbf{G},\mathbf{H}) &= \{(\pi_{\mathbf{G},\mathcal{M}},\pi_{\mathbf{H},\mathcal{M}}) : \mathcal{M} \in \mathcal{M}_{expl}(\mathbf{G},\mathbf{H})\},\\ \Pi_{int}(\mathbf{G},\mathbf{H}) &= \{(\pi_{\mathbf{G},\mathcal{M}},\pi_{\mathbf{H},\mathcal{M}}) : \mathcal{M} \in \mathcal{M}_{int}(\mathbf{G},\mathbf{H})\},\ \textit{and}\\ \Pi_{full}(\mathbf{G},\mathbf{H}) &= \{(\pi_{\mathbf{G},\mathcal{M}},\pi_{\mathbf{H},\mathcal{M}}) : \mathcal{M} \in \mathcal{M}_{full}(\mathbf{G},\mathbf{H})\}.\end{aligned}$$

The set $\Pi_{\text{expl}}(\mathbf{G},\mathbf{H})$ (resp. $\Pi_{\text{int}}(\mathbf{G},\mathbf{H})$, $\Pi_{\text{full}}(\mathbf{G},\mathbf{H})$) is thus the set of all permutations that correspond to valid exemplar (resp. intermediate, full) matchings between $\mathbf{G}$ and $\mathbf{H}$.

7.3 Comparing two compatible genomes

7.3.1 Introduction

We are now ready to compare genomes with respect to the match-and-prune model. Given two genome $\mathbf{G}$ and $\mathbf{H}$, our steps will be

1. find an matching (exemplar, intermediate or full) $\mathcal{M}$ between $\mathbf{G}$ and $\mathbf{H}$,

2. Construct the associated permutations $\pi_{\mathbf{G},\mathcal{M}}$ and $\pi_{\mathbf{H},\mathcal{M}}$, and

3. Compute the distance, similarity or dissimilarity measure we are interested in between $\pi_{\mathbf{G},\mathcal{M}}$ and $\pi_{\mathbf{H},\mathcal{M}}$.

Let π and σ be two signed permutations. We will focus in this section on the following standard measures:

- the *breakpoint distance* between π and σ, denoted $\mathrm{BK}(\pi, \sigma)$, is the number of breakpoints between π^1 and σ^1.
- the *signed reversal distance* between π and σ, denoted $\mathrm{SR}(\pi, \sigma)$, is the minimum number of signed reversals to transform π into σ, where a signed reversal is the operation of reversing an interval of π (together with signs).
- the *adjacency similarity* between π and σ, denoted $\mathrm{ADJ}(\pi, \sigma)$, is the number of adjacencies between π^1 and σ^1.
- the *common intervals similarity* between π and σ, denoted $\mathrm{CI}(\pi, \sigma)$, is the number of common intervals between π and σ.
- the MAD and SAD numbers whose precise definitions are deferred to the related subsection.

7.3.2 Breakpoint distance

Together with the signed reversal distance, the breakpoint distance is one of the first applications of Sankoff's exemplar model. The two distances are defined on different bases: the reversal distance counts a minimum number of operations to transform one genome into another one, whereas the breakpoint distance counts the structural differences between the two genomes. However, as we shall see soon, they are closely related.

Computing the breakpoint distance between two compatible signed genomes $\mathbf{G}$ and $\mathbf{H}$ reduces to finding a (exemplar, intermediate or full) matching between these two genomes that induces a minimum number of breakpoints between the two associated permutations $\pi_{\mathbf{G}}$ and $\pi_{\mathbf{H}}$.

Definition 7.3.1. *Let $\mathbf{G}$ and $\mathbf{H}$ be two signed compatible genomes. The measures BK_{expl}, BK_{int} and BK_{full} of $\mathbf{G}$ and $\mathbf{H}$ are defined by:*

$$\begin{aligned}
\mathrm{BK}_{expl}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{BK}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{expl}(\mathbf{G},\mathbf{H})\},\\
\mathrm{BK}_{int}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{BK}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{int}(\mathbf{G},\mathbf{H})\}, \text{ and}\\
\mathrm{BK}_{full}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{BK}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{full}(\mathbf{G},\mathbf{H})\}.
\end{aligned}$$

The measure $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G},\mathbf{H})$ has been introduced in [Sankoff, 1999], and the measure $\mathrm{BK}_{\mathrm{full}}(\mathbf{G},\mathbf{H})$ in [Blin et al., 2004]. As for the measure $\mathrm{BK}_{\mathrm{int}}(\mathbf{G},\mathbf{H})$, we have introduced it in [Angibaud et al., 2006]. In 2000, Bryant has shown that computing any of $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G},\mathbf{H})$, $\mathrm{BK}_{\mathrm{int}}(\mathbf{G},\mathbf{H})$ and $\mathrm{BK}_{\mathrm{full}}(\mathbf{G},\mathbf{H})$ is an **NP**-complete problem, even for pegged genomes $\mathbf{G}$ and $\mathbf{H}$ [Bryant, 2000]. Notice that Bryant's proof does not actually need to consider three separate cases as it holds even if $\deg(\mathbf{G}) = 1$ and $\deg(\mathbf{H}) = 2$. Nguyen has strengthened these results by proving that computing any of $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G},\mathbf{H})$ and $\mathrm{BK}_{\mathrm{int}}(\mathbf{G},\mathbf{H})$ is an **NP**-complete problem even if both $\mathbf{G}$ and $\mathbf{H}$ are unsigned pegged genomes [Nguyen, 2005]. The strongest inapproximability result known so far is ours.

Proposition 7.3.2 ([Angibaud et al., 2008b])**.** *Computing any of the three measures $\mathrm{BK}_{expl}(\mathbf{G},\mathbf{H})$, $\mathrm{BK}_{int}(\mathbf{G},\mathbf{H})$ and $\mathrm{BK}_{full}(\mathbf{G},\mathbf{H})$ is an* **APX***-hard problem.*

The proof of Proposition 7.3.2 actually holds for $\deg(\mathbf{G}) = 1$ and $\deg(\mathbf{H}) = 2$, and hence does not need to consider three separate cases.

Chen *et al.* [Chen et al., 2006] have shown that there exists a constant $c > 0$ such that there does not exist a polynomial-time algorithm with performance guarantee $c \log(n)$, $n = \max\{|\mathbf{G}|, |\mathbf{H}|\}$, to compute $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H})$ and $\mathrm{BK}_{\mathrm{int}}(\mathbf{G}, \mathbf{H})$. Of particular importance in this context, they have also shown that deciding whether $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H}) = 0$ is an **NP**-complete problem even if $\deg(\mathbf{G}) = 3$ and $\deg(\mathbf{H}) = 3$. The same result holds if we replace $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H}) = 0$ by $\mathrm{BK}_{\mathrm{int}}(\mathbf{G}, \mathbf{H}) = 0$. We have first completed the result of [Chen et al., 2006] in [Angibaud et al., 2008b] before proving a stronger result.

Proposition 7.3.3 ([Blin et al., 2009b]). *Deciding whether equality* $\mathrm{BK}_{expl}(\mathbf{G}, \mathbf{H}) = 0$ *holds is an* **NP***-complete problem even if* $\deg(\mathbf{G}) = 2$ *and* $\deg(\mathbf{H}) = 2$.

Notice that the above Proposition holds if we replace $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H}) = 0$ by $\mathrm{BK}_{\mathrm{int}}(\mathbf{G}, \mathbf{H}) = 0$. The above proposition carries definitive implications for research design in the form of the following corollary.

Corollary 7.3.4 ([Blin et al., 2009b]). *There does not exist any approximation algorithm to compute* $\mathrm{BK}_{expl}(\mathbf{G}, \mathbf{H})$ *or* $\mathrm{BK}_{int}(\mathbf{G}, \mathbf{H})$, *even if* $\deg(\mathbf{G}) = 2$ *and* $\deg(\mathbf{H}) = 2$.

The above corollary gains in interest if we notice that it precisely defines the inapproximability landscape. Indeed, if $\deg(\mathbf{G} =) = 1$ or $\deg(\mathbf{H}) = 1$, it can be shown that deciding whether equality $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H}) = 0$ holds is linear-time solvable. In other words, Proposition 7.3.3 is tight.

We mention to finish two results in this context that may be of independent interest. We have shown in [Angibaud et al., 2008b] that (i) deciding whether $\mathrm{BK}_{\mathrm{full}}(\mathbf{G}, \mathbf{H}) = 0$ holds is solvable in $O(nm \log\log(nm))$ time, where $n = |\mathbf{G}|$ and $m = |\mathbf{H}|$, and that (ii) computing $\mathrm{BK}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H})$ and $\mathrm{BK}_{\mathrm{int}}(\mathbf{G}, \mathbf{H})$ for two genomes $\mathbf{G}$ and $\mathbf{H}$ such that $\deg(\mathbf{G}) = 2$ and $\deg(\mathbf{H}) = 2$ is solvable in $O(\mathrm{poly}(k)\, 1.6182^{2k})$ time, where k is upper-bounded by the number of gene families that occur exactly twice in $\mathbf{G}$ and in $\mathbf{H}$.

7.3.3 Signed reversal distance

The signed reversal distance is the second distance considered by [Sankoff, 1999] to illustrate his theory of exemplar distances. Under the full model, the signed reversal distance is very well studied on balanced strings and not studied at all on general strings.

Computing the signed reversal distance between two compatible signed genomes $\mathbf{G}$ and $\mathbf{H}$ reduces to finding a (exemplar, intermediate or full) matching between these two genomes that induces a minimum signed reversal distance between the two associated permutations $\pi_{\mathbf{G}}$ and $\pi_{\mathbf{H}}$.

Definition 7.3.5. *Let* $\mathbf{G}$ *and* $\mathbf{H}$ *be two signed compatible genomes. The distances* SR_{expl}, SR_{int} *and* SR_{full} *of* $\mathbf{G}$ *and* $\mathbf{H}$ *are defined by:*

$$\begin{aligned}
\mathrm{SR}_{expl}(\mathbf{G}, \mathbf{H}) &= \min\{\mathrm{SR}(\pi_{\mathbf{G}}, \pi_{\mathbf{H}}) : (\pi_{\mathbf{G}}, \pi_{\mathbf{H}}) \in \Pi_{expl}(\mathbf{G}, \mathbf{H})\},\\
\mathrm{SR}_{int}(\mathbf{G}, \mathbf{H}) &= \min\{\mathrm{SR}(\pi_{\mathbf{G}}, \pi_{\mathbf{H}}) : (\pi_{\mathbf{G}}, \pi_{\mathbf{H}}) \in \Pi_{int}(\mathbf{G}, \mathbf{H})\}, \text{ and}\\
\mathrm{SR}_{full}(\mathbf{G}, \mathbf{H}) &= \min\{\mathrm{SR}(\pi_{\mathbf{G}}, \pi_{\mathbf{H}}) : (\pi_{\mathbf{G}}, \pi_{\mathbf{H}}) \in \Pi_{full}(\mathbf{G}, \mathbf{H})\}.
\end{aligned}$$

The distance $\mathrm{SR}_{\mathrm{expl}}$ has been introduced in [Sankoff, 1999], and the distance $\mathrm{SR}_{\mathrm{full}}$ in [Chen et al., 2005]. As far as we know, no specific result exists for the intermediate model.

Bryant has shown that computing $\mathrm{DR}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H})$ is an **NP**-complete problem even if $\mathbf{G}$ and $\mathbf{H}$ are pegged, and $\deg(\mathbf{G}) = 2$ and $\deg(\mathbf{H}) = 2$ [Bryant, 2000]. It turns out that the inapproximability results we gave for the breakpoint distance propagate to the signed reversal distance. Indeed, if $\mathbf{G}$ and $\mathbf{H}$ are two compatible genomes, then $2\ \mathrm{SR}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H}) \leq \mathrm{BK}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H}) \leq \mathrm{SR}_{\mathrm{expl}}(\mathbf{G}, \mathbf{H})$ (we refer the reader to our monograph [Fertin et al., 2009a] for an elementary proof).

7.3.4 Adjacency similarity

We consider in this subsection a similarity measure which is the complement of the breakpoint distance. The basis of this measure is the preserved adjacency between two consecutive characters in $\mathbf{G}$ and $\mathbf{H}$.

Computing the adjacency similarity between two compatible signed genomes $\mathbf{G}$ and $\mathbf{H}$ reduces to finding a (exemplar, intermediate or full) matching between these two genomes that induces a maximum number of adjacencies between the two associated permutations $\pi_{\mathbf{G}}$ and $\pi_{\mathbf{H}}$.

Definition 7.3.6. *Let $\mathbf{G}$ and $\mathbf{H}$ be two signed compatible genomes. The measures ADJ_{expl}, ADJ_{int} and ADJ_{full} of $\mathbf{G}$ and $\mathbf{H}$ are defined by:*

$$\begin{aligned}
\mathrm{ADJ}_{expl}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{ADJ}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{expl}(\mathbf{G},\mathbf{H})\},\\
\mathrm{ADJ}_{int}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{ADJ}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{int}(\mathbf{G},\mathbf{H})\}, \textit{ and}\\
\mathrm{ADJ}_{full}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{ADJ}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{full}(\mathbf{G},\mathbf{H})\}.
\end{aligned}$$

We have introduced the similarities $\mathrm{ADJ}_{\mathrm{expl}}$ and $\mathrm{ADJ}_{\mathrm{full}}$ in [Angibaud et al., 2007a] and $\mathrm{ADJ}_{\mathrm{int}}$ in [Angibaud et al., 2008a]. Chen *et al.* [Chen et al., 2007b] have proved that computing any of $\mathrm{ADJ}_{\mathrm{expl}}(\mathbf{G},\mathbf{H})$, $\mathrm{ADJ}_{\mathrm{int}}(\mathbf{G},\mathbf{H})$ and $\mathrm{ADJ}_{\mathrm{full}}(\mathbf{G},\mathbf{H})$ is an **NP**-complete problem and is not approximable within ratio $n^{1-\varepsilon}$ even when $\deg(\mathbf{G}) = 1$ and $\deg(\mathbf{H}) = 2$. The problem is also known to **W[1]**-hard. For restricted instances (*i.e.*, full matching and balanced genomes), we have obtained the following approximation results.

Proposition 7.3.7 ([Angibaud et al., 2008b])**.** *Let $\mathbf{G}$ and $\mathbf{H}$ be two balanced genomes with $\deg(\mathbf{G}) = \deg(\mathbf{H}) = k$. If $k = 2$, there exists an algorithm to compute $\mathrm{ADJ}_{full}(\mathbf{G},\mathbf{H})$ with performance ratio 1.1442. If $k = 3$, there exists an algorithm to compute $\mathrm{ADJ}_{full}(\mathbf{G},\mathbf{H})$ with performance ratio $3+\varepsilon$ for any $\varepsilon > 0$. Finally, for general k, there exists an algorithm to compute $\mathrm{ADJ}_{full}(\mathbf{G},\mathbf{H})$ with performance ratio 4.*

It is worth noticing that the above ratio 4 uses 2-intervals (more precisely a result of [Crochemore et al., 2008]) thereby fueling our arguments on the importance of 2-intervals in the design of approximation algorithms.

7.3.5 Common intervals similarity

Common intervals are a natural generalization of adjacencies, as they identify subsets of characters that appear contiguously, but possibly in a different order, in both genomes.

Computing the common intervals similarity between two compatible signed genomes $\mathbf{G}$ and $\mathbf{H}$ reduces to finding a (exemplar, intermediate or full) matching between these two genomes that induces a maximum number of common intervals between the two associated permutations $\pi_{\mathbf{G}}$ and $\pi_{\mathbf{H}}$.

Definition 7.3.8. *Let $\mathbf{G}$ and $\mathbf{H}$ be two signed compatible genomes. The similarities CI_{expl}, CI_{int} and CI_{full} of $\mathbf{G}$ and $\mathbf{H}$ are defined by:*

$$\begin{aligned}
\mathrm{CI}_{expl}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{CI}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{expl}(\mathbf{G},\mathbf{H})\},\\
\mathrm{CI}_{int}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{CI}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{int}(\mathbf{G},\mathbf{H})\}, \textit{ and}\\
\mathrm{CI}_{full}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{CI}(\pi_{\mathbf{G}},\pi_{\mathbf{H}}) : (\pi_{\mathbf{G}},\pi_{\mathbf{H}}) \in \Pi_{full}(\mathbf{G},\mathbf{H})\}.
\end{aligned}$$

Computing the number of common intervals of two permutations on n elements is done in $O(n+k)$ time by [Uno and Yagiura, 2000], where k is the number of common intervals. Notice that [Heber and Stoye, 2001] achieve the same running time for $q \geq 3$ permutations (in this case, n is the total size of the q permutations).

We have introduced the similarities $\mathrm{CI}_{\mathrm{expl}}$ and $\mathrm{CI}_{\mathrm{full}}$ in [Blin et al., 2007b] and $\mathrm{CI}_{\mathrm{int}}$ in [Angibaud et al., 2008b] (common intervals are indeed quite common in comparative genomics). Unfortunately, we are only able to prove inapproximability.

Proposition 7.3.9 ([Blin et al., 2007b]). *Computing any of the two three measures* $\mathrm{CI}_{expl}(\mathbf{G},\mathbf{H})$, $\mathrm{CI}_{int}(\mathbf{G},\mathbf{H})$ *and* $\mathrm{CI}_{full}(\mathbf{G},\mathbf{H})S$ *is an* **APX***-hard problem.*

Notice that the above (negative) result holds even if $\deg(\mathbf{G}) = 1$ and $\deg(\mathbf{G}) = 2$. No positive results are known.

7.3.6 Dissimilarity measures MAD and SAD

The measures to estimate the (dis)similarity between two permutations that we have mentioned so far fall into two categories: they estimate either the distance or the similarity between the two permutations. The two measures considered here (MAD and SAD), both defined by Sankoff and Haque [Sankoff and Haque, 2005], belong to neither category. Unlike distances, their value is never zero, and unlike similarities, their value grows as the dissimilarity of the permutations grows.

Intuitively, MAD and SAD measure how far genes have to move from their initial position in one genome in order to yield the other genome, and this measure focuses either on each gene (MAD) or on all genes altogether (SAD).

For the sake of presentation, we introduce a new notation. For any two permutations $\pi, \sigma \in \mathfrak{S}_n$, we let π^σ stand for the permutation obtained from σ by renaming the elements of π so as to obtain the identity permutation ι, and then renaming the elements of σ accordingly.

***Example* 5** For $\pi = (1\ 3\ 5\ 2\ 4)$ and $\sigma = (5\ 3\ 2\ 1\ 4)$, we obtain $\pi^\sigma = (4\ 2\ 1\ 3\ 5)$ and $\sigma^\pi = (3\ 2\ 4\ 1\ 5)$. ∎

Definition 7.3.10 (Maximum Adjacency Disruption (MAD))**.** *The dissimilarity measure* MAD *between two permutations* $\pi, \sigma \in \mathfrak{S}_n$*, denoted* $\mathrm{MAD}(\pi,\sigma)$*, is defined by:*

$$\mathrm{MAD}(\pi,\sigma) = \max_{1\le i\le n-1} \max\{|\sigma^\pi_i - \sigma^\pi_{i+1}|, |\pi^\sigma_i - \pi^\sigma_{i+1}|\}.$$

Intuitively, the MAD dissimilarity measure of two permutations π and σ is the largest gap between two consecutive elements in σ^π and π^σ. Notice that $\mathrm{MAD}(\pi,\pi) = 1$ for all $\pi \in \mathfrak{S}_n$.

