Université de Bordeaux 1
Laboratoire Bordelais de Recherche en Informatique

Habilitation à diriger des Recherches
Specialité Informatique

par

Pascal Ferraro

Analysis of Tree Structures : From Computational Biology to Music Analysis

Habilitation à Diriger des Recherches défendue le 6 Novembre 2009
devant le jury composé de :

Mme	Maylis Delest	Pr. Université de Bordeaux 1	(Présidente)
M.	Serge Dulucq	Pr. IUT de Bordeaux1	(Directeur de Recherche)
M.	Abraham Escobar	Pr. ESA d'Angers	(Rapporteur)
M.	Guillaume Fertin	Pr. Université de Nantes	(Rapporteur)
M.	Costas Iliopoulos	Pr. King's College of London	
M.	Laurent Imbert	C.R. 1 CNRS	
M.	David Sankoff	Pr. University of Ottawa	(Rapporteur)

À Louis, Cécile et Claire...

Acknowledgments

Acknowledgements are too important to be expressed in a language that is not well mastered. For this reason, I have chosen to write them in French.

Les travaux de recherche présentés dans cette Habilitation sont avant tout un travail d'équipe, si j'en suis l'auteur principal, avant d'en commencer l'exposé, je tiens à remercier celles et ceux qui y ont apporté leur contribution.

Je tiens en premier lieu à exprimer ma gratitude aux différentes institutions qui m'ont accueilli dans leurs locaux durant les 12 dernières années, parmi celles-ci : l'Unité Mixte de Recherche CIRAD/INRA Modélisation des Plantes à Montpellier, le Laboratoire Bordelais de Recherche en Informatique (LaBRI) et le *Department of Computer Science* de l'Université de Calgary (Canada).

Parmi les différents directeurs de ces laboratoires , je souhaite chaleureusement remercier Serge Dulucq, Professeur de l'IUT de Bordeaux qui en tant que directeur du LaBRI m'a fait confiance en permettant mon recrutement à Bordeaux, a toujours soutenu sans réserve mes travaux et a accepté de diriger mes recherches. Je lui dois en grande partie le début de ma carrière de Maître de Conférences et je ne saurai lui en être plus reconnaissant.

Je tiens à associer à ces remerciements Maylis Delest, Professeur de l'Université de Bordeaux, qui à la direction du LaBRI durant les 4 dernières années, a permis l'émergeance du projet Simbals, cœur de cette Habilitation. Je la remercie également de m'avoir a fait l'honneur de présider mon jury d'Habilitation.

C'est lors de la conférence de Bioinformatique JOBIM à Montréal en 2004 que j'ai fait la connaissance du Professeur David Sankoff. L'étendue et la variété de ces travaux dans le domaine de l'analyse de données structurées que ce soit sous forme de séquences ou d'arbres font référence dans la communauté. Il est de plus le premier à avoir "détourné" les algorithmes de bioinformatique de leur essence première pour les appliquer à la musique,ce qui a inspiré en grande partie les travaux que je présente ici. Je suis extrèmenent honoré qu'il ait accepté de rapporter mon mémoire. Je le remercie de plus d'avoir vaincu les affres de la technologie et le décalage horaire pour lui permettre de participer à ma soutenance et mon jury d'Habilitation.

J'ai rencontré le Professeur Abraham Escobar lors de l'un de mes nombreux séjours à l'Université de Calgary. Nos discussions autour des méthodes de modélisation des plantes et de leur évaluation ont renforcé mon sentiment sur la nécessité de les développer. Je suis très honoré qu'il ait accepté d'être rapporteur de mon Habilitation et qu'il ait pris sur son temps pour venir participer à mon jury.

J'ai régulièrement croisé la route du Professeur Guillaume Fertin, de part ses travaux

et l'intérêt qu'il a bien voulu porter aux miens, en m'invitant en séminaire à l'Université Nantes ou en acceptant de rapporter cette Habilitation. Je l'en remercie sincéremment.

C'est lors d'une invitation au King's College de Londres que j'ai rencontré le Professeur Costas Illiopoulos. Les échanges que nous avons eues à cette occasion autour de la comparaison des séquences musicales et de leurs structurations ont permis de lancer une indéniable collaboration entre nos équipes respectives. J'ai particulièrement apprécié qu'il puisse se libérer en une période notoirement chargée pour venir partiper à mon jury.