***Example* 6** Pursuing Example 5, *i.e.*, $\pi = (1\ 3\ 5\ 2\ 4)$ and $\sigma = (5\ 3\ 2\ 1\ 4)$, we get $\mathrm{MAD}(\pi,\sigma) = \max\{\max\{1,2\}, \max\{2,1\}, \max\{3,2\}, \max\{4,2\}\} = 4$. ∎

Definition 7.3.11. *Let* $\mathbf{G}$ *and* $\mathbf{H}$ *be two compatible genomes. The dissimilarity measures* MAD_{expl}, MAD_{int} *and* MAD_{full} *of* $\mathbf{G}$ *and* $\mathbf{H}$ *are defined by:*

$$\begin{aligned}
\mathrm{MAD}_{expl}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{MAD}(\pi_\mathbf{G},\pi_\mathbf{H}) : (\pi_\mathbf{G},\pi_\mathbf{H}) \in \Pi_{expl}(S,T)\},\\
\mathrm{MAD}_{int}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{MAD}(\pi_\mathbf{G},\pi_\mathbf{H}) : (\pi_\mathbf{G},\pi_\mathbf{H}) \in \Pi_{int}(S,T)\},\ \textit{et}\\
\mathrm{MAD}_{full}(\mathbf{G},\mathbf{H}) &= \min\{\mathrm{MAD}(\pi_\mathbf{G},\pi_\mathbf{H}) : (\pi_\mathbf{G},\pi_\mathbf{H}) \in \Pi_{full}(S,T)\}.
\end{aligned}$$

We have obtained the following result.

Proposition 7.3.12 ([Blin et al., 2007b]). *Unless* $\mathbf{P} = \mathbf{NP}$*, there does not exist an approximation algorithm with performance guarantee* $2-\varepsilon$*, for any* $\varepsilon > 0$*, to compute the dissimilarity measures* $\mathrm{MAD}_{expl}(\mathbf{G},\mathbf{H})$, $\mathrm{MAD}_{int}(\mathbf{G},\mathbf{H})$ *and* $\mathrm{MAD}_{full}(\mathbf{G},\mathbf{H})$.

The above proposition holds even if $\deg(\mathbf{G}) = 1$ and $\deg(\mathbf{H}) = 9$. It is worth mentioning that $2-\varepsilon$ is not the best bound. Indeed, for the sake of proof simplicity, the proof of Proposition 7.3.12, as presented in [Blin et al., 2007b], does not use the PCP theorem. However, resorting to the PCP theorem (more precisely, resorting to the non-approximation of a restriction of the MAX 3-SAT problem), we can show that there exists a constant $c > 2$ such that there does not exist an approximation algorithm with performance guarantee c to compute the dissimilarity measures $\mathrm{MAD}_{expl}(\mathbf{G},\mathbf{H})$, $\mathrm{MAD}_{int}(\mathbf{G},\mathbf{H})$ and $\mathrm{MAD}_{full}(\mathbf{G},\mathbf{H})$ (the exact value

of c has not been precisely computed). No algorithmic positive result to compute the MAD dissimilarity measure is known so far, but we conjecture Proposition 7.3.12 to be far from being tight.

We now turn to considering the SAD dissimilarity measure that takes into account all differences between consecutive elements.

Definition 7.3.13 (Summed Adjacency Disruption (SAD)). *The* SAD *dissimilarity measure of two permutations* $\pi, \sigma \in \mathfrak{S}_n$*, denoted* $\mathrm{SAD}(\pi, \sigma)$*, is defined by:*

$$\mathrm{SAD}(\pi, \sigma) = \sum_{i=1}^{n-1} \left(|\sigma_i^\pi - \sigma_{i+1}^\pi| + |\pi_i^\sigma - \pi_{i+1}^\sigma| \right).$$

Intuitively, the SAD dissimilarity measure is the sum of all gaps between two consecutive elements in σ^π and π^σ. Notice that $\mathrm{SAD}(\pi, \pi) = 2(n-1)$ for all $\pi \in \mathfrak{S}_n$.

Example 7 For $\pi = (1\ 3\ 5\ 2\ 4)$ and $\sigma = (5\ 3\ 2\ 1\ 4)$, we obtain $\mathrm{SAD}(\pi, \sigma) = (1+2)+(2+1)+(3+2)+(4+2) = 17$. ∎

Definition 7.3.14. *Let* $\mathbf{G}$ *and* $\mathbf{H}$ *be two compatible genomes. The dissimilarity measures* SAD_{expl}*,* SAD_{int} *and* SAD_{full} *of* $\mathbf{G}$ *and* $\mathbf{H}$ *are defined by:*

$$\begin{aligned}
\mathrm{SAD}_{expl}(\mathbf{G}, \mathbf{H}) &= \min\{\mathrm{SAD}(\pi_\mathbf{G}, \pi_\mathbf{H}) : (\pi_\mathbf{G}, \pi_\mathbf{H}) \in \Pi_{expl}(S, T)\}, \\
\mathrm{SAD}_{int}(\mathbf{G}, \mathbf{H}) &= \min\{\mathrm{SAD}(\pi_\mathbf{G}, \pi_\mathbf{H}) : (\pi_\mathbf{G}, \pi_\mathbf{H}) \in \Pi_{int}(S, T)\}, \text{ et} \\
\mathrm{SAD}_{full}(\mathbf{G}, \mathbf{H}) &= \min\{\mathrm{SAD}(\pi_\mathbf{G}, \pi_\mathbf{H}) : (\pi_\mathbf{G}, \pi_\mathbf{H}) \in \Pi_{full}(S, T)\}.
\end{aligned}$$

Not surprisingly, computing the SAD dissimilarity measure turns out the be more difficult than computing the MAD dissimilarity measure (as long as we cannot prove stronger inapproximability for the latter).

Proposition 7.3.15 ([Blin et al., 2007b]). *Unless* $\mathbf{P} = \mathbf{NP}$*, there exists a constant* $c > 0$ *such that no approximation algorithm with performance guarantee* $c \log n$ *is achievable to compute the dissimilarity measures* $\mathrm{SAD}_{expl}(\mathbf{G}, \mathbf{H})$*,* $\mathrm{SAD}_{int}(\mathbf{G}, \mathbf{H})$ *and* $\mathrm{SAD}_{full}(\mathbf{G}, \mathbf{H})$.

Once again, no positive algorithmic result to compute the dissimilarity measure SAD is known so far. Whereas Sankoff and Haque [Sankoff and Haque, 2005] claim that MAD and SAD are relevant for comparative genomics, algorithmic cannot help much here.

How related are the different measures discussed in Section 7.3? Do they answer different questions? Do they answer different parts of a research question? There are some of many important questions left open. Our woks [Angibaud et al., 2008a, 2007a,b] only provide partial answers.
At a more general level, our contributions are only algorithmic and actually we didn't introduce any new measure but express in our terms measures that are used by the comparative genomics community. We might regret this abundance of measures to the detriment of comparative analyses. As an example, MAD and SAD turn out to be very complicated measures (from an algorithmic point of view) but, to the best of our knowledge, no evidence has proved or disprove the benefit of such tortuous measures.

7.4 Exact algorithms and heuristics

7.4.1 Boolean linear programming

We have considered in [Angibaud et al., 2006], [Angibaud et al., 2007a] and [Angibaud et al., 2008a] exact algorithms for computing various genome rearrangement distances. It is worth noticing that the rational for considering such an approach was not to propose efficient general algorithms but to compute for a reference dataset a bunch of exact results new heuristic approaches can confront with. Our approach was by pseudo-boolean programming (linear integer programming where all variables are restricted to take values of either 0 or 1). All our experiments were conducted with Minisat+ [Eén and Sörensson., 2006] and ILOG CPLEX (`http://www.ilog.com/products/cplex`). This part was the subject of the PhD thesis of Annelyse Thevenin (defended November 2009).

To illustrate our approach, we present in Figure 7.1 a pseudo-boolean program to compute the minimum number of breakpoints between two genomes under the full matching viewpoint [Angibaud et al., 2007a]. We refer the reader to [Angibaud et al., 2006], [Angibaud et al., 2007a] and [Angibaud et al., 2008a] for a thorough discussion on pseudo-boolean-programming for genome comparison. In particular, we investigated in these papers the impact of the choice of the matching (exemplar, covering or maximum) on a γ-proteobacteria dataset [Lerat et al., 2003].

7.4.2 Heuristic approaches

We focus in this subsection on heuristics to deal with the aforementioned problems, and the motivation for our choice to present heuristics rather than exact algorithms (or rather than both heuristics and exact algorithms) relies on their universality. These heuristics are identical for all distances and need only minor changes to go from one model to another one.

The three heuristics presented here use the notion of a *longest common substring, up to a complete reversal* and are all based on the following easy idea. Assuming temporarily that one aims at finding a full matching between **G** and **H** which intuitively preserves the most conserved regions between the two strings, an easy way to have such a matching is given by Algorithm ILCS where we assume that each longest common substring found on **G** and **H** is identified by *one* precise occurrence on each of **G** and **H**. Figure 7.2 shows an example.

Algorithm 1: ILCS Heuristic (full matching)

Data: Two genomes **G** and **H**.
Result: A matching between **G** and **H**.
1 Compute a longest common substring L of **G** and **H**, up to a complete reversal, exclusively made of unmatched characters from **G** and **H**.
2 Match the characters of **G** and **H** belonging to the occurrences of L according to their positions in L.
3 Iterate the process until all possible characters have been matched.
4 Remove all unmatched characters.
return *The required distance on the resulting permutations*

As far as we know, this idea appeared in [Tichy, 1984] and was often used since. In [Angibaud et al., 2007b] and [Angibaud et al., 2008a] we proceeded to a large number of time consuming distance computations, and noticed that even small changes in the ILCS algorithm might improve both the execution time, the quality of the result and its applicability to various models. The Algorithm IILCS we proposed in [Angibaud et al., 2007b] is such a variant of ILCS where the removal of characters which cannot be matched is done *before*

starting a new iteration.

Algorithm 2: IILCS Heuristic (exemplar/intermediate/full matching)

Data: Two genomes **G** and **H**.
Result: A matching between **G** and **H**.
1 Compute a longest common substring L of **G** and **H**, up to a complete reversal, exclusively made of unmatched characters from **G** and **H**.
2 Match (all or part of) the characters of **G** and **H** belonging to the occurrences of L according to their positions in L, so as to fit the exemplar/intermediate/full model constraints.
3 Iterate the process until all possible characters have been matched.
4 Remove all unmatched characters.
return *The required distance on the resulting permutations*

This new heuristic allows to obtain in step **2** one or several pairs of matched characters in each gene family, according to the model, and to discard in step 3 all characters which become useless. Behind the flexibility introduced by this variant of ILCS in regard to the model, an improvement of the results may also be expected as IILCS better takes into account the final goal of matching characters, which is to identify as many conserved regions as possible in the resulting pruning, and not in **G** and **H**. Indeed, the resulting pruning has consecutive characters which were not consecutive in the initial strings, and thus has conserved regions which were possibly not conserved in the initial strings. The early removal of characters by IILCS allows non-neighboring characters on **G** and **H** to become neighbors at the end of some iteration, if the characters between them are not matched. New longest common substrings may then be formed in this way, thus improving the identification of common regions in the final pruning.

The argument these heuristics relies on is that long common substrings are strongly conserved regions that strongly affect the values of all measures, either distances or similarities. Such an argument is supported by the good performances of these heuristics (see below), but cannot be invoked when the longest common substrings are short, *i.e.*, not exceeding some given length ℓ. Consequently, it could be a reasonable idea to stop the execution of the IILCS heuristic when the threshold ℓ is reached for the length of the longest common substring, and then to apply some exact (and thus exponential) algorithm to optimally match the remaining characters according to the problem P to solve. Problem P is defined by the measure to compute, and the model to use. This idea yields the hybrid method we have developed in [Angibaud et al., 2007b].

The IILCS and hybrid heuristics were systematically evaluated together with ILCS in [Angibaud et al., 2007b] and [Angibaud et al., 2008a] on several problems and data sets for which exact results are known (these results are actually ours, see 7.4.1). These evaluations show that our heuristics perform very well on experimental data.

Program `Max-Matching-Breakpoint`

Objective :

`Maximize` $\sum_{0\le i<n_{\mathbf{G}}} \sum_{i<j\le n_{\mathbf{G}}} \sum_{0\le k<n_{\mathbf{H}}} \sum_{k<l\le n_{\mathbf{H}}} d(i,j,k,l)$

Constraints :

(`C.01`) $\forall\, 1 \le i \le n_{\mathbf{G}}, \sum_{\substack{1\le k\le n_{\mathbf{H}} \\ |\mathbf{G}[i]|=|\mathbf{H}[k]|}} a(i,k) = b_{\mathbf{G}}(i)$

$\forall\, 1 \le k \le n_{\mathbf{H}}, \sum_{\substack{1\le i\le n_{\mathbf{G}} \\ |\mathbf{G}[i]|=|\mathbf{H}[k]|}} a(i,k) = b_{\mathbf{H}}(k)$

(`C.02`) $\forall\, X \in \{\mathbf{G},\mathbf{H}\}, \forall\, g \in \mathcal{A}, \sum_{\substack{1\le i\le n_X \\ |X[i]|=|g|}} b_X(i) = \min\{|\mathbf{G}|_g, |\mathbf{H}|_g\}$

(`C.03`) $\forall\, X \in \{\mathbf{G},\mathbf{H}\}, \forall 1 \le i \le j-1 < n_X, c_X(i,j) + \sum_{i<p<j} b_X(p) \ge 1$

(`C.04`) $\forall\, X \in \{\mathbf{G},\mathbf{H}\}, \forall\, 1 \le i < p < j \le n_X, c_X(i,j) + b_X(p) \le 1$

(`C.05`) $\forall\, 1 \le i < j \le n_{\mathbf{G}}, \forall\, 1 \le k < l \le n_{\mathbf{H}}$,
such that $\mathbf{G}[i] = \mathbf{H}[k]$ and $\mathbf{G}[j] = \mathbf{H}[l]$,
$a(i,k) + a(j,l) + c_{\mathbf{G}}(i,j) + c_{\mathbf{H}}(k,l) - d(i,j,k,l) \le 3$

(`C.06`) $\forall\, 1 \le i < j \le n_{\mathbf{G}}, \forall\, 1 \le k < l \le n_{\mathbf{H}}$,
such that $\mathbf{G}[i] = \mathbf{H}[k]$ and $\mathbf{G}[j] = \mathbf{H}[l]$,
$a(i,k) - d(i,j,k,l) \ge 0$
$a(j,l) - d(i,j,k,l) \ge 0$
$c_{\mathbf{G}}(i,j) - d(i,j,k,l) \ge 0$
$c_{\mathbf{H}}(k,l) - d(i,j,k,l) \ge 0$

(`C.07`) $\forall\, 1 \le i < j \le n_{\mathbf{G}}, \forall\, 1 \le k < l \le n_{\mathbf{H}}$,
such that $\mathbf{G}[i] = -\mathbf{H}[l]$ `and` $\mathbf{G}[j] = -\mathbf{H}[k]$,
$a(i,l) + a(j,k) + c_{\mathbf{G}}(i,j) + c_{\mathbf{H}}(k,l) - d(i,j,k,l) \le 3$

(`C.08`) $\forall\, 1 \le i < j \le n_{\mathbf{G}},\ \forall\, 1 \le k < l \le n_{\mathbf{H}}$,
such that $\mathbf{G}[i] = -\mathbf{H}[l]$ `and` $\mathbf{G}[j] = -\mathbf{H}[k]$,
$a(i,l) - d(i,j,k,l) \ge 0$
$a(j,k) - d(i,j,k,l) \ge 0$
$c_{\mathbf{G}}(i,j) - d(i,j,k,l) \ge 0$
$c_{\mathbf{H}}(k,l) - d(i,j,k,l) \ge 0$

(`C.09`) $\forall\, 1 \le i < j \le n_{\mathbf{G}},\ \forall\, 1 \le k < l \le n_{\mathbf{H}}$,
such that $\{|\mathbf{G}[i]|, |\mathbf{G}[j]|\} \ne \{|\mathbf{H}[k]|, |\mathbf{H}[l]|\}$ `ou` $\mathbf{G}[i] - \mathbf{G}[j] \ne \mathbf{H}[k] - \mathbf{H}[l]$,
$d(i,j,k,l) = 0$

(`C.10`) $\forall\, 1 \le i < j \le n_{\mathbf{G}}, \sum_{1\le k<n_{\mathbf{H}}} \sum_{k<l\le n_{\mathbf{H}}} d(i,j,k,l) \le 1$

Domains :

$\forall\, X \in \{\mathbf{G},\mathbf{H}\}, \forall\, 1 \le i < j \le n_{\mathbf{G}}, \forall\, 1 \le k < l \le n_{\mathbf{H}}$,
$a(i,k),\ b_X(i),\ c_X(i,k),\ d(i,j,k,l) \in \{0,1\}$

Figure 7.1: Program `Max-Matching-Breakpoint` to compute $\mathrm{ADJ}_{\mathrm{full}}(\mathbf{G},\mathbf{H})$, where $n_{\mathbf{G}}$ and $n_{\mathbf{H}}$ denote the size of $\mathbf{G}$ and $\mathbf{H}$, respectively.

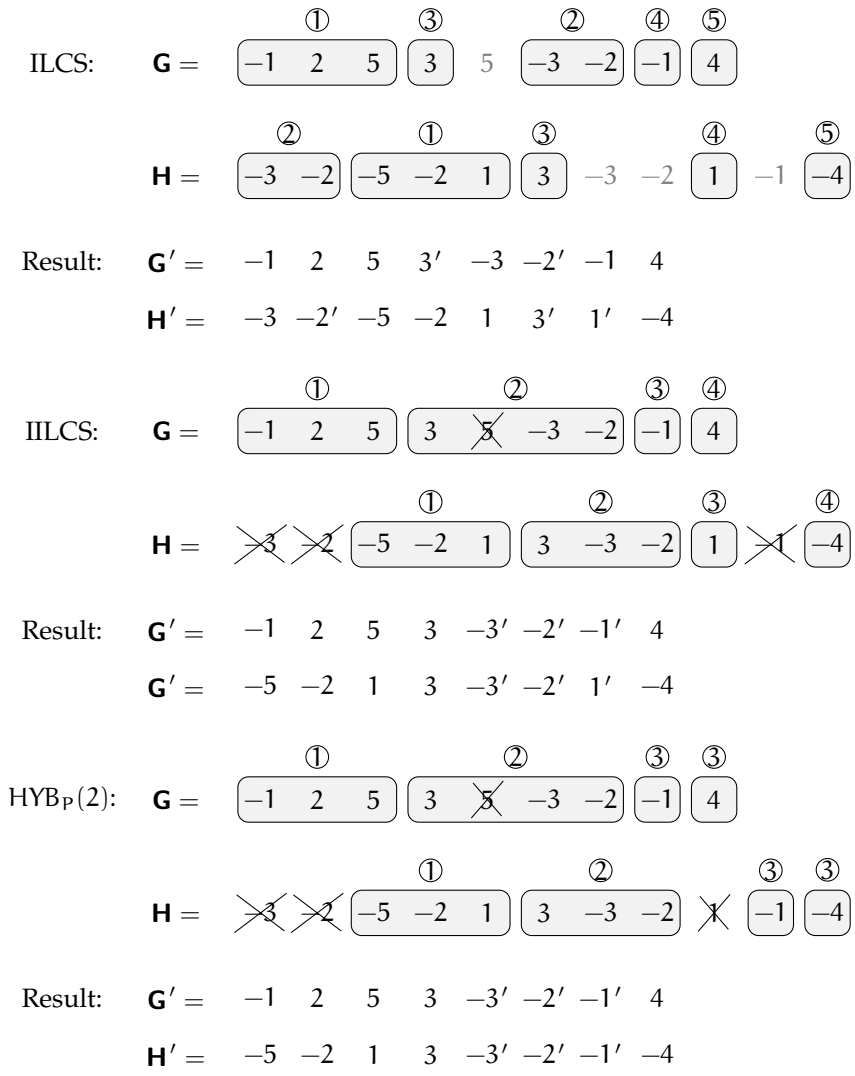


Figure 7.2: Execution and results of the three heuristics, seeking for a full matching, on the strings $\mathbf{G} = -1\,2\,5\,3\,5\,-3\,-2\,-1\,4$ and $\mathbf{H} = -3\,-2\,-5\,-2\,1\,3\,-3\,-2\,1\,-1\,-4$. As an example, the problem P in the HYB heuristic asks to compute the conserved intervals similarity. The circled numbers indicate in which order the LCS were identified, except for the ③ in the HYB heuristic which signifies that the matchings were decided simultaneously by the exact algorithm evoked in the last step of the HYB heuristic.