Avec Laurent Imbert, nous étions tous deux post-doctorants dans la même Université de Calgary en 2001. Son parcours a été bien plus rapide que le mien depuis. Bien que nos travaux n'étaient a priori pas destinés à se croiser, il m'a initié aux délices de l'implémentation sur cartes graphiques et m'a montré son importance pour le développement d'applications à destination de l'utilisateur. C'est avec beaucoup d'amitié que je le compte parmi les membres de mon jury.

La rencontre avec Christophe Godin, directeur de Recherche à l'INRIA, a été essentielle dans mes débuts d'enseignant-chercheur. Il fut mon directeur durant mes trois années de thèse, mais depuis, sa disponibilité permanente pour m'orienter dans mes choix de carrière, la passion et l'enthousiasme qui l'animent dans ses activités de recherche, son sens aigu du perfectionnisme ont continué de me guider dans mes travaux. Je veux lui témoigner, ici, ma plus sincère gratitude et mes plus chaleureux remerciements pour toutes les discussions que nous avons eues et son exigence constante.

Il y a bientôt 10 ans, accompagné de ma petite famille, j'ai atterri à Calgary en plein hiver Canadien, pour effectuer une année post-doctorale auprès du Computer Science Departement du Professeur Przemek Prusinkiewicz. Bien que j'ai par la suite obtenu un poste au sein du LaBRI, je crois que je n'ai depuis ce premier séjour jamais complètement quitté Calgary. Przemek Prusinkiewcz m'a communiqué sa passion pour la modélisation des plantes. Son émerveillement constant devant "l'Algorithmique Beauté des Plantes" m'ont depuis beaucoup stimulé et ont inspiré une grande partie de ce document. Je souhaite au travers de ces quelques lignes lui exprimer toute ma gratitude pour m'avoir accueilli à maintes reprises au sein de son équipe et de m'avoir donné l'opportunité de participer à ses travaux.

Dans cette histoire Pierre Hanna tient une place toute particulière. Lors d'une après-midi de septembre 2005, un jeune Maître de Conférences tout juste recruté, a trouvé une place dans mon bureau. De nos innombrables discussions sur les résultats de football puis sur nos travaux de recherche respectifs a priori antagosnistes, est née l'idée du projet Simbals et celle de developper ce lien entre la Bioinformatique et la Musique. Cette rencontre a marqué un tournant dans mes travaux de recherche pour me mener au contenu de cette Habilitation et je ne lui en serai jamais suffisament reconnaissant. Par la suite, Matthias Robine et Julien Allali se sont joints à notre

démarche. Nos échanges parfois vifs et contradictoires (selon le tempérament et l'humeur des interlocuteurs ...) a permis de confronter nos théories, nos idées et de faire progresser le projet. Je tenais à leur témoigner ici ma plus sincère amitié pour leur enthousiasme quotidien. Je souhaite de tout cœur que nos chemins continuent encore longtemps ensemble.

Ces remerciements ne seraient complets s'ils n'incluaient, parmi les personnes avec qui j'ai travaillé, celles qui ont été essentielles pour les développements logiciels présentés ici. Aida Ouangraoua a été une remarquable étudiante qui a marqué le projet TreeMatching. J'ai été particulièrement chanceux de l'avoir comme première doctorante. Je remercie aussi tous les étudiants de Master ou de DEA qui ont participé: Omer, Sylvain, Pierre, Ione, Anne-Laure, Thomas.

Je tiens également à remercier ceux qui m'ont chaleureusement accueilli dans leurs équipes ou leurs projets et qui font de l'activité de recherche un véritable plaisir: Yann Guédon, Frédéric Boudon, Evelynes Costes, Vincent Segura, Brendan Lane, Adam Runions, Cédric Chauve, Spiros Michalkopoulos, Pavlos Antoniou, et tous ceux que j'ai oublié mais qui savent ce que je leur dois.