8 Exemplar common subsequences

8.1 Introduction

This short chapter is devoted to presenting some results on exemplar longest common subsequences.

In the genome rearrangement domain, gene duplication is rarely considered as, as we have seen, it usually makes the problem at hand harder (see Chapter 7 for a patent illustration). Sankoff [Sankoff, 1999] proposed the so-called exemplar model, which consists in searching, for each family of duplicated genes, an exemplar representative in each genome. In biological terms, the exemplar gene may correspond to the original copy of the gene, which later originated all other copies. Following the parsimony principle, the choices of exemplars should be made so as to minimize the reversal distance between the two simpler versions of both genomes, composed only by the exemplar genes. An alternative to the exemplar model is the multigene family model, which consists in maximizing the number of paired genes among a family. Again, the gene pairs should be chosen so as to minimize the reversal distance between the genomes. Both exemplar and multigene models were proven to lead to **NP**-hard problems [Bryant, 2000; Blin et al., 2004].

To compare two sequences, we have proposed in [Bonizzoni et al., 2007] to study a similarity measure that takes into account the concept of exemplar genes. Observe that a repetition-free LCS can be seen as the edit distance between two sequences where only deletions are allowed and, furthermore, for each family with k duplicated genes, at least k1 of them must be deleted [Sadique Adi et al., 2008]. At a more general level, we considered both mandatory and optional letters and the measure we have proposed is the length of a constrained common subsequence (LCS) (see [Bergroth et al., 2000; Crochemore et al., 2007]) subject to two constraints: (i) the common LCS is required to contains exactly or at least one occurrence of each mandatory letter, and (ii) the common LCS is required to contains at most one occurrence of each optional letter (this constraint may be relaxed). Additional restrictions on optional letters allow us for varying restriction on the solution we are looking for. Notice that a complete treatment of pure exemplar LCS has been recently proposed in [Sadique Adi et al., 2008].

8.2 Definitions

We define four variants of the EXEMPLAR-LCS problem we are interested in. These different variants consider different situations: each mandatory letter is required to occur exactly or at most once in the common LCS, and the number of occurrences of each optional letter in the common LCS may be upper-bounded.

Exemplar-LCS-i

• **Input** : Two strings u and v over alphabet $\mathcal{A} = \mathcal{A}_o \dot{\cup} \mathcal{A}_m$, where $\mathcal{A}_o$ is the set of optional letters and $\mathcal{A}_m$ is the set of mandatory letters.
• **Solution** : A common subsequence w of u and v such that (depending on i):

$\mathbf{i = 1}$: w contains exactly one occurrence of each letter in $\mathcal{A}_m$ and at most one occurrence of each letter in $\mathcal{A}_o$,

$\mathbf{i = 2}$: w contains at least one occurrence of each letter in $\mathcal{A}_m$ and at most one occurrence of each letter in $\mathcal{A}_o$,

$\mathbf{i = 3}$: w contains exactly one occurrence of each letter in $\mathcal{A}_m$,

$\mathbf{i = 4}$: w contains at least one occurrence of each letter in $\mathcal{A}_m$.

• **Measure** : The length of the common subsequence, *i.e.*, $|w|$.

Notice, that, as we shall see soon, whereas the classical LCS problem is well-known to be polynomial-time solvable for two stings [Bergroth et al., 2000; Crochemore et al., 2007], adding various constraints on mandatory and optional letters result in much harder problems (not a big surprise).

8.3 Key results

We summarize in this section the results we have obtained for the EXEMPLAR-LCS-i, $1 \leq i \leq 4$, problem.

Proposition 8.3.1 ([Bonizzoni et al., 2007]). *Both the* EXEMPLAR-LCS-1 *problem and the* EXEMPLAR-LCS-2 *problem are* **APX***-hard.*

It is worth noticing that Proposition 8.3.1 holds even if each letter occurs at most twice in both u and v. Observe that, even if these two problems are very similar in their definition, two distinct proofs were needed.

Strongly related to the the EXEMPLAR-LCS-i, $1 \leq i \leq 4$, problems are the one of determining whether any feasible solution does exist. Let us focus on the EXEMPLAR-LCS-4 problem, *i.e.*, find for a common subsequence w that contains at least one occurrence of each mandatory letter. Clearly, optimality aside, the existence of a solution for the EXEMPLAR-LCS-4 problem implies the existence of a solution for all EXEMPLAR-LCS-i problems.

Proposition 8.3.2 ([Bonizzoni et al., 2007]). *Let* (u, v) *be an instance of the* EXEMPLAR-LCS-i *problem,* $1 \leq i \leq 4$. *If* $|u|_a + |v|_a \leq 3$ *for each letter* $a \in \mathcal{A}$, *then there exists a polynomial-time algorithm to decide whether there exists a feasible solution.*

Proposition 8.3.3 ([Bonizzoni et al., 2007]). *Let* (u, v) *be an instance of the* EXEMPLAR-LCS-i *problem,* $1 \leq i \leq 4$. *If* $|u|_a + |v|_a \leq 3$ *for each letter* $a \in \mathcal{A}$, *deciding whether there exists any feasible solution is* **NP***-complete.*

This latter result have a definitive consequence on the approximability of the EXEMPLAR-LCS-i problem when each mandatory letter occurs at most 3 times in each input string as it rules out any polynomial-time approximation algorithm.

In the light of the above negative results, we have considered in [Bonizzoni et al., 2007] parameterized issues of the EXEMPLAR-LCS-3 and EXEMPLAR-LCS-4 problems when the parameter is the number of mandatory letters, *i.e.*, $|\mathcal{A}_m|$.

Proposition 8.3.4 ([Bonizzoni et al., 2007]). *Both the* EXEMPLAR-LCS-3 *and the* EXEMPLAR-LCS-4 *problems are solvable in* $O(m2^m n^2)$ *time, where* $m = |\mathcal{A}_m|$ *and* $n = \max\{|u|, |v|\}$.

Our fixed-parameter algorithm for the EXEMPLAR-LCS-3 problem has been implemented and tested on randomly generated strings. Whereas the running time is acceptable, the space complexity (that grows exponentially in the size of $\mathcal{A}_m$) makes the algorithm not practical for $|\mathcal{A}_m| \geq 20$.

Part IV

Additional topics

Introduction

This part is devoted to presenting two additional contributions to computational molecular biology. The first one is concerned with selenocysteine-like insertion and more precisely the problem of computing an mRNA sequence of maximum codon-wise similarity to a given mRNA (and hence, to a given protein) that additionally satisfies some structure constraints. We do not write secondary structure, *i.e.*, pseudoknot-free secondary structures, intentionally as we shall consider pseudoknoted structures. The second problem is devoted to covering strings by substrings. Our initial motivation came from a paper by Bodlaender *et al.* [Bodlaender et al., 1995], who described an application for this problem in the context of protein folding.

This part is organized as follows. Chapter 9 is devoted generalized selenocysteine insertion and Chapter 10 with covering strings by substrings. The two following chapters are independent and as self-contained as possible.

9 Selenocysteine-like insertion

Contents

9.1 Introduction

Perhaps the most significant process in molecular biology known today is the transformation of genetic information encoded in DNA into proteins. In this process, segments of DNA are transcribed into messenger RNA (mRNA), which in turn serve as blueprints for manufacturing proteins. This protein blueprint is provided by triplets of nucleotides known as codons, which compose the mRNA nucleotide sequence, where each codon encodes a specific amino acid. An mRNA is thus translated into a protein by reading each of its codons in sequential fashion, and creating a chain of amino acids which forms the target protein. Recently, biologists found out that according to the folding structure of an mRNA molecule, a certain codon might encode for different amino acids. This folding structure is captured in many ways, in what is called the mRNA secondary structure, the set of all hydrogen bonds, or base pairings, formed by the molecule's nucleotides.

In [Backofen et al., 2002], Backofen *et al.* introduced the problem of computing an mRNA sequence of maximum codon-wise similarity to a given mRNA (and consequently, to a given protein) that additionally satisfies some secondary structure constraints, the so-called mRNA Structure Optimization (MRSO) problem. The initial motivation of the MRSO problem is concerned with selenocysteine insertion, *i.e.* generating new amino acid sequences containing selenocysteine. This rare amino acid was discovered as the 21-st amino acid [Böch et al., 1991], giving another clue to the complexity and flexibility of the mRNA translation mechanism. Selenocysteine is encoded by the UGA codon, which is usually a stop codon encoding the end of translation. It has been shown [Böch et al., 1991] that in case of selenocysteine, termination of translation is inhibited in the presence of a sequence of nucleotides, the SECIS element, which forms a hairpin-like structure in the 3′-region after the UGA codon (see Figure 9.1). It is even argued in [Backofen et al., 2002] that modifying existing proteins by incorporating selenocysteine instead of a catalytic cysteine is an important problem for catalytic activity enhancement and X-ray crystallography.

Selenocysteine insertion is concerned with a restricted type of secondary structure, *i.e.*, a secondary

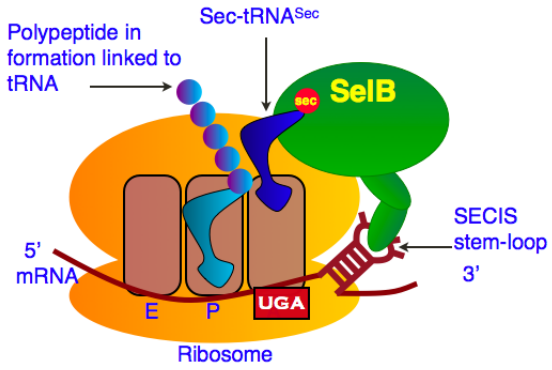


Figure 9.1: The translation of UGA into selenocysteine. Termination of translation is inhibited in the presence of the SECIS element.

structure without pseudoknots, and for this type of structure the linear-time algorithm presented in [Backofen et al., 2002] provides an optimal solution. However, it is reasonable to assume that the discovery of selenocysteine will lead to the discovery of several other amino acids of similar kind, some of which are likely to require more complex secondary structures. Even today, similar problems occur in programmed frameshifts which allow to encode two different amino acid sequences in one mRNA sequence [Jacks et al., 1988; Jacks and Varmus, 1985]. This motivates the investigation of the MRSO problem for more elaborate secondary structures (actually, this issues have been suggested in [Backofen et al., 2002; Bongartz, 2004]).

9.2 Preliminaries

An mRNA molecule is viewed as a string over the alphabet $\mathcal{A} = \{A, C, G, U\}$, where $\mathcal{A}$ represents the four different types of nucleotides in the molecule. The pairs $\{A, U\}$, $\{G, C\}$, and $\{G, U\}$ are known as *complementary nucleotide pairs*. Hydrogen bonds can only be formed between complementary nucleotides in an mRNA folding. A *codon* of an mRNA sequence is a segment of three nucleotides, *i.e.*, a string in $\mathcal{A}^3$. Thus, an mRNA sequence $S = s_1 s_2 \dots s_{3n}$ is a concatenation of n consecutive codons, where the i-th codon of S is $s_{3i-2} s_{3i-1} s_{3i}$.

Given a *source* mRNA sequence $S = s_1 s_2 \dots s_{3n}$, we consider the problem of evaluating the codon-wise similarity between S and another *target* mRNA sequence $T = t_1 t_2 \dots t_{3n}$. For this, we are provided with a set of n functions, $\mathcal{F} = f_1, f_2 \dots, f_n$, called *similarity functions* of S, such that for all i, $1 \leq i \leq n$, each function f_i is of the form $f_i : \Sigma^3 \to \mathbb{Q}$. Thus, f_i assigns a value to the i-th codon of T according to its level of similarity in comparison with the i-th codon of S. The total level of similarity between S and T is then given by $\sum_{i=1}^{n} f_i(t_{3i-2} t_{3i-1} t_{3i})$. Note that given a set of similarity functions $\mathcal{F} = f_1, f_2, \dots, f_n$ for S, one does not need to know anything else about S in order to compute the similarity score of S and T.

The *structure constraints* $\Gamma \subseteq \{\{i, j\} : 1 \leq i < j \leq 3n\}$ for a target mRNA sequence T of length $3n$, are pairings between distinct integers in $\{1, 2, \dots, 3n\}$. These represent necessary hydrogen bonds in the folding of T. It is assumed that each nucleotide can pair with at most one other nucleotide in any folding, hence each integer appears in at most one pair in Γ. Furthermore, there are no pairs of the form $\{i, i+1\}$ or $\{i, i+2\}$ in Γ, for all i, $1 \leq i \leq 3n-2$. Given a set of structure constraints $\Gamma \subseteq \{\{i, j\} : 1 \leq i < j \leq 3n\}$, and an arbitrary target mRNA sequence $T = t_1 t_2 \dots t_{3n}$, we say that nucleotides t_i and t_j in T are *compatible* with respect to Γ, if either $\{t_i, t_j\}$ is a complementary nucleotide pair or $\{i, j\} \notin \Gamma$. The entire sequence T is compatible with respect to Γ, if all pairs of nucleotides in T are compatible with respect to Γ. We are now in position to formally define the mRNA Structure Optimization (MRSO) problem we are interested in (see [Backofen

et al., 2002]).

MRSO

- **Input** : A set $\mathcal{F}$ of n similarity functions for a source mRNA sequence of length $3n$, and and a set of structure constraints $\Gamma \subseteq \{\{i,j\} : 1 \leq i < j \leq 3n\}$. .
- **Solution** : A target mRNA sequence which is compatible with respect to Γ.
- **Measure** : The similarity score of the target mRNA sequence with respect to $\mathcal{F}$.

It will convenient to formalize the MRSO problem in a slightly different manner using graph-theoretic concepts, as we shall consider Γ as a linear graph (see Definition 2.2.1 page 18) with $3n$ vertices which has a maximum degree of one. However, since we are really interested in codon-wise similarity, we use a more suitable representation of Γ:

Definition 9.2.1 (Implied structure graph [Backofen et al., 2002]). *Let $\Gamma \subseteq \{\{i,j\} : 1 \leq i < j \leq 3n\}$ be a set of structure constraints for a target mRNA sequence of length $3n$. The implied structure graph of Γ is the linear graph G_Γ defined by:*

$$\mathbf{V}(G_\Gamma) = \{1, 2, \ldots, n\}, \textit{ and}$$
$$\mathbf{E}(G_\Gamma) = \{\{i,j\} : \exists \{x,y\} \in \Gamma : x \in \{3i-2, 3i-1, 3i\} \wedge y \in \{3j-2, 3j-1, 3j\}\}.$$

In this way, a vertex i in $\mathbf{V}(G_\Gamma)$ corresponds to the i-th codon of the target mRNA sequence, and $i, j \in \mathbf{V}(G_\Gamma)$ are connected in $\mathbf{E}(G_\Gamma)$ if there are any structure constraints in Γ between the i-th and j-th codons of the sequence. Note that there can be at most three structure constraints between any pair of codons, therefore G_Γ has maximum degree of three, *i.e.*, it is a *subcubic* graph. Also note that, while this representation may seem lossy, Γ can be retained from G_Γ by adding up to three labels for each edge in $\mathbf{E}(G_\Gamma)$.

Given a subset of vertices $V \subseteq \mathbf{V}(G_\Gamma)$, we let $G_\Gamma[V]$ denote the subgraph of G_Γ *induced* by V, *i.e.*, the subgraph with V as its vertex set, and $\mathbf{E}(G_\Gamma) \cap (V \times V)$ as its edge set. Similarly, given a subset of edges $E \subseteq \mathbf{E}(G_\Gamma)$, $G_\Gamma[E]$ denotes the subgraph of G_Γ with vertex set $\{i \mid \{i,j\} \in E\}$ and edge set E. Also, we use $G_\Gamma[i, \ldots, j]$ to denote the subgraph of G_Γ induced by $\{i, \ldots, j\} \subseteq \mathbf{V}(G_\Gamma)$. Two edges $\{i,j\}$ and $\{i',j'\}$ *cross* in G_Γ if either $i < i' < j < j'$ or $i' < i < j' < j$. Note that two crossing edges might not cross under a different ordering of $\mathbf{V}(G_\Gamma)$. If there exists an ordering of $\mathbf{V}(G_\Gamma)$ which introduces no edge crossings then G_Γ is *outerplanar*. Recall that if G_Γ is outerplanar, the MRSO problem is solvable $O(n)$ time [Backofen et al., 2002].

A *codon assignment* for G_Γ is a mapping from some $V \subseteq \mathbf{V}(G_\Gamma)$ to Σ^3. An assignment for a pair of vertices $i, j \in \mathbf{V}(G_\Gamma)$, $i \rightarrow t_{3i-2}t_{3i-1}t_{3i}$ and $j \rightarrow t_{3j-2}t_{3j-1}t_{3j}$, is compatible with respect to G_Γ, if either $\{i,j\} \notin \mathbf{E}(G_\Gamma)$ or $t_{i'}$ and $t_{j'}$ are complementary nucleotides for any $\{i',j'\} \in \Gamma \cap \{3i-2, 3i-1, 3i\} \times \{3j-2, 3j-1, 3j\}$. More generally, an assignment $\phi : V \rightarrow \Sigma^3$ for some $V \subseteq \mathbf{V}(G_\Gamma)$ is compatible with respect to G_Γ, if for any $i, j \in V$, the assignment $i \rightarrow \phi(i)$ and $j \rightarrow \phi(j)$ is compatible with respect to G_Γ. Henceforth, we consider instances for the MRSO problem of the form $(G_\Gamma, \mathcal{F})$. Our goal in this setting is to find an assignment $\phi : \mathbf{V}(G_\Gamma) \rightarrow \Sigma^3$ (*i.e.*, a target mRNA sequence $T = \phi(1)\phi(2)\ldots\phi(n)$), which is compatible with G_Γ, and which maximizes $\sum_{i=1}^{n} f_i(\phi(i))$.

For the MRSO problem, it has been shown in [Backofen et al., 2002] that there exists a linear-time algorithm if the considered secondary structure corresponds to an outerplanar graph (as it is the case for selenocysteine insertion). In this sequel, we refer to this algorithm as A_{OP}. For the general case, the problem was proved to be **NP**-complete [Backofen et al., 2002], and Bongartz showed that in fact the problem is **APX**-hard [Bongartz, 2004]. An algorithm for approximating the MRSO problem within ratio 2 is given in [Backofen et al., 2002]. A slightly slower but somewhat simpler 4-approximation algorithm is given in [Bongartz, 2004]. We mention also that an extension of the MRSO problem, where insertions and deletions are allowed in the amino acid sequence is presented in [Backofen and Busch, 2004].

9.3 Key results

We shall be concerned with two natural parameters for the MRSO problem. These are the number of crossings edges in G_Γ, and the number of degree three vertices in G_Γ, denoted $p(G_\Gamma)$ and $q(G_\Gamma)$, respectively.

Our initial interest in parameters $p(G_\Gamma)$ and $q(G_\Gamma)$ stems from the fact that we strongly believe them to be small for most practical applications. Consider parameter $p(G_\Gamma)$, the number of edge crossings in G_Γ (this parameter was previously suggested in [Bongartz, 2004]). Indeed, almost all currently known mRNAs have secondary structures which induce outerplanar formations, *i.e.*, formations with no edge crossings. Furthermore, many secondary structure prediction algorithms restrict their search space to structures with bounded edge crossings, since prediction with unbounded edge crossings usually becomes **NP**-hard, and is anyhow assumed unnatural (see for instance [Akutsu, 2000]). As for parameter $q(G_\Gamma)$, the number of degree three vertices, recall that a vertex of degree three in G_Γ represents a codon with three nucleotides, each pairing with complementary nucleotides in three different codons. Although this situation can occur in a folding of an mRNA molecule, it can be expected to be quite rare due to the geometric constraints imposed on any such folding. Also, pairs of hydrogen bonds of the form $\{i, j\}$ and $\{i+1, j-1\}$, called *stacking pairs*, tend to contribute quite substantially to the overall stability of the folding structure of the mRNA [Ieong et al., 2003; Lyngsø and Pedersen, 2000]. A secondary structure is hence expected to have a relatively high number of stacking pairs, and therefore to induce an implied structure graph with a relatively small number of degree three vertices.

It turns out that the MRSO problem is polynomial-time solvable when either $p(G_\Gamma)$ or $q(G_\Gamma)$ are fixed. The notion of edge bipartition plays a central role in this setting.

Definition 9.3.1 ((Nice) Edge bipartition)**.** *Let G_Γ be an implied structure graph with n vertices. An* edge bipartition *$\mathcal{P} = (E_t, E_b)$ of G_Γ is a partitioning of the edges in G_Γ into E_t and E_b, the* top *and* bottom *edges of $\mathcal{P}$ respectively, such that $E_t \cup E_b = \mathbf{E}(G_\Gamma)$, $E_t \cap E_b = \emptyset$ and $E_t \neq \emptyset$. Furthermore, $\mathcal{P}$ is said to be* nice *, if the subgraph $G_\Gamma[E_t]$ is outerplanar.*

Recall that a graph is called *outerplanar* if it has an embedding in the plane such that the vertices lie on a fixed circle and the edges lie inside the disk of the circle and don't intersect.

Central in our approach is Algorithm A_{NEB} (page 97). We shall use Algorithm A_{NEB} only for a nice edge bipartition of G_Γ with a fixed number of bottom edges. At the heart of algorithm A_{NEB} lies the following simple observation. Suppose we want to find the highest-scoring compatible mRNA sequence which starts with codon AAA. For this, we can replace the similarity function $f_1 \in \mathcal{F}$ by a different function f', where $f'(\mathtt{AAA}) = f_1(\mathtt{AAA})$ and $f'(C) = -\infty$ for all codons $C \neq \mathtt{AAA}$. Solving the MRSO problem for the instance $(G_\Gamma, \mathcal{F}')$, where $\mathcal{F}' = f', f_2, \ldots, f_n$, will then give us our desired mRNA. The following definition generalizes this example.

Definition 9.3.2 (Corresponding similarity functions)**.** *Let $(G_\Gamma, \mathcal{F})$ be an instance of the* MRSO *problem with $\mathcal{F} = f_1, f_2, \ldots, f_n$. Also, let $\phi : V \to \Sigma^3$ be a codon assignment for some $V \subseteq \mathbf{V}(G_\Gamma)$. The corresponding set of similarity functions of assignment ϕ, denoted $\mathcal{F}_\phi = f_1^\phi, \ldots, f_n^\phi$, is defined as follows:*

- *For all $i \in V$: $f_i^\phi(\phi(i)) = f_i(\phi(i))$, and $f_i^\phi(C) = -\infty$ for any $C \neq \phi(i)$.*
- *For all $j \in \mathbf{V}(G_\Gamma) - V$: $f_j^\phi = f_j$.*

Algorithm A_{NEB} uses Algorithm A_{OP}, the algorithm given in [Backofen et al., 2002] for outerplanar implied structure graphs, as a subprocedure in its computation. At its core, Algorithm A_{NEB} is basically an exhaustive search procedure that searches through all possible codon assignments for vertices which are incident to edges in E_b. For each such assignment, Algorithm A_{NEB} first checks if the assignment is compatible with respect to $G_\Gamma[E_b]$, and if so, it invokes Algorithm A_{OP} with the set of similarity functions corresponding to this assignment. Any solution returned by Algorithm A_{OP} is guaranteed to be compatible

Algorithm 3: Algorithm $A_{NEB}(G_\Gamma, \mathcal{F}, \mathcal{P})$

Data: An implied structure graphG_Γ of order n, a set of similarity functions $\mathcal{F} = f_1, f_2, \ldots, f_n$ and a nice edge bipartition $\mathcal{P} = (E_t, E_b)$.
Result: An optimal target mRNA sequence T which is compatible with respect to G_Γ.
foreach *codon assignment* ϕ *to vertices incident to edges in* E_b **do**
  if ϕ *is compatible with respect to* $G_\Gamma[\mathcal{E}_b]$ **then**
    (a) Construct $\mathcal{F}_\phi$, the similarity functions corresponding to ϕ.
    (b) Invoke $A_{OP}(G_\Gamma[E_t], \mathcal{F}_\phi)$.
  end
end
return *the target mRNA sequence found in Step (*b*) with the highest similarity score.*

with G_Γ since it is simultaneously compatible with both $G_\Gamma[E_b]$ and $G_\Gamma[E_t]$. Finally, Algorithm A_{NEB} terminates by outputting the maximum solution over all target mRNAs returned by Algorithm A_{OP}. Most of our interest in Algorithm A_{NEB} stems from the following lemma.

Lemma 9.3.3 ([Blin et al., 2008])**.** *Given an instance* $(G_\Gamma, \mathcal{F})$ *for the* MRSO *problem accompanied by a nice edge bipartition* $\mathcal{P} = (E_t, E_b)$ *of* G_Γ*,* A_{NEB} *computes an optimal target mRNA sequence for this instance in* $O(2^{12m}n)$ *time, where* $n = |\mathbf{V}(G_\Gamma)|$ *and* $m = |E_b|$.

Thanks to this general lemma, we have obtained the following results (details omitted).

Proposition 9.3.4 ([Blin et al., 2008])**.** *The* MRSO *problem is solvable in* $O(2^{12p(G_\Gamma)}n)$ *time.*

Proposition 9.3.5 ([Blin et al., 2008])**.** *The* MRSO *problem is solvable in* $O(2^{12q(G_\Gamma)}n)$ *time.*

Also, omitted here are our results that the MRSO problem is solvable in polynomial-time if G_Γ has bounded *cutwidth* (although cutwidth is perhaps not as natural as the two previously discussed parameters, it has been studied quite considerably for other problems dealing with RNAs Evans [1999c]; Evans and Wareham [2001]; Jiang et al. [2000a]) or bounded *treewidth* (see [Blin et al., 2008] for details). Finally, notice that in a recent paper, the cliquewidth of G_Γ was also argued to be an interesting parameter for the MRSO problem [Gurski, 2008].

10

How many words are needed to build up all words ?

Contents

10.1 Introduction

In a *covering problem* we are faced with the following situation: We are given two (not necessarily disjoint) sets of elements, the *base elements* and the *covering elements*, and the goal is to find a minimum (weight) subset of covering elements that "*covers*"all the base elements. The exact notion of covering differs from problem to problem, yet this abstract setting is common to many classical combinatorial problems in various application areas. Two famous examples are (i) the MINIMUM SET COVER problem where the covering elements are subsets of the base elements and the notion of covering corresponds to set inclusion, and (ii) the MINIMUM VERTEX COVER problem where the setting is graph-theoretic and the notion of covering corresponds to incidence between vertices and edges. Ever since the early days of combinatorial optimization, research on covering problems such as the two examples above proved extremely fruitful in laying down fundamental techniques and ideas. The early work of Johnson [Johnson, 1974] and Lovász [Lovász, 1974] on the MINIMUM SET COVER problem pioneered the greedy analysis approach, while Chvátal [Chvátal, 1979] gave the first analysis based on linear programming (LP) while tackling the same problem. The first LP-rounding algorithm by Hochbaum [Hochbaum, 1982] was also designed for MINIMUM SET COVER, while Bar-Yehuda and Even gave the first Primal-Dual [Bar-Yehuda and Even, 1981] and Local-Ratio [Bar-Yehuda and Even, 1985] algorithms for the MINIMUM VERTEX COVER problem.

In this chapter we introduce a new covering problem which resides in the realm of strings. A string u is a *substring* of a string v, if u can be obtained by deleting any number of consecutive letters from both ends of v. In our covering problem, the base elements are strings and the covering elements are their substrings. The notion of covering corresponds to string-factorization, or to the generation of strings by substring concatenation. More formally, for a given set of strings S, let $\mathcal{C}(S)$ denote the set of all substrings of strings in

S. We define a *cover* of S to be a subset $C \subseteq \mathcal{C}(S)$ such that any string $s \in S$ can be written as a concatenation of strings in C. If each string in S can be written as a concatenation of at most ℓ strings in C, we say that C is an *ℓ-cover* of S. Given a weight function $w : \mathcal{C}(S) \to \mathbb{Q}^+$, we are interested in computing an ℓ-cover of S with minimum possible weight.

The problem is formally defined as follows.

Minimum Substring Cover

- **Input** : A set of strings S, a weight function $\omega : \mathcal{C}(S) \to \mathbb{Q}^+$, and an integer $\ell \geq 2$.
- **Solution** : An ℓ-cover C of S. That is, a set of strings $C \subseteq \mathcal{C}(S)$, where for each $s \in S$ there exist $c_1, \ldots, c_p \in C$, $p \leq \ell$, with $s = c_1 \cdots c_p$.
- **Measure** : The total weight of the cover, *i.e.*, $\omega(C) = \sum_{c \in C} w(c)$.

***Example* 8** Consider the set of strings $S = \{a, aab, aba\}$. Then

$$\mathcal{C}(S) = \{a, b, aa, ab, ba, aab, aba\}$$

and $C_1 = \{a, b\}$ and $C_2 = \{a, ab\}$ are covers of S. The cover C_1 is a 3-cover of S, while C_2 is a 2-cover. ∎

Throughout this chapter, we use n to denote the number of strings in S, and m to denote the maximum length of any string in S, *i.e.*, $n = |S|$ and $m = \max\{|s| : s \in S\}$.

Note that in case $\ell \geq m$, there is no actual bound on the concatenation length of the required cover, and this case is denoted by $\ell = \infty$. An ∞-cover is referred to simply as a cover. Another interesting special case is when $\ell = 2$. In this case, we are required to cover S with a set of prefixes and suffixes in S, where a *prefix* (resp. *suffix*) of a string s is a substring of s which is obtained by removing consecutive letters only from the end (resp. beginning) of s. As we will see, these two extremal cases both give a certain amount of combinatorial leverage, and therefore deserve particular consideration. We also wish to point out that our use of general weight functions $\omega : \mathcal{C}(S) \to \mathbb{Q}^+$ allows for more robustness in modeling different scenarios. For instance, when ω is the *unitary function*, *i.e.*, $\omega(c) = 1$ for every $c \in \mathcal{C}(S)$, this corresponds to the situation where we want to minimize the size of a cover of S. When $\omega(c) = |c|$, *i.e.*, the weight of every substring is its length (w is the *length-weighted function*), this corresponds to the case where we want to minimize the total length of the cover. Often some sort of middle ground between these two situations might also be desirable.

***Example* 9** Consider the two covers C_1 and C_2 of the set of strings S in Example 8. If ω is the unitary function, then $\omega(C_1) = \omega(C_2) = 2$. However, if ω is the length-weighted function, we have $\omega(C_1) = 2 < \omega(C_2) = 3$. ∎

Our initial inspiration for studying the MINIMUM SUBSTRING COVER problem came from a paper by Bodlaender *et al.* [Bodlaender et al., 1995], who described an application for this problem in the context of protein folding (The authors of [Bodlaender et al., 1995] actually referred to our problem as the DICTIONARY GENERATION problem, and considered its unweighted variant under the parameterized complexity framework.) Protein folding is the problem of determining the folding structure of proteins using their amino-acid sequential description. This problem is extremely important, since most of the functionally of a protein is determined by its folding structure, and because current biological methods for extracting the sequential description of a given protein exceed by far the methods for extracting the folding structure of the protein. In [Bodlaender et al., 1995], it is argued that since all known approaches for protein folding are **NP**-hard, a possible heuristic for this problem is to break the protein sequence into small segments, small enough for allowing efficient folding computation. This heuristic is justified by the fact that many proteins seem to be composed of relatively small regions which fold independently of other regions. The theory of *exon shuffling* proposes that all proteins are concatenations of such regions, where the regions are drawn from a common ancestral dictionary [Dorit and Gilbert, 1991; Patthy, 1991].

The MINIMUM SUBSTRING COVER problem can also model interesting computational issues which arise in formal language theory, and in particular, in the area of combinatorics of words. Our notion of ∞-cover actually corresponds to the notion of *combinatorial rank*, an important parameter of a set of words (the best general reference here is [Choffrut and Karhumäki, 1997]). Néraud [Néraud, 1990] studied the problem of determining whether a given set of words is *elementary*, where a set of strings is said to be elementary if it does not have a cover of size strictly less than its own. Neraud describes a direct application of this notion to the famous D0L-sequence equivalence problem (see [Rozenberg and Salomaa, 1980]) via so-called elementary morphisms [Ehrenfeucht and Rozenberg, 1978]. He also argues that this notion appears frequently in numerous sub-areas such as test sets, code theory, representation of formal languages, and the theory of equations in free monoids. His main result is in showing that deciding whether a given set of words is elementary is **coNP**-complete, which implies that MINIMUM SUBSTRING COVER is **NP**-hard.

Apart from the work of Bodlaender *et al.* [Bodlaender et al., 1995] and Néraud [Néraud, 1990], there has also been some recent work on problems closely related to MINIMUM SUBSTRING COVER, especially for the case of $\ell = 2$. The MINIMUM SET COVER WITH PAIRS problem introduced by Hassin and Segev in [Hassin and Segev, 2005], is a variant of the MINIMUM SET COVER problem where base elements are now covered by pairs of sets, and the goal is to cover all base elements using a minimum weight collection of sets. Hassin and Segev gave an $O(\sqrt{n}\log(n))$ approximation algorithm for this problem, along with a few other algorithms for special cases of this problem. Another closely related problem is the HAPLOTYPE INFERENCE BY MAXIMUM PARSIMONY problem, an important problem in computational molecular biology. Huang *et al.* [Huang et al., 2005] gave an algorithm for this problem, which translates to an $O(m\log(n))$ algorithm for the MINIMUM SUBSTRING COVER problem with $\ell = 2$. Hajiaghayi *et al.* [Hajiaghayi et al., 2006] introduced the MINIMUM MULTICOLORED SUBGRAPH problem within the same context, and gave an algorithm which in our terms obtains a performance guarantee of $O(\sqrt{m}\log(n))$ with high probability.

10.2 Approximation and inapproximation results

We begin the presentation by giving some negative results (to fix the context). Combining an approximation preserving reduction from the MINIMUM HYPERGRAPH VERTEX COVER problem to the MINIMUM SUBSTRING COVER problem together with the results of [Raz and Safra, 1997] and [Dinur et al., 2005], we have obtained the following inapproximability result.

Proposition 10.2.1 ([Hermelin et al., 2008]). *It is* **NP***-hard to approximate the* MINIMUM SUBSTRING COVER *problem (i) within ratio* $c\log(n)$ *for some constant* c*, and (ii) within ratio* $\lfloor m/2 \rfloor - 1 - \varepsilon$ *for any* $\varepsilon > 0$*.*

It is worth noticing that the above proposition relies (i) on a somewhat unnatural weight function ω, and (ii) on the fact that the strings in S are allowed to be fairly long. The following proposition relaxes both these conditions at the cost of reducing the lower bounds to only a constant.

Proposition 10.2.2 ([Hermelin et al., 2008]). *The* MINIMUM SUBSTRING COVER *problem is* **NP***-hard to approximate (i) within ratio* $c\log(n)$ *for some constant* c*, (ii) within ratio* $\lfloor m/2 \rfloor - 1 - \varepsilon$ *for any* $\varepsilon > 0$*, and (iii) within some constant* c*, when* m *and* ℓ *are constant, and* ω *is either the unitary or the length-weighted function.*

We now turn to presenting some positive results in the form of approximations algorithms with performance ratios depending on the length of the longest word in S. We first apply the local-ratio technique [Bar-Yehuda, 2000; Bar-Yehuda and Even, 1985], and next linear programming techniques. The local-ratio technique [Bar-Yehuda, 2000] is based on the Local-Ratio Lemma [Bar-Yehuda and Even, 1985], which in our terms is stated as follows.

Lemma 10.2.3 (Local-Ratio). *Let* C *be a cover for* S*, and let* ω_1 *and* ω_2 *be weight functions for* $\mathcal{C}(S)$*. If* C *is an* α*-approximate solution, both with respect to* w_1 *and with respect to* w_2*, then* C *is also an* α*-approximate solution with respect to* $\omega_1 + \omega_2$*.*

We need a new definition. Given a weight function $\omega : \mathcal{C}(S) \to \mathbb{Q}^+$, we say that ω is *proper* if for any $c, c_1 \in \mathcal{C}(S)$, $\omega(c_1) \leq \omega(c)$ whenever c_1 is a prefix or a suffix of c. For example, unitary and length-weighted functions are proper. Based on the local-ratio technique, we have obtained the following positive results.

Proposition 10.2.4 ([Hermelin et al., 2008]). *The* MINIMUM SUBSTRING COVER *problem is approximable (i) within ratio* $\binom{m+1}{2} - 1$ *for general values of* ℓ*, (ii) within ratio* $m - 1$ *for* $\ell = 2$*, and (iii) within ratio* m *for* $\ell = \infty$ *and proper weight function* ω*.*

Is the MINIMUM SUBSTRING COVER problem approximable within ratio m for $\ell = \infty$ if the weight function is not proper?

We now consider the MINIMUM SUBSTRING COVER problem from a linear programming point of view. We have shown in [Hermelin et al., 2008] that the MINIMUM SUBSTRING COVER problem can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} \frac{(1+x_i)^2}{4} \\ \text{subject to} \quad & \sum_{v_p\, v_q = u_i} \frac{(1+x_p)(1+x_q)}{4} \geq 1, \qquad i = 1, 2, \ldots, n \\ & x_i \in \{-1, +1\}, \qquad j = 1, 2, \ldots, m \end{aligned} \tag{10.1}$$

By combining the above linear program with a randomized rounding procedure we have obtained the following approximation result.

Proposition 10.2.5 ([Hermelin et al., 2008]). *With high probability, the* MINIMUM SUBSTRING COVER *problem is approximable within ratio* $O(m^{(\ell-1)^2/\ell} \log^{1/\ell}(n))$.