En écrivant ces quelques lignes, je réalise que je mets le point final à mon parcours universitaire. Pour tous les lecteurs qui s'arrêteront à la lecture de ces remerciements, et je sais qu'ils sont nombreux (si, si, je vous ai reconnus), j'aurai souhaité faire preuve d'originalité ou d'humour. Hélas, ou plutôt heureusement, je ne dérogerai finalement pas à la routine des thèses et autres HDR, et terminerai mes remerciements par ceux qui me sont les plus proches. Ironiquement, j'ai réalisé ma recherche principalement sur 3 sites, et chacun de ceux-ci a vu la naissance de l'un de mes enfants. Je remercie Louis, Cécile et Claire pour leur patience et je devine l'effroi d'Elise en imaginant que je puisse prendre un poste dans un nouveau laboratoire. Elise a joué un rôle prépondérant dans mon épanouissement professionnel. Elle a accepté à maintes reprises de mettre entre parenthèses sa propre carrière professionnelle pour me suivre dans un pays où l'hiver dure plus de 6 mois et me permettre de continuer à vivre mon métier pleinement. Elle a subi régulièrement mes nuits blanches pour me permettre de respecter les dead-lines, mes retards ou mes voyages à l'autre bout du monde et ma mauvaise humeur qui en résultait parfois. Avec tout mon amour, je veux la remercier pour sa patience, son soutien, sa disponibilité de tous les instants et ses encouragements.

Contents

Introduction

1

1.1 Introduction

This document present a synthetic point of view of my research for the last ten years. However, it is often difficult, when you work on various subjects to write a synthesis both complete and coherent. This dissertation, as every habilitation manuscript, tells a story. And this story, as usual, can be told from different standpoints.

1.1.1 From an Application Perspective

The increasingly number of databases describing structured biological data (RNA secondary structures, gene sequences, plant architecture, *etc.*) generates a need for new investigational tools.

Recent development in computer science and applied mathematics has allowed me to define new methods integrating simultaneously structure and dynamic of development. I have thus aimed at analyzing structures and their developments at different scales in order to answer two different goals:

- identifying regularities and gradients within biological structures in order to deduce interpretations in terms of function,
- integrating the structure within a biological analysis.

In this context, the development of methods to investigate plant architecture and genomic structures takes an important place. For instance, in plant modeling, the structural comparison of plants allows to give a different point of view on the evaluation of models to simulate plant growth.

1.1.1.1 Plant Modeling

For more than 10 years now, in collaboration with biologists from INRA and bioinformaticians from Cirad and INRIA, I aim at analysing the development of plant structure at a macroscopic scale in order to answer to real biological problems. Our approach was based on the three following methods and models:

- Algorithms of tree alignment, allowing to define measures between plants,
- Hidden tree Markov models, allowing to identify ruptures and homogeneous zones within measured plants,
- branching process with dependency, describing the generative aspect in plant growth.

I have mainly focused my research in the first item. Our methodology is comparable to the one developed in the context of genomic data analysis. Indeed, in fruit tree species, for example, understanding genetic determinism of architectural traits is considered as a promising manner to control vegetative development and yield regularity via the integration of these traits in selection schemes. Within this context, we aimed at (i) analysing genetic parameters of architectural traits on segregating progenies and (ii) classifying plants on the basis of their architectural traits.

In parallel, I concentrate my research on the identification of repetitions in plants. Actually, in term of applications, we have intended to formalize the connection between macroscopic observations and microscopic, mostly invisible, processes. Apical meristems are small embryonic regions, located at the tip of plant axes, that build up plant organs by cellular division. The production of the meristems depends on their internal physical, physiological and genetic state and is controlled by contextual factors (like micro-environment, availability of nutrients, *etc.*). In principle, the number of variables that may be used to define the state of a meristem, taking account the nature and the concentrations of molecules in each cell, their position, the physical stresses at each point, the geometry of cells, their genetic contents, *etc.*, is infinitely large. Due to this intrinsic complexity, and to the current lack of hindsight on processes at such small scales, the connection between a meristem state, its micro-environment and what it produces at varying time scales seems until now largely out of reach.

However, the remarkable organization of plants at macroscopic scales makes the situation not so hopeless. The fact that plants are made up of the repetition of

many similar components, at different scales, provides macroscopic evidence for regularities and similarities in processes that drive meristem activity at microscopic scales.

1.1.1.2 RNA Secondary Structures

In the second part of my research (mainly from 2002), I focused in the development of tools to compare RNA Secondary Structures.