10.3 Jumping to numbers

10.3.1 Introduction

What is the complexity of the MINIMUM SET COVER problem for a one-letter alphabet? If $\ell = 2$ in addition? Unfortunately, we don't have any answer for that, even for $\ell = 2$. The problem is, however, known to be **APX**-hard for a binary alphabet (D. Hermelin, and S. Vialette, Unpublished result), but none of our attempts succeeded in proving either **NP**-hardness or membership to **P** for a one-letter alphabet. We consider in this section the MINIMUM SET COVER problem for a one-letter alphabet ... in quite a relaxed form. More precisely, what about the complexity of the MINIMUM SET COVER problem for a one-letter alphabet in case the strings are given by their length? Indeed, in the special case of a one-letter alphabet, there is no ambiguity in giving either the string or their length. Observe that the two problems are, however, different from an algorithmic theory point of view. Indeed, if S is composed of n strings, each of length at most m, the size of the input is $O(nm)$ whereas it reduces to $O(n \log(m))$ if the strings are presented by their length (assuming a natural binary encoding). This section is devoted to investigating this problem, referred hereafter as the MINIMUM k-GENERATING SET problem.

We use $\mathbb{N}^*$ to refer to the set of all natural numbers excluding zero, *i.e.*, $\mathbb{N}^* = \{1, 2, \ldots\}$. Let $S = \{s_1, s_2, \ldots, s_n\} \subset \mathbb{N}^*$. For any $k \in \mathbb{N}^*$, we write kS for the set of all integers that can be expressed as the

What is the complexity of the MINIMUM SET COVER problem for a one-letter alphabet and $\ell = 2$? This question to equivalent to deciding whether the MINIMUM 2-GENERATING SET problem is strongly **NP**-complete.

sum of *exactly* k *non necessarily distinct* integers of S, *i.e.*, $kS = \{s_{i_1} + s_{i_2} + \ldots + s_{i_k} : s_{i_1}, s_{i_2}, \ldots, s_{i_k} \in S\}$. According to this definition, for any set S, $S = 1S$. A set $X \subset \mathbb{N}^*$ is a k-*generating set* of S (or k-*generates* S) if $S \subseteq \bigcup_{i=1}^{k} iX$. (Notice here that we do not require the additional constraint $\bigcup_{i=1}^{k} iX \subseteq S$.) It is called a *minimum* k-generating set of S if X is a k-generating set of S of minimum cardinality. The k-*rank* of S, denoted $\mathrm{rk}_k(S)$, is the cardinality of a minimum k-generating set of S. A set $S \subset \mathbb{N}^*$ is k-*elementary* if $\mathrm{rk}_k(S) = |S|$. Let $\min(S)$ and $\max(S)$ stand for $\min\{s_i : s_i \in S\}$ and $\max\{s_i : s_i \in S\}$, respectively.

MINIMUM k-GENERATING SET

- **Input** : A set $S = \{s_1, s_2, \ldots, s_n\} \subset \mathbb{N}^*$, and a positive integer k.
- **Solution** : A k-generating set X of S, *i.e.*, a set $X \subset \mathbb{N}^*$ such that $S \subseteq \bigcup_{1 \leq i \leq k} iX$.
- **Measure** : The cardinality of X.

Notice that it has been shown (in the context of conformal radiotherapy and Intensity Modulated Radiation Therapy) that computing the k-ranks of a set of integers *for unbounded* k is **NP**-complete [Collins et al., 2007].

We focus here only on the case $k = 2$. In [Fagnot et al., 2009], some easy properties (including expansion, contraction and shift) of minimum 2-generating sets are given. Actually, from our point of view, the only property that does not follow intuition is that, for any $S \subset \mathbb{N}^*$ and any $c \in \mathbb{N}^*$ common divisor of S, we have $\mathrm{rk}_2(S/c) = \mathrm{rk}_2(S)$ if c is odd and $\mathrm{rk}_2(S) \leq \mathrm{rk}_2(S/c) \leq 2\,\mathrm{rk}_2(S)$ if c is even, where $S/c = \{s_i/c : s_i \in S\}$. In other words, dividing the elements of S by a common divisor does not reduce the 2-rank (thereby ruling out this approach for approximation considerations), and the result is quite tight (see [Fagnot et al., 2009]).

10.3.2 Hardness

The complexity of the MINIMUM 2-GENERATING SET problem is settled in [Fagnot et al., 2009].

Proposition 10.3.1 ([Fagnot et al., 2009])**.** *The* MINIMUM 2-GENERATING SET *problem is* **APX**-*hard.*

The proof is from the VERTEX COVER problem for cubic graphs and uses combinatorial properties (more precisely matching properties) of intervals [Gyárfás, 2003].

It remains open, however, whether the MINIMUM 2-GENERATING SET problem is **NP**-complete if every integer in S is bounded by a polynomial in the length of the input. We now complement the above hardness result by presenting a new result that shows intractability of finding any non-trivial 2-generating set. We need a new definition. A set $S \subset \mathbb{N}^*$ is said to be k-*simplifiable* , $k \in \mathbb{N}^*$, if there exists a k-generating set X of S such that $|X| < |S|$. In other words, S is k-simplifiable if $\mathrm{rk}_k(S) < |S|$.

Proposition 10.3.2. *Deciding whether a set* $S \subset \mathbb{N}^*$ *is 2-simplifiable is* **coNP**-*hard.*

Proof. The reduction if from the EQUAL SUM SUBSET OF EQUAL CARDINALITY problem: Given a set $T \subset \mathbb{N}^*$, are there two disjoint nonempty subsets $A, B \subseteq T$ with $|A| = |B|$ such that $\sum_{a \in A} a = \sum_{b \in B} b$? The EQUAL SUM SUBSET OF EQUAL CARDINALITY problem has been shown to be **NP**-complete in [Cieliebak et al., 2008].

Computing a minimum 2-generating set of $\{1, 2, \ldots, n\}$ turns out to be a special case of the POSTAGE STAMP problem ([Alter and Barnett, 1980; Tripathi, 2006]). The POSTAGE STAMP problem is defined as follows: for fixed positive integers $h, k \in \mathbb{N}^*$, find $N(h, k)$, *i.e.*, the largest n for which a h-generating set of $\{1, 2, \ldots, n\}$ of size k exists. It is easily seen that $N(1, k) = k$ (for $\{1, 2, \ldots, k\}$) and $N(h, 1) = h$ (for $\{1\}$). Stöhr [Stöhr, 1955a,b] proved that $N(h, 2) = \lfloor (h^2 + 6h + 1)/4 \rfloor$ (Tripathi gives an alternate proof in [Tripathi, 2006]). Surprisingly enough, no other closed-form formula is known for any other pair (h, k) where one of h and k is fixed [Tripathi, 2006]. Computing a closed-form formula for $N(2, k)$ (or, going back to our vocabulary, "*computing* $rk_2(\{1, 2, \ldots, n\})$") remains a challenging open problem (an asymptotic bound for $N(2, k)$ is given in [Moser, 1960]).
To give some insights of the context, we give the minimum size of of each 2-generating set of $\{1, 2, \ldots, n\}$, *i.e.*, $rk_2(\{1, 2, \ldots, n\})$, for n from 1 to 64 (notice that $rk_2(\{1, 2, \ldots, 65\}) = 13$).

n	$rk_2(\{1, 2, \ldots, n\})$	length of the slot
$1, 2$	1	2
$3, 4$	2	2
$5, 6, 7, 8$	3	4
$9, 10, 11, 12$	4	4
$13, 14, 15, 16$	5	4
$17, 18, 19, 20$	6	4
$21, 22, \ldots, 26$	7	6
$27, 28, \ldots, 32$	8	6
$33, 34, \ldots, 40$	9	8
$41, 42, \ldots, 46$	10	6
$47, 48, \ldots, 54$	11	8
$55, 56, \ldots, 64$	12	10

Let $n \in \mathbb{N}^*$, and X be a minimum 2-generating set of $\{1, 2, \ldots, n\}$. We certainly have $\max(X) \geq \lceil \frac{n}{2} \rceil$ (indeed, $(\lceil \frac{n}{2} \rceil - 1) + (\lceil \frac{n}{2} \rceil - 1) < n$). We conjecture that $\max(X) \leq \lceil \frac{n}{2} \rceil$ is actually enough.

Conjecture 5. *For every $n \in \mathbb{N}^*$, there exists a minimum 2-generating set X of $\{1, 2, \ldots, n\}$ such that $\max(X) = \lceil \frac{n}{2} \rceil$.*

A computer-aided verification (linear programming) shows the conjecture to be correct for $1 \leq n \leq 80$. If True, this property might help in better understanding the structure of optimal 2-generating sets (more precisely, the structure of some interesting optimal 2-generating sets) of consecutive integers.

Let $T = \{t_1, t_2, \ldots t_n\} \subset \mathbb{N}^*$ be an arbitrary instance of the EQUAL SUM SUBSET OF EQUAL CARDINALITY problem. We are due to find out whether this set contains two disjoint subsets with the same cardinality and with the same sum. We take a rather big natural B ($B = 1 + \sum_{i=1}^{n} t_i$ is actually enough) and define the set $S = \{s_0, s_1, \ldots, s_n\} \subset \mathbb{N}^*$ by $s_0 = 1$ and $s_i = (2(t_i + B) + 1, 1 \leq i \leq n$. We claim that S is 2-simplifiable if and only if the original set T admits two disjoint subsets of equal sum and cardinality.

Assume for simplicity that $t_1 + t_3 + \ldots + t_{2k-1} = t_2 + t_4 + \ldots + t_{2k}$ for some k. This is clearly equivalent to assuming that $s_1 + s_3 + \ldots + s_{2k-1} = s_2 + s_4 + \ldots + s_{2k}$. Assume further that $t_{2i-1} \leq t_{2j-1}$ and $t_{2i} \geq t_{2j}$

for all $1 \leq i < j \leq k$. This can be safely assumed through a relabeling of the indexes of $t_1, t_3, \ldots, t_{2k-1}$ (and of $s_1, s_3, \ldots, s_{2k-1}$) and a relabeling of the indexes of $t_2, t_4, \ldots, t_{2k}$ (and of $s_2, s_4, \ldots, s_{2k}$). Now, we have that

$$s_1 + s_3 + \ldots + s_{2q-1} \leq s_2 + s_4 + \ldots + s_{2q} \tag{10.2}$$

for every $1 \leq q \leq k$. Consider now the $2k$ naturals $d_0, d_1, d_2, \ldots, d_{2k-1}$ defined by $d_0 = 1$ and $d_i = s_i - d_{i-1}$ for $1 \leq i \leq 2k-1$. We claim that each d_i, $1 \leq i \leq 2k-1$, is a positive integer. Indeed, for $1 \leq i \leq k-1$, we have

$$\begin{aligned} d_{2i+1} &= s_{2i+1} - d_{2i} \\ &= (s_1 + s_3 + \ldots + s_{2i+1}) - s_0 - (s_2 + s_4 + \ldots + s_{2i}) \\ &= 2(t_1 + t_2 + \ldots + t_{2i+1}) + 2B(i+1) + (i+1) - s_0 - 2(t_2 + t_4 + \ldots + t_{2i}) - 2Bi - i \\ &= 2(t_1 + t_2 + \ldots + t_{2i+1}) - 2(t_2 + t_4 + \ldots + t_{2i}) + 2B + 1 \\ &> 0 \end{aligned}$$

for $B > \sum_{i=1}^{n} t_i$, and

$$\begin{aligned} d_{2i} &= s_{2i} - d_{2i-1} \\ &= (s_2 + s_4 + \ldots + s_{2i}) + s_0 - (s_1 + s_3 + \ldots + s_{2i-1}) \\ &\geq s_0 \\ &= 1 \end{aligned}$$

by (10.2). Notice that it follows from the above that

$$d_{2k-1} = (s_1 + s_3 + \ldots + s_{2k-1}) - s_0 - (s_2 + s_4 + \ldots + s_{2k-2}).$$

Combining this with our hypothesis

$$s_{2k} = (s_1 + s_3 + \ldots + s_{2k-1}) - (s_2 + s_4 + \ldots + s_{k-2})$$

yields $s_{2k} = d_{2k-1} + s_0 = d_{2k-1} + d_0$, and hence $s_i = d_{i-1} + d_{i \bmod 2k}$ holds for $1 \leq i \leq 2k$. Then it follows that $X = \{d_i : 0 \leq i \leq 2k-1\} \cup \{s_i : 2k+1 \leq i \leq n\}$ is a 2-generating set of S. But $|X| = n - 1 < n = |S|$, and hence S is 2-simplifiable.

For the other direction, assume the set $S = \{s_0, s_1, \ldots s_n\}$ is 2-simplifiable and let $D = \{d_1, d_2, \ldots, d_k\}$, $k \leq n$, be a 2-generating set for S. Let us represent this situation with a graph G (for the very first time in this manuscript we shall allow for loops in a graph). The graph G has $k+1$ vertices labeled $0, d_1, \ldots d_k$. As for the edges, for every $0 \leq i \leq n$, we have an edge labeled s_i between two vertices d_p and d_q such that $s_i = d_p + d_q$ (if there exists severals possibilities for 2-generating s_i with D we choose one arbitrarily). Notice that, we have a loop labeled s_i if $s_i = d_p + d_p$ for some $d_p \in D$, and en edge labeled s_i with an endpoint labeled 0 allows us to conveniently represent the case $s_i = d_p$ as $s_i = d_p + 0$, $d_p \in D$.

Since G has $k+1 \leq n+1$ vertices and $n+1$ edges then G must contain some cycle C. Furthermore, the sum of the labels on the edges of C is twice the sum of the labels on the nodes of C, and must therefore be an even number. Now, remembering that the s_i's are all odd (by construction from the t_i's) it follows that C is an even cycle. Moreover, the edges of C can be partitioned into two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$. Notice now that the sum of the labels on the edges of $\mathcal{M}_1$ equals the sum of the labels on the edges of $\mathcal{M}_2$. From this, and since B is so big, we can exclude the possibility that one edge of C is labelled with $s_0 = 1$. Then it follows that the edges of $\mathcal{M}_1$ and the edges of $\mathcal{M}_2$ give two disjoint subsets of $T = \{t_1, t_2, \ldots, t_n\}$ with equal sum and cardinality. □

Observe, however, that Proposition 10.3.2 does rule out the existence of an approximation algorithm for finding a minimum 2-generating set. Quite a surprising fact at first glance!

10.3.3 Put the blame on $\mathrm{rk}_2(S)$

Write $n = |S|$, $m = \max(S)$ and $k = \mathrm{rk}_2(S)$. We now consider the problem of finding a minimum cardinality 2-generating set of S from an effective computational point of view [Downey and Fellows, 1999; Niedermeier, 2006]. As a first attempt, let us consider the brute-force approach (we do think it is good practices to always begin with the naive brute-force algorithm): generate all k-subsets X of $\{1, 2, \ldots, m\}$ and check for each of them whether 2-generates S, *i.e.*, $S \subseteq X \cup 2X$. Correctness of this algorithm is of course immediate. There are $\binom{m}{k}$ such subsets and each subset X can be identified as a 2-generating set of S in $O(k^2 \log(k))$ time (assuming a unit-cost RAM model with $\log(m)$ word size). Therefore, the brute-force algorithm is, as a whole, a $O(m^k k^2 \log(k))$ time procedure. But m (and even $\log(m)$) can be arbitrarily large compared to $n = O(k^2)$ and this naturally leads us to the problem of trying to confine the seemingly inevitable combinatorial explosion of computational difficulty to a function of k only [Downey and Fellows, 1999; Niedermeier, 2006]. We prove here that such an algorithm does exist for finding a minimum cardinality 2-generating set of S. The following lemma is central in our approach.

Lemma 10.3.3 ([Fagnot et al., 2009])**.** *Let $S = \{s_i : 1 \leq i \leq n\} \subset \mathbb{N}^*$ and X be a minimum 2-generating set of S. There exist rationals $\alpha_{i,j} \in \{-1, -2^{-1}, 0, 2^{-1}, 1\}$, $1 \leq i \leq \mathrm{rk}_2(S)$ and $1 \leq j \leq n$, such that*

$$X = \left\{ \sum_{j=1}^{n} \alpha_{i,j}\, s_j : 1 \leq i \leq \mathrm{rk}_2(S) \right\}.$$

Combining Lemma 10.3.3 with a brute-force algorithm for finding a (representation of a) minimum cardinality 2-generating set of a set $S \in \mathbb{N}^*$ yields the following result.

Proposition 10.3.4. *Assuming a unit-cost RAM model with $\log(m)$ word size ($m = \max(S)$), there exists a $O(5^{\frac{k^2(k+3)}{2}}\, k^2 \log(k))$ time algorithm for finding a minimum cardinality 2-generating set of S, where $k = \mathrm{rk}_2(S)$.*

Let us conclude by clarifying a point that we think some people have found confusing: Can integer linear programming techniques cope here with trying to confine the seemingly inevitable combinatorial explosion to a function of k only? For one a classical result in parameterized algorithms is that the INTEGER LINEAR PROGRAMMING problem parametrized by the number of variables is fixed-parameter tractable. This powerful result, first proved by Lenstra in [Lenstra, 1983] (this paper received Fulkerson Prize in 1985 for an outstanding contribution in the area of discrete mathematics) and later improved by Kannan [Kannan, 1987]. For another, it is not hard to design an integer linear program for finding a minimum cardinality 2-generating set of a set of integers. Therefore, integer linear programming seems to be an appealing approach in our context, and Yes this is indeed a good question. However, as long as our integer linear program uses a number of variables depending – at least linearly – on $\max(S)$, Lenstra's result does not apply since $\max(S)$ can be arbitrarily big compared to $|S|$. We did not succeed in obtaining such an integer linear program (and actually we doubt such an approach is possible).

Before concluding, we would like to address the following question: does there exist a better representation lemma to improve Proposition 10.3.4 ? To this end, consider the set $\mathbf{A} \subset \mathcal{P}(\mathbb{Q})$ defined as follows: $A \in \mathbf{A}$ if and only if, for any set $S = \{s_i : 1 \leq i \leq n\} \subset \mathbb{Z}^+$, $\mathrm{rk}_2(S) = k$, there exist rationals $\alpha_{i,j} \in A$, $1 \leq i \leq n$ and $1 \leq j \leq k$, such that $X = \left\{\sum_{j=1}^{n} \alpha_{i,j} s_j : 1 \leq i \leq k\right\}$ is a (necessarily minimum) 2-generating set of S. According to Lemma 10.3.3, $\mathbf{A}$ is not empty as it contains $\{-1, -1/2, 0, 1/2, 1\}$. Furthermore, as shown in Proposition 10.3.4, the time complexity of our algorithm to compute a minimum 2-generating set depends - in an exponential way - on the size of the minimum cardinality set in $\mathbf{A}$. For any $i \in \mathbb{Z}^+$, define $\mathbf{A}_i \subseteq \mathbf{A}$ to be the set of those sets of $\mathbf{A}$ of cardinality i. Our question thus reduces to finding the minimum $i \in \mathbb{Z}^+$ such that $\mathbf{A}_i \neq \emptyset$. Lemma 10.3.3 shows that $\mathbf{A}_5 \neq \emptyset$. Proving or disproving $\mathbf{A}_i \neq \emptyset$ for some $1 \leq i < 5$ remains a challenging problem.

Here are some lines of thought to reduce the above problem to "*Is* $\mathbf{A}_4$ *empty?*". Let $A \in \mathbf{A}$. We first observe that $0 \in A$ (it is enough to consider the set $S = \{1, n\}$, for any unbounded n). Furthermore, we must have $1 \in A$ (it is enough here to consider the set $S = \{1\}$). As an immediate consequence, $\mathbf{A}_1 = \emptyset$ since $|A| \geq 2$. We now show that $\mathbf{A}_2 = \emptyset$, *i.e.*, $\{0, 1\} \notin \mathbf{A}_2$. Indeed, the set $S_1 = \{4, 5, 6\}$ has a unique minimum cardinality 2-generating set, namely $X_1 = \{2, 3\}$. But $2 \neq \sum_{i \in S_1} \alpha_i i$ for $\alpha_i \in \{0, 1\}$, $i \in S_1$, and hence $\{0, 1\} \notin \mathbf{A}_2$. Therefore, $\mathbf{A}_2 = \emptyset$. We now turn to proving that $\mathbf{A}_3 = \emptyset$. Suppose, aiming at a contradiction, that $\mathbf{A}_3 \neq \emptyset$, and let $A \in \mathbf{A}_3$. According to the above, A has the general form $\{0, 1, x\}$ for some $x \in \mathbb{Q}$. For one, as we already noticed, the set $S_1 = \{4, 5, 6\}$ has a unique minimum cardinality 2-generating set, namely $X_1 = \{2, 3\}$. Considering the pair (S_1, X_1), an exhaustive computation now shows that $A \in \mathbf{A}_3 \subseteq \mathbf{A}_{S_1}$, where

$$\begin{aligned}\mathbf{A}_{S_1} = \{ &\{-1, 0, 1\}, \{-3/4, 0, 1\}, \{-1/2, 0, 1\}, \{-1/3, 0, 1\}, \{0, 1/5, 1\}, \\ &\{0, 1/3, 1\}, \{0, 1/2, 1\} \}.\end{aligned}$$

For another, the set $S_2 = \{6, 7, 8\}$ has a unique minimum cardinality 2-generating set, namely $X_2 = \{3, 4\}$. Considering the pair (S_2, X_2), an exhaustive computation shows that $A \in \mathbf{A}_3 \subseteq \mathbf{A}_{S_2}$,where

$$\mathbf{A}_{S_2} = \{\{-3/8, 0, 1\}, \{-1/2, 0, 1\}, \{-2/3, 0, 1\}, \{-2/7, 0, 1\}, \{0, 1/2, 1\}\}.$$

Then it follows that $A \in \mathbf{A}_3 \subseteq \mathbf{A}_{S_1} \cap \mathbf{A}_{S_2} = \{\{-1/2, 0, 1\}, \{0, 1/2, 1\}\}$. We now proceed to show that neither $\{-1/2, 0, 1\}$ nor $\{0, 1/2, 1\}$ belongs to $\mathbf{A}_3$, thereby proving $\mathbf{A}_3 = \emptyset$. Indeed, consider first the set $S' = \{3, 4, 6, 7, 8, 10, 14, 15, 16, 18, 22, 30\}$. It has a unique minimum cardinality 2-generating set, namely $X' = \{1, 3, 7, 15\}$. But $1 \neq \sum_{i \in S'} \alpha_i i$ for $\alpha_i \in \{0, 1/2, 1\}$, $i \in S'$, since $1 < \min(S')$ and all rationals in $\{0, 1/2, 1\}$ are positive. Then it follows that $\{0, 1/2, 1\} \notin \mathbf{A}_3$. Consider now the set $S'' = \{4, 9, 11\}$. It has a unique minimum cardinality 2-generating set, namely $X'' = \{2, 9\}$. But $2 \neq \sum_{i \in S''} i \alpha_i$ for $\alpha_i \in \{-1/2, 0, 1\}$, $i \in S''$. Indeed, we must have $\alpha_i = -1/2$ for some $i \in S''$ since $2 < \min\{i : i \in S''\}$. We now need to consider three cases. First, if $\alpha_9 = -1/2$ then we must have $\alpha_{11} = -1/2$ since 9 and 11 are odd integers and 4 is even. But $-\frac{9+11}{2} + 4 = -6 < 2$, a contradiction. Second, if $\alpha_{11} = -1/2$, we also end up with a contradiction by a symmetric argument to one from the previous case. Third, if $\alpha_4 = -1/2$, $\alpha_9 \in \{0, 1\}$ and $\alpha_{11} \in \{0, 1\}$, we obtain $4 = 9\alpha_9 + 11\alpha_{11}$ for $\alpha_9 \in \{0, 1\}$ and $\alpha_{11} \in \{0, 1\}$, which is impossible. Then it follows that $\{-1/2, 0, 1\} \notin \mathbf{A}_3$. Combining $\mathbf{A}_3 \subseteq \{\{-1/2, 0, 1\}, \{0, 1/2, 1\}\}$ with $\{0, 1/2, 1\} \notin \mathbf{A}_3$ and $\{-1/2, 0, 1\} \notin \mathbf{A}_3$, we obtain $\mathbf{A}_3 = \emptyset$.

Finally, notice that, according to the above, if any set $A \in \mathbf{A}$ has to be symmetric around zero (note that since our solution subset $\{-1, -1/2, 0, 1/2, +1\}$ is, this may be an intrinsic property), then we would immediately obtain the desired result $\mathbf{A}_4 = \emptyset$. Proving or disproving this symmetry property remains an intriguing open problem as well.

Perspectives

Thanks to my *thinking notes* along this manuscript, I think I have already pointed out most of the problems and questions related to my exposition I am interested in. I hope I have succeeded in trying to convince the reader that multidimensional intervals, linear matchings, permutations, connected occurrences and k-generating sets are nice, interesting and useful (and fun? quite an important point for me) combinatorial objects for algorithmic investigations. It is thus understood that the above-mentioned topics will constitute definitively a large part of my research in the coming years. Therefore I shall not reproduce nor discuss any longer any of them here, and I shall focus instead on some new additional topics.

In the sequel, I present and briefly discuss four research topics I am particularly interested in and on which I plan to work in the very near future. Let me first put these different topics in their context. Both the MINIMUM COMMON STRING PARTITION problem and the Tandem duplication-random loss model are related to comparative genomics. Actually, I think that these two problems (together with algorithmic aspects of next generation sequencing) will constitute in a short-term – not to say are now – my main research activity in comparative genomics (I am now left with the feeling that heuristic approaches is the best we can do for the match-an-prune model). The MEDIAN problem for the Kendall-Tau distance is related to rank aggregation. Finally, I present one combinatorial problem on graphs that has recently caught most of our attention.

The MAXIMUM COMMON STRING PARTITION problem

A partition of a string u is a sequence $P = (p_1, p_2, \ldots, p_m)$ of strings, called the *blocks*, whose concatenation is equal to u, *i.e.*, $u = p_1\, p_2\, \ldots\, p_m$. Given a partition P of a string u and a partition Q of a string v, the pair (P, Q) is a *common partition* of u and v if Q is a permutation of P. The MINIMUM COMMON STRING PARTITION (MCSP) problem is to find a common partition of two strings u and v with the minimum number of blocks. The restricted version of MCSP where each letter occurs at most k times in each input string, is denoted by k-MCSP. It has been shown that the 2-MCSP problem is **NP**-hard and, moreover, even **APX**-hard [Kolman et al., 2005]. The 2-MCSP and 3-MCSP problems are approximable within ratio 1.1037 and 4, respectively [Kolman et al., 2005]. The MCSP problem is approximable within ratio $O(k)$, where k is the maximum number of occurrences of a letter, and within ratio $O(\log(n)\log^*(n))$ in the general case [Kaplan and Shafrir, 2006].

Let us embed the MCSP problem into multidimensional interval graphs. Recall that the MINIMUM INDEPENDENT DOMINATING SET problem is to find in a graph G is minimum cardinality subset $V' \subseteq \mathbf{V}(G)$ such that (i) V' is an independent set and (ii) for each $u \in \mathbf{V}(G) \setminus V'$, $N_G(u) \cap V' \neq \emptyset$. Most of our interest in

the MINIMUM INDEPENDENT DOMINATING SET problem stems from the following easy observation: *Given two strings u and v, both of length n, built over some alphabet $\mathcal{A}$ such that $|u|_a = |v|_a$ for all $a \in \mathcal{A}$, one can construct in polynomial-time a 2-track interval graph $G = \Omega(\mathbf{D})$ with at most $n(n+1)/2$ vertices such that independent dominating sets of size k of G are in one-to-one correspondence with common partitions of size k of u and v.* What about approximating the MINIMUM INDEPENDENT DOMINATING SET problem for 2-track interval graphs? The nice results of [Butman et al., 2007] do not help much here as it seems impossible (let me moderate this, D. Rawitz and I did not succeed) to push up the local-ratio based approximation for the MINIMUM DOMINATING SET problem in d-interval graphs, $d \geq 2$, to the MINIMUM INDEPENDENT DOMINATING SET problem for d-interval (or even d-track interval) graphs. However, it is conceivable that this problem is in **APX** for d-track interval (and actually even for d-interval) graphs (even if we have proved this problem to be **W[1]**-hard for 2-interval graphs [Hermelin et al., 2009]). Notice that, even if most – not to say all – standard combinatorial graph problems remain **NP**-complete for 2-interval graphs, they usually belong to **APX**. In any case, I believe this point of view might shed some new light upon the approximation of the MCSP problem.

Tandem duplication-random loss model

In the Tandem duplication-random loss model, a genome (given in the form of a permutation) evolves via the tandem duplication of a contiguous segment of genes (*i.e.*, the duplicated copy is inserted immediately after the original copy), followed by the loss of one copy of each of the duplicated genes. In most, though not all cases, this process will result in a genome rearrangement [Chaudhuri et al., 2006]. See Figure 10.1 for an illustration; the goal is to find a minimum cost sequence of duplication-loss steps to transform the identity into a target permutation.

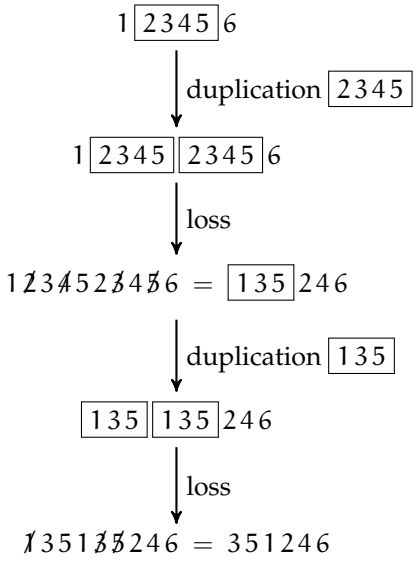


Figure 10.1: Example of a genome rearrangement caused by two rounds of tandem duplication and random loss.

Although it seems intuitively clear that the cost of a duplication event should be some non-decreasing function of the length of the duplication, it is not clear what functional form this cost duplication should take. If the cost of a duplication of a segment of k genes is α^k for some constant parameter $\alpha \geq 1$, it has been

proved in [Chaudhuri et al., 2006] that computing a minimum cost sequence of duplication-loss steps to transform the identity into a target permutation is solvable in linear-time if $\alpha = 1$ and in $O(n \log(n))$ time for any $\alpha \geq 2$ (in the latter case, the problem reduces to computing the Kendall-Tau distance). The complexity is open for $1 < \alpha < 2$. More important, the complexity of computing this distance is unknown for affine functions.

The MEDIAN problem for the Kendall-Tau distance

In the traditional (undirected) version of the MEDIAN problem, we are given k genomes and we are asked to find the genome which minimizes the sum of the distances to the other k genomes (see our monograph [Fertin et al., 2009a] for an up-to-date survey of this field). The MEDIAN problem for the Kendall-Tau distance has also been studied in the context of social choice theory and rank aggregation. It has been shown to be **NP**-complete for $k = 4$ [Dwork et al., 2001] (actually, it is claimed in [Dwork et al., 2001] that the result holds for $k \geq 4$ but it is not clear that the hardness propagates upwards to odd n, they had no success at all in convincing us), and to be approximable within ratio 1.57 [Ailon et al., 2005]. For $k = 3$, the case of most interest in the context of phylogeny since it is often use as a subroutine in phylogenetic reconstruction, **NP**-hardness has been established and nothing but a trivial $4/3$ approximation ratio is known. What about other distances? We have presented some preliminary results of this topic in [Blin et al., 2009a].

Security in graphs

Given a graph G, a subset $S \subseteq \mathbf{V}(G)$ is a *defensive alliance* if for all $u \in S$, $|N_G[u] \cap S| \geq |N_G[u] \setminus S|$, *i.e.*, every vertex $u \in S$ has as many neighbors, including u itself, in S as those in $\mathbf{V}(G) \setminus S$ [Kristiansen et al., 2004] The various concepts of alliances in graphs are motivated from a security issue that whether defenders of $u \in S$ can defeat the attackers of $v \in \mathbf{V}(G) \setminus S$. Brigham *et al.* have described a global concept of alliances based on the idea that defensive alliances do not accurately model real-world situations [Brigham et al., 2007]. Given a graph G, a subset $S \subseteq \mathbf{V}(G)$ is a *secure set* if for all $X \subseteq S$, $|N_G[X] \cap S| \geq |N_G[X] \setminus S|$, *i.e.*, every subset $X \subseteq S$ has as many neighbors, including X, in S as those in $\mathbf{V}(G) \setminus S$. The *security number* of G is the cardinality of a minimum secure set of G. The complexity of determining the security number of a graph is completely open (it has, however, be found for some families of graphs). In fact, there is no known polynomial algorithm for determining if a given set $S \subseteq \mathbf{V}(G)$ is a secure set [Dutton, 2009].

Recently, security in graphs has caught most of our attention, both from the standard complexity and the parameterized point of view. Our recent results (collaboration with R. Rizzi, Univ. Udine, Italy) suggest that (i) determining if a given set $S \subseteq \mathbf{V}(G)$ is a secure set is **coNP**-hard, and that (ii) the problem of computing the security number is not approximable but is fixed-parameter tractable for the standard parameterization (quite an unusual situation in parameterized complexity theory). Determining the security number of bipartite and bounded degree graphs is still completely open. More generally, very little is known about approximation issues of the various concepts of alliances in graphs [Rodríguez-Velázquez and Sigarreta, 2009; Favaron et al., 2004].

Bibliography

Ailon, N., Charikar, M., and Newman, A. (2005). Aggregating inconsistent information. In Gabow, H. and Fagin, R., editors, *Proc. of the 37th Annual ACM Symposium on Theory of Computing (STOC), Baltimore, MD, USA*, pages 684–693. ACM.

Akiyama, J., Exoo, G., and Harary, F. (1981). Covering and packing in graphs IV: Linear arboricity. *Networks*, 11:69–72.

Akutsu, T. (2000). Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104:45–62.

Alber, J., Gramm, J., Guo, J., and Niedermeier, R. (2002). Towards optimally solving the longest common subsequence problem for sequences with nested arc annotations in linear time. In Apostolico, A. and Takeda, M., editors, *Proc. 13th Annual Symposium on Combinatorial Pattern Matching (CPM), Fukuoka, Japan*, volume 2373 of *Lecture Notes in Computer Science*, pages 99–114. Springer.

Alber, J., Gramm, J., Guo, J., and Niedermeier, R. (2004). Computing the similarity of two sequences with nested arc annotations. *Theoretical Computer Science*, 312(2-3):337–358.

Albert, M., Aldred, R., Atkinson, M., and Holton, D. (2001). Algorithms for pattern involvement in permutations. In *Proc. International Symposium on Algorithms and Computation (ISAAC)*, volume 2223 of *Lecture Notes in Computer Science*, pages 355–366.

Alekseyev, M. and Pevzner, P. (2007). Colored de Bruijn graphs and the genome halving problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):98–107.

Alm, E. and Arkin, A. (2003). Biological networks. *Curr. Opin. Struct. Biol.*, 13(2):193–202.

Alon, N. (1988). The linear arboricity of graphs. *Israel J. of Mathematics*, 62(3):311–325.

Alon, N. and Spencer, J. (1992). *The Probabilistic Method*. Wiley.

Alon, N., Yuster, R., and Zwick, U. (1995). Color coding. *Journal of the ACM*, 42(4):844–856.

Alonso, L. and Schott, R. (1993). On the tree inclusion problem. In Borzyszkowski, A. and Sokolowski, S., editors, *Proc. 18th Mathematical Foundations of Computer Science (MFCS), Gdansk, Poland*, volume 711 of *Lecture Notes in Computer Science*, pages 211–221.

Alter, R. and Barnett, J. (1980). A postage stamp problem. *Amer. Math. Montly*, 87:206–210.

Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., and Vialette, S. (2007a). A pseudo-boolean programming approach for computing the breakpoint distance between two genomes with duplicate genes. In *Proc. 5th RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG)*, volume 4751 of *Lecture Notes in Bioinformatics*, pages 16–29. Springer.

Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., and Vialette, S. (2008a). Efficient tools for computing the number of breakpoints and the number of adjacencies between two genomes with duplicate genes. *Journal of Computational Biology*. To appear.

Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., and Vialette, S. (2008b). On the approximability of comparing genomes with duplicates. *Journal of Graph Algorithms and Applications*, 13(1):19–53.

Angibaud, S., Fertin, G., Rusu, I., and Vialette, S. (2006). How pseudo-boolean programming can help genome rearrangement distance computation. In *Proc. 4th RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG), Montreal, Canada*, volume 4205 of *Lecture Notes in Bioinformatics*, pages 75–86.

Angibaud, S., Fertin, G., Rusu, I., and Vialette, S. (2007b). A general framework for computing rearrangement distances between genomes with duplicates. *Journal of Computational Biology*, 14(4):379–393.

Arnborg, S., Corneil, D., and Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *Journal on Algebraic and Discrete Methods*, 8(2):277–284.

Arnborg, S. and Proskurowski, A. (1989). Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23:11–24.

Atkinson, M., Murphy, M., and Ruskuc, N. (2005). Pattern Avoidance Classes and Subpermutations. *European Journal of Combinatorics*, 12(1).

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (1999). *Complexity and Approximation: Cominatorial optimization problems and their approximability properties*. Springer-Verlag.

Backofen, R. and Busch, A. (2004). Computational design of new and recombinant selenoproteins. In Sahinalp, S. C., Muthukrishnan, S., and Dogrusöz, U., editors, *Proc. 15th Annual Symposium on Combinatorial Pattern Matching (CPM), Istanbul, Turkey*, volume 3109 of *Lecture Notes in Computer Science*, pages 270–284.

Backofen, R., Narayanaswamy, N., and Swidan, F. (2002). Protein similarity search under mRNA structural constraints: application to targeted selenocystein insertion. *In Silico Biology*, 2(3):275–290.

Bafna, V., Narayanan, B., and Ravi., R. (1996). Non-overlapping local alignments (weighted independent sets of axis-parallel rectangles). *Discrete Applied Mathematics*, 71(1):41–54.

Bar-Yehuda, R. (2000). One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27(2):131–144.

Bar-Yehuda, R. and Even, S. (1981). A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203.

Bar-Yehuda, R. and Even, S. (1985). A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46.

Bar-Yehuda, R., Halldorsson, M., Naor, J., Shachnai, H., and Shapira, I. (2002). Scheduling split intervals. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 732–741.

Ben Rebea, A. (1981). *Étude des stables dans les graphes quasi-adjoints*. PhD thesis, Université de Grenoble.

Bergeron, A., Chauve, C., de Montgolfier, F., and Raffinot, M. (2005). Computing common intervals of k permutations, with applications to modular decomposition of graphs. In Brodal, G. S. and Leonardi, S., editors, *Proc. 13th Annual European Symposium, Palma de Mallorca, Spain*, volume 3669 of *Lecture Notes in Computer Science*, pages 779–790.

Bergroth, L., Hakonen, H., and Raita, T. (2000). A survey of longest common subsequence algorithms. In *Proc. of the 7th International Symposium on String Processing Information Retrieval (SPIRE), Coruña, Spain*, pages 39–48. IEEE Computer Society.