The biological function of a RNA molecule is very closely related to its spatial structure (tertiary structure), and two rather different RNA sequences may lead to very similar structures. This is why the classical sequence comparison tools (developed since the early 70's) are not relevant for RNA molecules. From the algorithmic point of view, the pairwise structure comparison problem has been a very active research area during the last 20 years. Basically, the problem is as follows : the RNA structure is modelled as a graph whose nodes are the nucleotides and whose edges are the chemical bonds between them. A set of basic operations on these graphs is given, with a score for each operation. Then there are two major ways to compare two structures S_1 and S_2 : edition and alignment. The edition problem consists in finding a best-scoring sequence of operations which changes S_1 into S_2. The alignment problem consists in finding a "superstructure" S_3 which contains S_1 and S_2 as substructures, in such a way to maximise the sum of the edition scores between S_1 an S_3 and between S_2 and S_3. The algorithmic complexity of these problems mainly depends on:

- the model of RNA structure that is taken into account. Briefly, the graph which represents the whole structure is called the tertiary structure of the molecule. A secondary structure is a partial representation of the tertiary structure where only edges corresponding to the strongest chemical bounds (Watson-Crick pairs, and possibly wobble pairs) are present. If a secondary structure does not contain crossing edges, so-called pseudo-knots, then it can be represented by a rooted ordered tree.
- the set of operations that are allowed. Several authors have defined a set of biologically relevant edit operations on bases and base-pairs in RNA structures: the classical substitution, insertion, and suppression operations, plus some very important operations for RNA, such as pairing or unpairing nucleotides.

In its full generality, the RNA structure comparison problem is difficult: comparing tertiary structures with the above complete set of operations has been proved to be NP-complete for the edition problem, and very recently for the alignment problem. Meanwhile, during the last 20 years much effort has been done on designing polynomial comparison methods for secondary structures without pseudo-knots, either by heuristics approaches, or by restricting the set of allowed operations, or

by fixing constraints on their scores. Some heuristics have also been developed for pseudo-knotted structures. Based on some of these works, I have developed with my students a few software to compare RNA structures at different scales of decomposition. We aimed to provide new efficient, biologically relevant and user-friendly software of RNA secondary structure comparison to the scientific community.

1.1.1.3 Music Analysis

Although music analysis is apparently not directly related to bioinformatics, I have spent the last few years developing new methods to analyse symbolic music. Symbolic music comparison has been actually introduced in the early 1990 based on concepts from the theory of sequence comparison and thus from ideas developed in bioinformatics.

Nowadays, the number of audio documents available on the Internet highly increasing, new methods for browsing, retrieving or classifying have to be offered to users. The growing Music Information Retrieval (MIR) research community identifies and explicates the problems induced by these new methods. One of the key problems of this research area is the estimation of the musical similarity between audio data. In collaboration, with Pierre Hanna (from the LaBRI), we have thus proposed a new project (started in 2006) to extend methods developed in biological context to evaluate, and more generally, to analyze music. Measuring similarity between sequences is a well-known problem and its solutions are highly related to the one proposed in computational biology.

However, musical sequences are characterized by specific properties. Thus, developing efficient and accurate algorithms implies to take into account areas such as sound perception and music theory. This emerging research area thus requires interdisciplinary collaborations between experts in the field of audio signal processing, algorithmic, pattern recognition, computer music, music theory, etc.

1.1.2 Data Structure Organization

Applications have guided my research from the beginning. However another way to present my work is to characterize the data structures investigated and the tools developed for their studies. Looking back on my work, these data structures can be organized in function of their structural complexity.

1.1.2.1 Sequences

Bio-molecular sequence comparison is the origin of bio-informatics. Today, powerful sequence comparison methods, together with comprehensive biological databases,

have changed the practice of molecular biology and genomics. In the mean-time sequence analysis seems to be a promising tool in music applications.

Alignment and sequence edition constitute two of the main processes commonly used to compare sequences. They allow to visualize the resemblance between strings. Computations are usually realized using the dynamic programming principle. I have dealt, during the last few years, in collaboration with Costas Iliopoulos from King's College in London, with extension of comparison methods. These extensions were mainly focused to application to music analysis. However my research has been clearly oriented by the analysis of tree-structured data.

1.1.2.2 Tree-Structured Data and Multi-scale Tree Graphs

Tree structure data analysis is a standard operation for which there are well-known utilities. In particular, trees are widely used structures for coding data in biology. A huge number of projects have been proposed to develop method for analysing these biological tree structured data.

A first set of approaches that makes it possible to compare quantitatively the structure of two trees have been developed. These approaches are based on the use of an edit-distance approach, in which a metric is defined, that reflects the minimum number of elementary edit operations necessary to transform one tree into the other. These algorithms solve different tree-to-tree comparison problems, such as defining a metric between trees, finding whether a tree is included into another one, finding the consensus tree between two trees (*i.e.* the minimal tree that contains both), or the maximum common subtree, *etc.* Usually, to answer each question, a whole family of algorithms is developed to account for different characteristics of either the input trees (ordering of nodes, labeling) or the comparison problem (constraints on the valid edit operations, *etc.*). Based on algorithms used to compare strings, I have studied how to use and adapt such algorithms to compare tree structures from a structural point of view. This notions have been furthermore generalized to multi-scale tree graphs, in order to answer more precisely to biological problem.

In the meantime, I was interested in identifying repetitions into tree-structures. This notion is closely related to the definition of self-similarity in trees. While self-similarity usually refers to purely geometric properties of objects (parts of an object are geometrically similar to the entire object up to a scaling factor), structural self-similarity attempts to capture an equivalent idea for structures and graphs. Different approaches of structural self-similarities have been tentatively proposed. Authors defined self-similarity by analyzing either global branching parameter of trees or topological structural properties of graphs. We have introduced, with Christophe Godin from the INRIA and Przemyslaw Prusinkiewicz from the University of Calgary, the notion of structural self-similarity in the context of plant modeling. Our approach to structural self-similarity in plants used edit-distance metrics to find recursively similarities between the high order branches of a plant and its trunk.

1.1.2.3 DAGS

Data compression is widely used in different domain of applications. The data to be compressed is often stored within a structure of some special form, *e.g.* list, tree, graph, etc. Plain data is usually worthless or at least of little value outside its context. Thus, besides the data itself, we must also store and compress if possible the structure in which the data is stored. In this part of my research I supposed that the data structure to be compressed and then manipulated is a tree.

During the comparison of trees, introduced previously, all the studies focus on the comparison between two different trees and usually pay no attention on the characterization of the internal structure of a tree. In a different spirit, I have addressed the problem of studying internal repetitions of structures in a tree by eliminating the structural redundancy appearing in trees (or in graphs). For this, similar parts in a tree are condensed, resulting in a directed acyclic graph (DAG). Such an approach was used in different domains. In collaboration with Christophe Godin, we have depicted an algorithm that reduces these function DAGs to canonical DAGs from which all the structural redundancy of the initial DAG has been removed. It is closely related to the algorithm described for testing whether two trees are isomorphic or not.

1.2 Organization of the Document

These two points of view of my work appear in this document. However, since my research was always led by the applications, I deliberately chose to organize my Habilitation following these applications.

The next chapter details my curriculum vitæ : my research activities since 1996 are given, then my teaching activities and student supervision, my scientific collaborations are reviewed, my administrative tasks are roughly sketched and this chapter is closed by the list of all my publications so far.

Chapters 3 to 5 intended to be decomposed in the same way. I first introduce the representation of the object studied, then I present some combinatorics methods to analyse them. The final sections are dedicated to particular results in the considered application field. These Chapters end up with a section presenting a selection of papers related to them.

Thus, Chapter 3 motivates the work that I have done in the field of plant modelling and analysis. I first introduced a formal representation of plant architecture commonly used in the literature. This formal representation is then used to define methods for comparing plants, mainly based on the computation of a distance between unordered trees and quotiented trees. We have extended these methods to compute a distance between semi-ordered trees which are more suitable to plant

structure representation. The Chapter ends up with the notions of self-similarity and compression of plants that I have introduced in the last few years.

Researches in Bio-informatics of RNA Secondary Structures are presented in Chapter IV. This Chapter starts with some considerations about the representation of RNA secondary structures as trees. Extensions of ordered tree graph comparison methods are then introduced. In particular, a local distance between a quotiented representation of RNA secondary structures is detailed. Finally an evaluation of the methods between comparison methods is presented at this end of the Chapter. This analysis has been done in the context of the Brasero project.

The last part of my work fall down in the field of music analysis and is presented in Chapter V. I introduce the representation of symbolic music that we are currently using in order to compare two musics. The general method and their extensions are presented in Section 2 of this Chapter. Applications of music comparison are numerous, I have deliberately chosen to present the automatic identification of structure since it is the most "algorithmic" application. Finally I present some future works in Chapter VI.