Betzler, N., Fellows, M., Komusiewicz, C., and Niedermeier, R. (2008). Parameterized algorithms and hardness results for some graph motif problems. In *Proc. 19th Annual Symposium on Combinatorial Pattern Matching (CPM), Pisa, Italy*, volume 5029 of *Lecture Notes in Computer Science*, pages 31–43. Springer.

Björklund, A., Husfeldt, T., Kaski, P., and Koivisto, M. (2007). Fourier meets möbius: fast subset convolution. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 67–74. ACM New York, NY, USA.

Blair, J. and Peyton, B. (1993). An introduction to chordal graphs and clique trees. *Graph Theory and Sparse Matrix Computation*, 56:1–29.

Blin, G., , Fertin, G., Herry, G., and Vialette, S. (2007a). Comparing RNA structures: towards an intermediate model between the edit and the lapcs problems. In Sagot, M.-F., Walter, W. T., and Maria, E., editors, *1st Brazilian Symposium on Bioinformatics (BSB), Angra dos Reis, Brazil*, volume 4643 of *Lecture Notes in Bioinformatics*, pages 101–112. Springer.

Blin, G., Chauve, C., and Fertin, G. (2004). The breakpoint distance for signed sequences. In *Proc. 1st Algorithms and Computational Methods for Biochemical and Evolutionary Networks (CompBioNets), Recife, Brazil*, pages 3–16. KCL publications.

Blin, G., Chauve, C., Fertin, G., Rizzi, R., and Vialette, S. (2007b). Comparing genomes with duplications: a computational complexity point of view. *ACM/IEEE Trans. Computational Biology and Bioinformatics*, 14(4):523–534.

Blin, G., Crochemore, M., Hamel, S., and Vialette, S. (2009a). Finding the median of three permutations under the Kendall-Tau distance. In *Proc. 7th annual international conference on Permutation Patterns, Firenze, Italy*. electronic version (6 pp).

Blin, G., Crochemore, M., and Vialette, S. (2010a). *Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications*, chapter Algorithmic Aspects of Arc-Annotated Sequences. Wiley. To appear.

Blin, G., Fertin, G., Hermelin, D., and Vialette, S. (2008). Fixed-parameter algorithms for protein similarity search under mrna structure constraints. *Journal of Discrete Algorithms*, 6(4):618–626.

Blin, G., Fertin, G., Rizzi, R., and Vialette, S. (2005a). What makes the arc-preserving subsequence problem hard ? *T. Comp. Sys. Biology*, 2:1–36.

Blin, G., Fertin, G., Sikora, F., and Vialette, S. (2009b). The exemplar breakpoint distance for non-trivial genomes cannot be approximated. In Das, S. and Uehara, R., editors, *Proc. 3rd Annual Workshop on Algorithms and Computation (WALCOM'09), Kolkata, India*, volume 5431 of *Lecture Notes in Computer Science*, pages 357–368. Springer.

Blin, G., Fertin, G., and Vialette, S. (2005b). What makes the arc-preserving subsequence problem hard ? *LNCS Transactions on Computational Systems Biology*, 2:1–36.

Blin, G., Fertin, G., and Vialette, S. (2007c). Extracting constrained 2-interval subsets in 2-interval sets. *Theoretical Computer Science*, 385(1-3):241–263.

Blin, G., Sikora, F., and Vialette, S. (2009c). Querying Protein-Protein Interaction Networks. In Istrail, S., Pevzner, P., and Waterman, M., editors, *5th International Symposium on Bioinformatics Research and Applications (ISBRA'09)*, volume 5542 of *LNBI*, pages 52–62, Fort Lauderdale, FL, USA. Springer-Verlag.

Blin, G., Sikora, F., and Vialette, S. (2010b). GraMoFoNe: a cytoscape plugin for querying motifs without topology in protein-protein interactions networks. In Al-Mubaid, H., editor, *2nd International Conference on Bioinformatics and Computational Biology (BICoB-2010)*, page 3843, Honolulu, USA. International Society for Computers and their Applications (ISCA).

Böch, A., Forchhammer, K., Heider, J., and Baron, C. (1991). Selenoprotein synthesis: a review. *Trends in Biochemical Sciences*, 16(2):463–467.

Bodlaender, H. (1993). A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23.

Bodlaender, H., Downey, R., Fellows, M., Hallett, M., and Wareham, H. (1995). Parameterized complexity analysis in computational biology. *Computer Applications in the Biosciences*, 11(1):49–57.

Bóna, M. (2004). *Combinatorics of permutations*. Discrete Mathematics and its Applications. Chapman & Hall/CRC.

Bongartz, D. (2004). Some notes on the complexity of protein similarity search under mRNA structure constraints. In *Proc. of the 30th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 174–183.

Bonizzoni, P., Vedova, G. D., Dondi, R., Fertin, G., Rizzi, R., and Vialette, S. (2007). Exemplar longest common subsequences. *IEEE/ACM Transactions on Computational Biology and Bionformatics*, 4(4):535–543.

Bonsma, P. (2003). Complexity results for restricted instances of a paint shop problem. Technical Report 1681, Dep. of Applied Mathematics, Univ. of Twente.

Bonsma, P., Epping, T., and Hochstättler, W. (2006). Complexity results on restricted instances of a paint shop problem for words. *Discrete Applied Mathematics*, 154(9):1335–1343.

Bose, P., J.F.Buss, and Lubiw, A. (1998). Pattern matching for permutations. *Information Processing Letters*, 65(5):277–283.

Bouvel, M., Rossin, D., and Vialette, S. (2007). Longest common separable pattern between permutations. In Ma, B. and Zhang, K., editors, *Proc. Symposium on Combinatorial Pattern Matching (CPM'07), London, Ontario, Canada*, volume 4580 of *Lecture Notes in Computer Science*, pages 316–327.

Brandstädt, A., Le, V. B., and Spinrad, J. (1999). *Graph Classes: A Survey*. volume of the SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia.

Brevier, G., Rizzi, R., and Vialette, S. (2007). Pattern matching in protein-protein interaction graphs. In Csuhaj-Varjú, E. and Ésik, Z., editors, *Proc. 16th Fundamentals of Computation Theory, 16th International Symposium (FCT), Budapest, Hungary*, Lecture Notes in Computer Science, pages 137–148. Springer.

Brevier, G., Rizzi, R., and Vialette, S. (2009). Complexity issues in color-preserving graph embeddings. *Theoretical Computer Science*. In press.

Brigham, R., Dutton, R., and Hedetniemi, S. (2007). Security in graphs. *Discrete Applied Mathematics*, 155(13):1708–1714.

Bruckner, S., Hüffner, F., Karp, R., Shamir, R., and Sharan, R. (2009a). Topology-free querying of protein interaction networks. In *Proc. 13th Annual International Conference on Computational Molecular Biology (RECOMB), Tucson, USA*, page 74. Springer.

Bruckner, S., Hüffner, F., Karp, R., Shamir, R., and Sharan, R. (2009b). Torque: topology-free querying of protein interaction networks. *Nucleic Acids Research*, 37 (Web-Server-Issue:106–108.

Bryant, D. (2000). The complexity of calculating exemplar distances. In Sankoff, D. and Nadeau, J., editors, *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, volume 1, pages 207–212. Kluwer Academic Publisher.

Bui-Xuan, B.-M., Habib, M., and Paul, C. (2005). Revisiting t. uno and m. yagiura's algorithm. In Deng, X. and Du, D.-Z., editors, *Proc. 16th International Symposium on Algorithms and Computation (ISAAC), Sanya, Hainan, China*, volume 3827 of *Lecture Notes in Computer Science*, pages 156–165.

Butman, A., Hermelin, D., Lewenstein, M., and Rawitz, D. (2007). Optimization problems in multiple-interval graphs. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. To appear.

Chaudhuri, K., Chen, K., Mihaescu, R., and Rao, S. (2006). On the tandem duplication-random loss model of genome rearrangement. In Giancarlo, R. and Sankoff, D., editors, *Proc. of the* 17*th annual ACM-SIAM symposium on Discrete algorithm (SODA), Miami, Florida, USA*, pages 564–570.

Chen, E., Yang, L., and Yuan, H. (2007a). Improved algorithms for largest cardinality 2-interval pattern problem. *Journal of Combinatorial Optimization*, 13(3):262–275.

Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S., and Jiang, T. (2005). Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):302–315.

Chen, Z., Fu, B., Xu, J., Yang, B., Zhao, Z., and Zhu, B. (2007b). Non-breaking similarity of genomes with gene repetitions. In Ma, B. and Zhang, K., editors, *Proc. Symposium on Combinatorial Pattern Matching (CPM'07), London, Ontario, Canada*, volume 4580 of *Lecture Notes in Computer Science*, pages 119–130.

Chen, Z., Fu, B., and Zhu, B. (2006). The approximability of the exemplar breakpoint distance problem. In *2nd International Conference on Algorithmic Aspects in Information and Management (AAIM)*, volume 4041 of *Lecture Notes in Computer Science*, pages 291–302. Springer.

Choffrut, C. and Karhumäki, J. (1997). *Combinatorics of Words, in G. Rozenberg and A. Salomaa (eds), Handbook of Formal Languages*. Springer-Verlag.

Chudnovsky, M. and Seymour, P. (2005). The structure of claw-free graphs. In *Surveys in Combinatorics*, volume 327 of *London. Math. Soc. Lecture Notes*, pages 153–172. Cambridge University Press.

Chung, M.-J. (1998). More efficient algorithm for ordered tree inclusion. *Journal of Algorithms*, 26(2):370–385.

Chvátal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235.

Cieliebak, M., Eidenbenz, S., Pagourtzis, A., and Schlude, K. (2008). On the complexity of variations of equal sum subsets. *Nordic Journal of Computing*, 14(3):151–172.

Collins, M., Kempe, D., Saia, J., and Young, M. (2007). Nonnegative integral subset representations of integer sets. *Information Processing Letters*, 101(3):129–133.

Crochemore, M., Hancart, C., and Lecroq, T. (2007). *Algorithms on Strings*. Cambridge.

Crochemore, M., Hermelin, D., Landau, G., Rawitz, D., and Vialette, S. (2008). Approximating the 2-interval pattern problem. *Theoretical Computer Science*, 395(2-3):283–297. (special issue for Alberto Apostolico).

Davydov, E. and Batzoglou, S. (2006). A computational model for rna multiple structural alignment. *Theoretical Computer Science*, 368(3):205–216.

Dent, P., Yacoub, A., Fisher, P., Hagan, M., and Grant, S. (2003). MAPK pathways in radiation responses. *Oncogene*, 22:5885–5896.

Diestel, R. (2000). *Graph Theory*. Number 173 in Graduate texts in Mathematics. Springer-Verlag, second edition.

Dilworth, R. (1950). A decomposition theorem for partially ordered sets. *Ann. Math.*, 51:161–166.

Dinur, I., Guruswami, V., Khot, S., and Regev, O. (2005). A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146.

Dondi, R., Fertin, G., and Vialette, S. (2007). Weak pattern matching in colored graphs: Minimizing the number of connected components. In *Proc.10th Italian Conference on Theoretical Computer Science (ICTCS), Roma, Italy*, pages 27–38. World-Scientific.

Dondi, R., Fertin, G., and Vialette, S. (2009). Maximum motif problem in vertex-colored graphs. In Kucherov, G. and Ukkonen, E., editors, *Proc. 20th Annual Symposium on Combinatorial Pattern Matching (CPM'09), Lille, France*, volume 5577 of *Lecture Notes in Computer Science*, pages 221–235.

Dorit, R. and Gilbert, W. (1991). The limited universe of exons. *Current Opinions in Structural Biology*, 1:973–977.

Dost, B., Shlomi, T., Gupta, N., Ruppin, E., Bafna, V., and Sharan, R. (2007). QNet: A Tool for Querying Protein Interaction Networks. *RECOMB*, pages 1–15.

Downey, R. and Fellows, M. (1999). *Parameterized Complexity*. Springer-Verlag.

Dutton, R. (2009). On a graph's security number. *Discrete Mathematics*, 309:4443–4447.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. (2001). Rank aggregation methods for the web. In *Proc. of the 10th international conference on World Wide Web (WWW), Hong Kong, Hong Kong*, pages 613–622.

Eén, N. and Sörensson., N. (2006). Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26.

Ehrenfeucht, A. and Rozenberg, G. (1978). Elementary homomorphisms and a solution of the D0L sequence equivalence problem. *Theoretical Computer Science*, 7:169–183.

El-Mabrouk, N. (2005). Genome rearrangements with gene families. In *Mathematics of evolution and phylogeny*, pages 291–320. Oxford University Press.

Epping, W. H. T. and Oertel, P. (2004). Complexity results on a paint shop problem. *Discrete Applied Mathematics*, 136(2-3):217–226.

Erdős, P. and Szekeres, G. (1935). A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470.

Erdös, P. and Lovász, L. (1975). Problems and results on 3-chromatic hypergraphs and some related questions. In Hajnal, A., Rado, R., and Sós, V., editors, *Infinite and Finite Sets (Colloq., Keszthely, 1973; dedicated to P. Erdös on his 60th birthday)*, volume 2 of *Coll. Math. Soc. J. Bolyai*, pages 609–627. North-Holland, Amsterdam.

Estivill-Castro, S. and Wood, D. (1992). A survey of adaptive sorting algorithms. *ACM Computing Surveys*, 24(9):441–476.

Evans, P. (1999a). *Algorithms and complexity for annotated sequence analysis*. PhD thesis, University of Victoria.

Evans, P. (1999b). *Algorithms and Complexity for Annotated Sequences Analysis*. PhD thesis, University of Victoria.

Evans, P. and Wareham, H. (2001). Exact algorithms for computing d pairwise alignments and 3-medians from structure-annotated sequences (extended abstract). In *Proc. of the 6th Pacific Symposium on Biocomputing (PSB)*, pages 559–570.

Evans, P. A. (1999c). Finding common subsequences with arcs and pseudoknots. In Crochemore, M. and Paterson, M., editors, *Proc. 10th Annual Symposium Combinatorial Pattern Matching (CPM), Warwick University, UK*, volume 1645 of *Lecture Notes in Computer Science*, pages 270–280. Springer.

Fagnot, I., Fertin, G., and Vialette, S. (2009). On finding small 2-generating sets. In Ngo, H., editor, *Proc. 15th Annual International Conference (COCOON), Niagara Falls, NY, USA*, volume 5609 of *Lecture Notes in Computer Science*, pages 378–387. Springer.

Fagnot, I., Lelandais, G., and Vialette, S. (2004). Bounded list injective homomorphism for comparative analysis of protein-protein interaction graphs. In *Proc. 1st Algorithms and Computational Methods for Biochemical and Evolutionary Networks (CompBioNets), Recife, Brazil*, pages 45–70. KCL publications.

Fagnot, I., Lelandais, G., and Vialette, S. (2008). Bounded list injective homomorphism for comparative analysis of protein-protein interaction graphs. *Journal of Discrete Algorithms*, 6(2):178–191.

Faudree, R., Flandrin, E., and Ryjáček, Z. (1997). Claw-free graphs - a survey. *Discrete Mathematics*, 164:87–147.

Favaron, O., Fricke, G., Goddard, W., Hedetniemi, S., Hedetniemi, S., Kristiansen, P., Laskar, R., and Skaggs, R. (2004). Offensive alliances in graphs. *Discussiones Mathematicae Graph Theory*, 24(2):263–275.

Fellows, M., Fertin, G., Hermelin, D., and Vialette, S. (2007). Sharp tractability borderlines for finding connected motifs in vertex-colored graphs. In *Proc. 34th International Colloquium on Automata, Languages and Programming (ICALP), Wroclaw, Poland*, volume 4596 of *Lecture Notes in Computer Science*, pages 340–351. Springer.

Felsner, S., Müller, R., and Wernisch, L. (1997). Trapezoid graphs and generalizations: Geometry and algorithms. *Discrete Applied Math.*, 74:13–32.

Fertin, G., Hermelin, D., Rizzi, R., and Vialette, S. (2007). Common structured patterns in linear graphs: Approximations and combinatorics. In Ma, B. and Zhang, K., editors, *Proc. Symposium on Combinatorial Pattern Matching (CPM'07), London, Ontario, Canada*, volume 4580 of *Lecture Notes in Computer Science*, pages 241–252.

Fertin, G., Labarre, A., Rusu, I., Tannier, É., and Vialette, S. (2009a). *Combinatorics of Genome Rearrangements*. MIT press.

Fertin, G., Rizzi, R., and Vialette, S. (2005). Finding exact and maximum occurrences of protein complexes in protein-protein interaction graphs. In Jedrzejowicz, J. and Szepietowski, A., editors, *Proc. 30th International Symposium on Mathematical Foundations of Computer Science (MFCS), Gdansk, Poland*, volume 3618 of *Lecture Notes in Computer Science*, pages 328–339. Springer.

Fertin, G., Rizzi, R., and Vialette, S. (2009b). Finding occurrences of protein complexes in protein-protein interaction graphs. *Journal of Discrete Algorithms*, 7(1):90–101.

Fertin, G. and Vialette, S. (2009). On the s-labeling problem. In *Proc. 5th European conference on Combinatorics, Graph Theory and Applications (EuroComb), Bordeaux, France*, volume 34 of *Electronic Notes on Discrete Mathematics*, pages 273–277.

Flum, J. and Grohe, M. (2006). *Parameterized Complexity Theory*. Springer Verlag.

Fulkerson, D. and Gross, O. (1965). Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855.

Gambette, P. and Vialette, S. (2007). On restrictions of balanced 2-interval graphs. In Brandstädt, A., Kratsch, D., and Müller, H., editors, *33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'07)*, volume 4769 of *Lecture Notes in Computer Science*, pages 55–65.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A guide to the theory of NP-completeness*. W.H. Freeman, San Francisco.

Garey, M., Johnson, D., and Stockmeyer, L. (1976). Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267.

Gavin, A., Boshe, M., et al. (2002). Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 414(6868):141–147.

Goldman, D., Istrail, S., and Papadimitriou, C. (1999). Algorithmic aspects of protein structure similarity. In *Proc. 40th Annual Symposium of Foundations of Computer Science (FOCS), New York, NY, USA*, pages 512–522. IEEE Computer Society.

Golumbic, M. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York.

Gramm, J. (2004a). A polynomial-time algorithm for the matching of crossing contact-map patterns. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):171–180.

Gramm, J. (2004b). A polynomial-time algorithm for the matching of crossing contact-map patterns. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(4):171–180.

Gramm, J., Guo, J., and Niedermeier, R. (2002). Pattern matching for arc-annotated sequences. In Agrawal, M. and Seth, A., editors, *Proc. 22nd Foundations of Software Technology and Theoretical Computer Science (FSTTCS), Kanpur, India*, Lecture Notes in Computer Science, pages 182–193.

Gramm, J., Guo, J., and Niedermeier, R. (2006). Pattern matching for arc-annotated sequences. *ACM Transactions on Algorithms*, 2(1):44–65.

Griggs, J. and West, D. (1979). Extremal values of the interval number of a graph, I. *SIAM Journal on Algebraic and Discrete Methods*, 1:1–7.

Guignon, V., Chauve, C., and Hamel, S. (2005). An edit distance between rna stem-loops. In Consens, M. P. and Navarro, G., editors, *12th International Conference SPIRE*, volume 3772 of *LNCS*, pages 335–347.

Guillemot, S. (2008). Parameterized complexity and approximability of the SLCS problem. In *Proc. International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5018 of *Lecture Notes in Computer Science*, pages 115–128. Springer.