Curriculum Vitae

2

Personal Informations

Ferraro Charles-Pascal
Associate Professor, section 27 since sept. 2002
LaBRI - University of Bordeaux 1
Since sept. 2008, in a leave of absence with the CNRS at the Pacific Institute for Math Sciences
37 years, Married, 3 children
French Citizenships

Computer Science Department,
University of Calgary
2500, University Drive NW,
Calgary, AB, T2N 1N4, Canada
ph. : (403) 220 5139
ferraro@cpsc.ucalgary.ca
http//www.labri.fr/∼ferraro

Education

2009 Habilitation Degree in Computer Science - University of Bordeaux 1. *Title*: Analysis of Tree Structures. From Computational Biology to Music Analysis
▷ *Reviewers: A. Escobar (ESA Angers), G. Fertin (U. Nantes), and D. Sankoff (U. Ottawa)*
▷ *Commitee Members: M. Delest (U. Bordeaux 1), S. Dulucq (U. Bordeaux), C. Iliopoulos (King's College of London), L. Imbert (CNRS)*

1997-2000 Phd Degree in Computer Science - Toulouse Intitute of Technology (INPT). *Title*: Algorithmic methods for comparing trees. Application to the topological comparison of plant structures.
▷ *Supervisor: C. Godin (CIRAD) and A. Ayache (ENSEEIHT)*
▷ *Reviewers : P. Prusinkiewicz (U. Calgary) et M. Habib (U. Montpellier) and Commitee Members: F. Houllier (INRA) et L. Miclet (ENSSAT Rennes)*

1995-1996 Master Degree in Computer Science - National School of Electricity, Electrotechnic, Hydraulic and Informatic Engineering - ENSEEIHT. *Title*: Development of a 3D graphic library based on PHIGS and MOTIFS. Applications to spatial mechanic.
▷ *Supervisor: Jean-François Goester (CNES) and Bernard Thiesse (ENSEEIHT)*

1993-1996 Engineer Degree in Computer Science ENSEEIHT - Toulouse. *Option*: Applied Mathematic and Computer Science

Employment

Sept. 2002 (Current)	Associate Professor (Maître de Conférences) ▷ *University of Bordeaux 1*
Jan. 2002 (9 months)	Post-Doctoral fellowship - Académie des Sciences ▷ *Plant Modeling Program of CIRAD - Montpellier*
Jan. 2001 (1 year)	Post-Doctoral fellowship - INRIA ▷ *Computer Science Department - University of Calgary (Canada)*
Sept. 1997 (3 years)	Phd Student - MNESR ▷ *Plant Modeling Program of CIRAD - Montpellier*

Administrative Tasks

Local scientific responsibility

▷ *Head of the Biological Structure Analysis group in LaBRI (since 2007)*
▷ *Member of the scientific comittee of the LaBRI (since 2007)*
▷ *Webmaster of the LaBRI website (since 2005)*
▷ *Elected member of the "commission de spécialistes section CNU 27" of LaBRI (2006-2007). This committee is in charge of recruiting the new associate professors in Computer Science.*

Project involvement (detailled in long version)

▷ *Committee Member (local coordinator) of the coordination group of ACI Arborescences project (2004-2007)*
▷ *Committee Member (local coordinator) of the coordination group of ARN project Brasero (2006-2010)*
▷ *Committee Member of the coordination group of ARN project SIMBALS (2007-2010)*

Conference Organisation

▷ *Organizing comittee member and program comittee member of the 4th International Workshop on Structure/Function Plant Models (June 2004, Montpellier)*
▷ *Organizing comittee member of JOBIM (June 2006, Bordeaux)*
▷ *Organizing comittee member of JOBIM (July 2008, Lille)*

Article reviewer

▷ *International journal (New Phytologists, Journal of Theoretical Biology, Pattern Recognition Letters),*
▷ *International conferences (STACS and FSPM),*
▷ *National conferences (JOBIM).*

Teaching Activities

Teaching responsability

▷ *Head of the Bioinformatic speciality of the Master of Informatic - University of Bordeaux (2003-2007)*

Head of Teaching Units

▷ *Computer Architecture (Licence 2, 2008),*

▷ *Probabilities and Statistics for biologists (DESS in BioInformatic and then Master 1 in BioInformatic, 2002-2008),*

▷ *Algorithmic 2 : analysis of genomic sequences (DESS in BioInformatic and then Master 2 in BioInformatic, 2002-2008),*

▷ *Algorithm for Bioinformatic (Master 2 in Computer Science, 2002-2008)*

Teaching assistant task

▷ *Initiation to Computer Science (DEUG 1 , 2002-2004),*

▷ *Algorithmic 1 (DEUG 2 and then Licence 1, 2002-2006),*

▷ *Programmation 1 (Licence 2, 2003-2007),*

▷ *Project in programming development (Master 1, 2005-2008),*

▷ *Unix systems (DEUG, and then Licence 1, 2002-2004),*

▷ *Object Oriented Programming (Miage, 2005 and Master 1, 2006).*

2.1 Research Activities

Since September 2002, I am Associate Professor in the Laboratory of Computer Science of Bordeaux (LaBRI) in the team Models and algorithms for Bio-informatics and information Visualisation (MaBioVis).

This team gathers biologists and computer scientists who develop methods for modeling in biology. I am particularly interested in the conception of generic methodologies for measuring and analysing biological objects, from plant architectures to secondary structures of RNA.

2.1.1 Projet de recherche

For the last few years, the expansion of database describing biological objects gave rise to the need of new automatic tools for browing and analysis these database. My initial works on plant architecture comparison have initiated new algorithmic technics that take into account the structural nature of data. Biological applications lead us to develop our researches toward two different axes:

- **Theoretical axis**. We aim at developping data structures and the corresponding algorithms to browse biological database in which data are strongly structured (*e.g.* sequences, trees, graphs, multiscale graphs). The developped tools are combinatoric or statistical methods to caracterize (evaluation of self-similarity degree), methods for comparings (comparison between trees or multi-scale trees) or to visualise biologocal objects.
- **Application axis**. Plant modeling is a privilegied domain of application. However, the generecity of methods developped in the former axis allows us to apply the methods to any others biological field or to any others domains (like the music analysis) where data are well structured.

The challenge is to build up a set of methods and tools allowing the biologists to quantitatively analyse strongly structured biological objects or eventually the music consumer to browse huge musical database.

2.1.2 Plant Modeling

For the last 10 years, in collaboration with the Virtual Plants INRIA team and the Biological Modeling and Visualization research group in the Department of Computer Science at the University of Calgary, several conceptual and computing tools have been developed and implemented for plant simulation purpose: the *Virtual Laboratory* or for analyzing plants : *OpenAlea*. OpenAlea and Vlab are now open-source software, used by a lot of institutes all over the world. They allow to define new standards for measuring, analyzing and simulating plant architectures. Such a standardisation process would facilitate the constitution and diffusion of plant

database and would enable modellers to compare their models on the basis of publicly available database.

My first research works introduced new algorithmic methods for comparing plant architectures. Indeed, in many biological fields (e.g. horticulture, forestry, botany), a important need exists to quantify different types of variability within a set of plants. I have proposed several methods [J9, J10, J3, J4] to compare plant individuals based on a detailed comparison of their architectures. The core of the methods relies on an adaptation of algorithms for comparing rooted tree graphs. Using these algorithms, a distance between two plants can be defined as the cost of the transformation of one tree into the other (using basic "edit operations"). These methods have been used in different application fields and then evaluated for quantifying plant similarity [J11, C26, J5, SJ12].

Furthermore, they have been used to define measure of self-similarity and compression methods of plant architecture. Self-similarity of plants has captivated the attention of biologists for the last 50 years, yet its formal treatment is rare, and no measure for quantifying the degree of self-similarity currently exists. I have proposed a formal definition and a measure of self-similarity [J8], tailored to branching plant structures. Recently, the identification of repetitions used to define self-similarity as been extended in order to introduce a new method of compression of plant architecture [J1].

2.1.3 RNA Secondary Structure Comparison

In the meantime, during the last few years, I was interested in generalizing the mathematical methods developed in plant architecture context to other biological data. I have thus partially focused my research to the analysis of RNA Secondary structures.

During the last 10 years, our knowledge about the role of RNA molecules in the cell life process has been considerably modified. Notably, it has been stated that small non coding RNAs (such as small nuclear RNAs, small nuclear RNAs, microRNAs, siRNAs ...) play a very important role in nearly all aspects of gene regulation. Therefore, there is a crucial need for efficient computer-based methods to help handling the structural modeling and functional assignment of RNAs. From a historical perspective, many computational methods have been proposed to deal with the RNA folding problem. But there is still no reference tool to compare RNA molecules. In the context of BRASERO project supported by the ANR foundation, I have developed several methods for comparing RNA secondary structures and contributed to the analyse of the different existing methods [J7, J3]. From a computational perspective, these methods rely on global and local comparison algorithms of ordered and quotiented ordered trees. There are generalisation of methods introduced in the context of plant comparison.