Guillemot, S. and Vialette, S. (2009). Pattern matching for 321-avoiding permutations. In Dong, Y., Du, D.-Z., and Ibarra, O., editors, *Proc. 20-th International Symposium on Algorithms and Computation (ISAAC), Hawaii, USA*, volume 5878 of *Lecture Notes in Computer Science*, page 10641073.

Guo, J. (2002). Exact algorithms for the longest common subsequence problem for arc-annotated sequences. Master's thesis, Univeristy of Tübingen.

Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., and Wernicke, S. (2006). Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396.

Gupta, U., Lee, D., and Leung, J.-T. (1982). Efficient algorithms for interval graph and circular-arc graphs. *Networks*, 12:459–467.

Gurski, F. (2008). Polynomial algorithms for protein similarity search for restricted mrna structures. *Information Processing Letters*, 105(5):170–176.

Gyárfás, A. (2003). Combinatorics of intervals, preliminary version. Institute for Mathematics and its Applications (IMA) Summer Workshop on Combinatorics and Its Applications. available online at `http://www.math.gatech.edu/news/events/ima/newag.pdf`.

Gyárfás, A. and Lehel, J. (1970). A helly-type problem in trees. In ös, P. E., Rényi, A., and Sós, V., editors, *Combinatorial Theory and its Application*, pages 571–584. North-Holland.

Gyárfás, A. and West, D. (1995). Multitrack interval graphs. *Congressus Numerantium*, 109:109–116.

Hajiaghayi, M., Jain, K., Lau, L., Mandoiu, I., Russell, A., and Vazirani, V. (2006). Minimum multicolored subgraph problem in multiplex PCR primer set selection and population haplotyping. In *Proceedings of the 6th International Conference on Computational Science (ICCS)*, pages 758–766.

Halldórsson, M. and Karlsson, R. (2006). Strip graphs: Recognition and scheduling. In Fomin, F., editor, *Proc. 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG), Bergen, Norway*, volume 4271 of *Lecture Notes in Computer Science*, pages 137–146.

Hamel, S., Blin, G., and Vialette, S. (2010). Comparing RNA structures with biologically relevant operations cannot be done without strong combinatorial restrictions. In Fujita, S. and Rahman, S., editors, *Proc. 4th Workshop on Algorithms and Computation (WALCOM'10), Dhaka, Bangladesh*, Lecture Notes in Computer Science. To appear.

Hassin, R. and Segev, D. (2005). The set cover with pairs problem. In Ramanujam, R. and .Sen, S., editors, *Proceedings of the 25th international conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), Hyderabad, India*, pages 164–176.

Heber, S. and Stoye, J. (2001). Finding all common intervals of k permutations. In Amir, A. and Landau, G., editors, *Proc. Annual Symposium on Combinatorial Pattern Matching (CPM), Jerusalem, Israel*, volume 2089 of *Lecture Notes in Computer Science*, pages 207–218. Springer.

Hein, J., Jiang, T., Wang, L., and Zhang, K. (1996). On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71:153–169.

Hell, P. and Nešetřil, J. (2004). *Graphs and Homomorphisms*. Lecture Series in Mathematics and Its Applications. Oxford University Press.

Hermelin, D., Fellows, M., Rosamond, F., and Vialette, S. (2009). On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61.

Hermelin, D., Rawitz, D., Rizzi, R., and Vialette, S. (2008). The minimum substring cover problem. *Information and Computation*, 206(11):1303–1312.

Herrbach, C. and Vialette, S. (2005). Linear graph non-crossing structural alignment under the rna stacking-pair scoring scheme. In *Proc. 2nd Algorithms and Computational Methods for Biochemical and Evolutionary Networks (CompBioNets), Lyon, France*. KCL publications.

Hirschberg, D. (1977). Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675.

Ho, Y., Gruhler, A., et al. (2002). Systematic identification of protein complexes in Saccharomyces cerevisae by mass spectrometry. *Nature*, 415(6868):180–183.

Hochbaum, D. (1982). Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556.

Huang, Y.-T., Chao, K.-M., and Chen, T. (2005). An approximation algorithm for haplotype inference by maximum parsimony. In *Proceedings of the 20-th ACM Symposium on Applied Computing (SAC)*, pages 146–150.

Huffner, F., Wernicke, S., and Zichner, T. (2007). Algorithm Engineering For Color-Coding To Facilitate Signaling Pathway Detection. In *Proceedings of the 5th Asia-Pacific Bioinformatics Conference*. Imperial College Press.

Hunt, J. and Szymanski, T. (1977a). A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20:350353.

Hunt, J. and Szymanski, T. (1977b). A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20:350–353.

Ibarra, L. (1997). Finding pattern matchings for permutations. *Information Processing Letters*, 61(6):293–295.

Ideker, T., Karp, R., Scott, J., and Sharan, R. (2006). Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology*, 13(2):133–144.

Ieong, S., Kao, M.-Y., Lam, T.-W., Sung, W.-., and Yiu, S.-. (2003). Predicting RNA secondary structures with arbitrary pseudoknots by maximizing the number of stacking pairs. *Journal of Computational Biology*, 10(6):981–995.

Jacks, T., Masiarz, M. P. F., Luciw, P., Barr, P., and Varmus, H. (1988). Characterization of ribosomal frameshifting in HIV-1 gag-pol expression. *Nature*, 331:280–283.

Jacks, T. and Varmus, H. (1985). Expression of the Rous sarcoma virus pol gene by ribosomal frameshifting. *Science*, 230:1237–1242.

Jiang, M. (2007a). A 2-approximation for the preceding-and-crossing structured 2-interval pattern problem. *Journal of Combinatorial Optimization*, 13:217–221. Special Issue on Bioinformatics.

Jiang, M. (2007b). A PTAS for the weighted 2-interval pattern problem over the preceding-and-crossing model. In Dress, A., Xu, Y., and Zhu, B., editors, *1st Annual International Conference on Combinatorial Optimization and Applications (COCOA'07), Xi'an, Shaanxi, China*, volume 4616 of *Lecture Notes in Computer Science*, pages 378–387.

Jiang, M. (2008). Approximation algorithms for predicting RNA secondary structures with arbitrary pseudoknots. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. To appear.

Jiang, M. (2010). On the parameterized complexity of some optimization problems related to multiple-interval graphs. In *Proc. 21th Annual Symposium on Combinatorial Pattern Matching (CPM), New York, USA*, volume 6129 of *Lecture Notes in Computer Science*, pages 125–137. Springer.

Jiang, T. and Li, M. (1995). On the approximation of shortest common supersequences and longest common subsequences. *SIAM Journal on Computing*, 24:1122–1139.

Jiang, T., Lin, G., Ma, B., and Zhang, K. (2000a). The longest common subsequence problem for arc-annotated sequences. In Giancarlo, R. and Sankoff, D., editors, *Proc. of the 11th annual symposium on Combinatorial Pattern Matching (CPM), Montreal, Canada*, pages 154–165.

Jiang, T., Lin, G.-H., Ma, B., and Zhang, K. (2000b). The longest common subsequence problem for arc-annotated sequences. In Giancarlo, R. and Sankoff, D., editors, *Proc. 11th Annual Symposium on Combinatorial Pattern Matching (CPM), Montreal, Canada*, volume 1848 of *Lecture Notes in Computer Science*, pages 154–165. Springer.

Johnson, D. (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278.

Kanehisa, M., Goto, S., Kawashima, S., Okuno, Y., and Hattori, M. (2004). The KEGG resource for deciphering the genome. *Nucleic acids research*, 32:277–280.

Kannan, R. (1987). Minkowskis convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440.

Kaplan, H. and Shafrir, N. (2006). The greedy algorithm for edit distance with moves. *Information Processing Letters*, 97(1):27–37.

Karger, D., Motwani, R., and Ramkumar, G. (1995). On approximating the longest path in a graph. *SIAM Journal on Computing*, 24:1122–1139.

Kelley, B., Sharan, R., Karp, R., Sittler, T., Root, D. E., Stockwell, B., and Ideker, T. (2003). Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences*, 100(20):11394–11399.

Kilpeläinen, P. and Mannila, H. (1995). Ordered and unordered tree inclusion. *SIAM J. Comp.*, 24(2):340–356.

King, A. and Reed, B. (2007). Bounding χ in terms of ω ans δ for quasi-line graphs. Article in preparation.

Knuth, D. (1973). *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading MA, 3rd edition.

Knuth, D. (1998). *Sorting and Searching, The Art of Computer Programming Vol. 3 (Second Edition)*. Addison-Wesley.

Kolman, P., Goldstein, A., and Zheng, J. (2005). Minimum common string partition problem: Hardness and approximations. *The Electronic Journal of Combinatorics*, 12(1). Paper R50.

Kostochka, A. (1988). On upper bounds on the chromatic numbers of graphs. *Transactions of the Institute of Mathematics (Siberian Branch of the Academy of Sciences in USSR)*, 10:204–226.

Kostochka, A. and Kratocvil, J. (1997). Covering and coloring polygon-circle graphs. *Discrete Mathematics*, 163:299–305.

Kristiansen, P., Hedetniemi, S., and Hedetniemi, S. (2004). Alliances in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 48:157–177.

Kubica, M., Rizzi, R., Vialette, S., and Waleń, T. (2006). Approximation of RNA multiple structural alignment. In Lewenstein, M. and Valiente, G., editors, *Proc. 17th Annual Symposium on Combinatorial Pattern Matching (CPM),Barcelona, Spain*, volume 4009 of *Lecture Notes in Computer Science*. Springer.

Lacroix, V., Fernandes, C., and Sagot, M.-F. (2006). Motif search in graphs: application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3(4):360–368.

Lelandais, G., Crom, S. L., Devaux, F., Vialette, S., Church, G., Jacq, C., and Marc, P. (2004a). yMGV: a cross-species expression data mining tool. *Nucl. Acids. Res.*, 32:D323–D325.

Lelandais, G., Marc, P., Vincens, P., Jacq, C., and Vialette, S. (2004b). MiCoViTo: a tool for gene-centric comparison and visualization of yeast transcriptome states. *BMC Bioinformatics*, 5(20).

Lelandais, G., Vincens, P., Badel-Chagnon, A., Vialette, S., Jacq, C., and Hazout, S. (2006). Comparing gene expression networks in multi-dimensional space to extract similarities and differences between organisms. *Bioinformatics*, 22(11):1359–1366.

Lenstra, H. (1983). Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548.

Lerat, E., Daubin, V., and Moran, N. (2003). From gene trees to organismal phylogeny in prokaryotes: the case of the γ-proteobacteria. *PLoS biology*, 1(1):101–109.

Li, S. and Li, M. (2006). On the complexity of the crossing contact map pattern matching problem. In *Proc. Proc. of 6th Workshop on Algorithms in Bioinformatics (WABI)*, volume 4175 of *Lecture Notes in Computer Science*, pages 231–241.

Li, S. and Li, M. (2009a). On two open problems of 2-interval patterns. *Theoretical Computer Science*, 410(24-25):2410–2423.

Li, S. and Li, M. (2009b). On two open problems of 2-interval patterns. *Theoretical Computer Science*, 410(24-25):2410–2423.

Lin, G., Chen, Z.-Z., Jiang, T., and Wen, J. (2002). The longest common subsequence problem for sequences with nested arc annotations. *Journal of Computer and System Sciences*, 65(3):465–480. Special issue on computational biology.

Lovász, L. (1974). On the ratio of optimal integeral and fractional solutions. *Discrete Mathematics*, 13:383–390.

Lozano, A. and Valiente, G. (2004). On the maximum common embedded subtree problem for ordered trees. In Iliopoulos, C. and Lecroq, T., editors, *String Algorithmics*, chapter 7. King's College London Publications.

Lyngsø R. and Pedersen, C. (2000). RNA pseudoknot prediction in energy-based models. *Journal of Computational Biology*, 7(3-4):409–427.

Marcus, A. and Tardos, G. (2004). Excluded permutation matrices and the Stanley-Wilf conjecture. *Journal of Combinatorial Series A*, 107(1):153–160.

Margeot, E., Blugeon, C., Sylvestre, J., Vialette, S., Jacq, C., and Corral-Debrinski, M. (2002). In *saccharomyces cerevisae*, ATP2 mRNA sorting to the vicinity of mitochondria is essential for respiratory function. *EMBO Journal*, 21(24):6893–6904.

McConnell, R. and Spinrad, J. (1999). Modular decomposition and transitive orientation. *Discrete Mathematics*, 201:189–241.

McKee, T. and McMorris, F. (1999). *Topics in intersection graph theory*. SIAM monographs on discrete mathematics and applications.

Micali, S. and Vazirani, V. (1980). An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proc. 21st Annual Symposium on Foundation of Computer Science (FOCS)*, pages 17–27. IEEE.

Moser, L. (1960). On the representation of $1, 2, \ldots, n$ by sums. *Acta Arith.*, 6:11–13.

Néraud, J. (1990). Elementariness of a finite set of words is co-NP-complete. *Theoretical Informatics and Applications*, 24(5):459–470.

Nguyen, C. (2005). Algorithms for calculating exemplar distances. Technical report, National University of Singapore. Honours Year Project.

Niedermeier, R. (2006). *Invitation to Fixed Parameter Algorithms*. Lecture Series in Mathematics and Its Applications. Oxford University Press.

Ohno, S. (1970). *Evolution by gene duplication*. Springer-Verlag.

P. Thébault, S. d. G., Schiex, T., and Gaspin, C. (2006). Searching rna motifs and their intermolecular contacts with constraint networks. *Bioinformatics*, 22(17):2074–2080.

Patthy, L. (1991). Exons - original building blocks of proteins? *BioEssays*, 13(4):187–192.

Pellegrini, M., Marcotte, E., Thompson, M., Eisenberg, D., and Yeates, T. (1999). Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *PNAS*, 96(8):4285–4288.

Pereira-Leal, J., Enright, A., and Ouzounis, C. (2004). Detection of functional modules from protein interaction networks. *Proteins*, 54(1):49–57.

Pinter, R., Rokhlenko, O., Yeger-Lotem, E., and Ziv-Ukelson, M. (2005). Alignment of metabolic pathways. *Bioinformatics*, 21(16):3401–3408.

Raz, R. and Safra, S. (1997). A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on the Theory Of Computing (STOC)*, pages 475–484.

Robertson, N. and Seymour, P. (1986). Graph minors. II. Algorithmic aspects of tree-width. *SIAM Journal of Algorithms*, 7:309–322.

Rodríguez-Velázquez, J. A. and Sigarreta, J. (2009). Global defensive k-alliances in graphs. *Discrete Applied Mathematics*, 157(2):211–218.

Rossin, D. and Bouvel, M. (2006). The longest common pattern problem for two permutations. *Pure Mathematics and Applications*, 17:55–69.

Rozenberg, G. and Salomaa, A. (1980). *The Mathematical Theory of L Systems*. Academic Press, New York.

Sadique Adi, S., Braga, M., Fernandes, C., Eduardo Ferreira, C., Viduani Martinez, F., M.-F Sagot, M. S., Tjandraatmadja, C., and Wakabayashi, Y. (2008). Repetition-free longest common subsequence. *Electronic Notes in Discrete Mathematics*, 30:243–248.

Sankoff, D. (1999). Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917.

Sankoff, D. and Haque, L. (2005). Power boosts for cluster tests. In *Proc. 3rd RECOMB Comparative Genomics Satellite Workshop, Dublin, Ireland*, volume 3678 of *Lecture Notes in Bioinformatics*, pages 11–20.

Scheinerman, E. and West, D. (1983). The interval number of a planar graph: three intervals suffice. *Journal of Combinatorial Theory - Series B*, 35:224–239.

Sharan, R. and Ideker, T. (2006). Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, 24:427–433.

Sharan, R., Ideker, T., Kelley, B., Shamir, R., and Karp, R. (2004). Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. In *Proc. 8th annual international conference on Computational molecular biology (RECOMB), San Diego, California, USA*, pages 282–289. ACM Press.

Sharan, R., Suthram, S., Kelley, R., Kuhn, T., McCuine, S., Uetz, P., Sittler, T., Karp, R., and Ideker, T. (2005). Conserved patterns of protein interaction in multiple species. *Proc. Natl Acad. Sci. USA*, 102(6):1974–1979.

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

Shen-Orr, S., Milo, R., Mangan, S., and Alon, U. (2002). Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31(1):64–68.

Shlomi, T., Segal, D., Ruppin, E., and Sharan, R. (2006). QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics*, 7:199.

Stanley, R. (1999). *Enumerative Combinatorics*, volume 2. Cambridge University Press, Cambridge.

Stöhr, A. (1955a). Gelöste und ungelöste fragen über basen der natürlichen zahlenreihe, i. *J. reine Angew. Math.*, 194:40–65.

Stöhr, A. (1955b). Gelöste und ungelöste fragen über basen der natürlichen zahlenreihe, ii. *J. reine Angew. Math.*, 194:111–140.

Sylvestre, J., Vialette, S., Corral-Debrinski, M., and Jacq, C. (2003). Long mRNA coding for yeast mitochondrial proteins of prokaryotic origin preferentially localize to the vicinity of mitochondria. *Genome Biology*, 4(7):1–9.

Tang, J. and Moret, B. (2003). Phylogenetic reconstruction from gene-rearrangement data with unequal gene content. In Dehne, F., Sack, J.-R., and Smid, M., editors, *Proc. 8-th International Workshop on Algorithms and Data Structures (WADS), Ottawa, Ontario, Canada*, volume 2748 of *Lecture Notes in Computer Science*, pages 37–46. Springer.

Tarjan, R. and Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13:566–579.

Thomasse, S. (2009). A quadratic kernel for feedback vertex set. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*.

Tichy, W. (1984). The string-to-string correction problem with block moves. *ACM Transactions on Computer Systems*, 2(4):309–321.

Tiskin, A. (2006). Longest common subsequences in permutations and maximum cliques in circle graphs. In Lewenstein, M. and Valiente, G., editors, *Proc. 17th Combinatorial Pattern Matching (CPM), Barcelona, Spain*, volume 4009 of *Lecture Notes in Computer Science*, pages 270–281.

Titz, B., Schlesner, M., and Uetz, P. (2004). What do we learn from high-throughput protein interaction data? *Expert Review of Anticancer Therapy*, 1(1):111–121.

Tripathi, A. (2006). A note on the postage stamp problem. *Journal of Integer Sequences*, 9. 06.013.

Trotter, W. and Harary, F. (1979). On double and multiple interval graphs. *J. Graph Theory*, 3:205–211.

Uetz, P., Giot, L., et al. (2000). A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisae. *Nature*, 403(6770):623–627.

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

Vazirani, V. (2002). *Approximation Algorithms*. Springer. 1st ed. 2001. Corr. 2nd printing, 2002.

Vialette, S. (2004). On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science*, 312(2-3):223–249.

Vialette, S. (2006). Packing of $(0, 1)$-matrices. *Theoretical Informatics and Applications RAIRO*, 40(4):519–536.

Vialette, S. (2008). Two-interval pattern problems. In Kao, M.-Y., editor, *Encyclopedia of Algorithms*, pages 985–989. Springer.

Waterman, M. (1995). *Introduction to computational biology - Maps, sequences and genomes*. Chapman and Hall, London.

Wernicke, S. (2006). Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):347–359.

West, D. and Shmoys, D. (1984). Recognizing graphs with fixed interval number is NP-complete. *Discrete Applied Mathematics*, 8:295–305.

Whang, M.-S. and Wang, G.-H. (1992). Efficient algorithms for the maximum weight clique and maximum weight independent set problems on permutation graphs. *Information Processing Letters*, 43:293–295.

Xu, J., Jiao, F., and Berger, B. (2007). A parameterized algorithm for protein structure alignment. *Journal of Computational Biology*, 14(5):564–577.

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