2.1.4 Applications to Music Analysis

Lately, in 2005, in collaboration with two young researchers, Julien Allali and Pierre Hanna, we have developed a multidisciplinary group to evaluate the similarity between two musics. In this project we proposed to adapt the different tree-structured data analysis methods to the context of symbolic music analysis and then develop new approaches to analyse musical tree data structures.

The number of digital musical documents available on the Internet is always increasing. Unfortunately, users can only browse these database using text queries (title, singer, ...) and then need new classification and retrieval methods adapted to such huge database. For example, applications such as query-by-humming (whistling, singing) or query-by-example allowing the user to retrieve musical pieces similar from an audio query would make easier handling database.

The new browsing systems are mainly based on the estimation of similarity between two musical pieces. These systems compute a numeric score on how well a query matches each piece of the database and rank the music pieces according to this score. Computing such a degree of resemblance between two pieces of music is a difficult problem. Several approaches have been already proposed, but they are based on the similarity of timbre and evaluated using low level descriptors. The problem of defining musical similarity is a major problem in music information retrieval. From a computing perspective, the evaluation of similarity consists in developping algorithms that compute a measure of resemblance between two pieces (Sony CSL, Paris).

Techniques based on string matching [C19] are generally more accurate as they can take into account errors in the query or in the pieces of music of the database. This property is of major importance in the context of music retrieval systems since audio analysis always induces approximations. Moreover, some music retrieval application require specific robustness. Query by humming (QbH), a music retrieval system where the input query is a user-hummed melody, is a very good example. Since a hummed query can be transposed, played faster or slower, without degrading the melody, retrieval systems have to be both transposition and tempo invariant. Edit distance algorithms, mainly developed in the context of DNA sequence recognition, have been adapted in the context of music similarity.

We have thus proposed several models and methods to compare both sequence and tree representations of Music. These methods have been applied sucessfully to the problem of music information retrieval. First promising results have been obtained during the last few years and published in different international conferences and journals, for example [J6, C22, C19, C13]. In collaboration with Pierre Hanna, Julien Allali and Matthias Robine, since 2006, we took part in the MIREX context,where we obtained the best results in 2008.

2.2 Mobility

My research activity has been achieved in different labs:

- *From 1997 to 2000*, at the Botany and computational plant architecture" of Cirad Montpellier (France) with a grant of the French Ministry of Research.
- *In 2001*, at Biological Modeling and Visualization research group in the Department of Computer Science at the University of Calgary (AB, Canada) with a grant from INRIA (French National Institute for Research in Computer Science and Control)
- *From January 2002 to August 2002* at the Botany and computational plant architecture" of Cirad Montpellier (France) with a grant from the French Science Academy.
- *since 2002*, at the Computer Science Laboratory of Bordeaux 1 as an Associate Professor (Maître de Conférences).
- *from Septembre 2006 to February 2007*, during a six months delegation at CNRS, at Biological Modeling and Visualization research group in the Department of Computer Science at the University of Calgary (AB, Canada) with a grant of the French National Research Agency.
- Since september 2008, I am recipient of a "Delegation CNRS" at the Pacific Institute for Mathematical Sciences of the University of Calgary until 2010.

2.3 Involvement in ACI and ANR Programs

Since 2004, I have been supported by different call for proposal from the french minestry of research, the CNRS or the French National Agency for Research (ANR).

- I have been committee Member (local coordinator) of the coordination group of ACI Arborescences project (2004-2007) led by Yann Guédon. This project aimed at analysing the development of plant structure at a macroscopic scale in order to answer to both biological questions and finalised agronomic applications. We have based our research on three families of methods or models:

  1. Algorithms for tree alignment, allowing to compare plants,
  2. Hidden tree Markov models, allowing to identify homogeneous zones in measured plants,
  3. branching processes with dependency, describing generative aspect of plant growth.

  The methodology is similar to the one developed in genomic data analysis context where the structural specificity of data is highly related to the analysing method proposed.

- I have been committee Member (local coordinator) of the coordination group of ANR BRASERO project (2006-2009), led by Alain Denise. This project includes three original parts: the design of combinatorics models (based on graph and tree theory) along with efficient algorithms, the constitution of annotated benchmark sets for evaluation and validation on real biological problematics, and the development of freely-distributed software, as well as appropriate visualization facilities. In the end, this work should provide valuable tools which could become essential for the RNA community in the next coming years. This project relies on a close multidisciplinary cooperation between four partners that have a strong expertise in RNA algorithmic and biology.

- I have been committee Member (co-leader with Pierre Hanna) of the young researcher project SIMBALS granter by the ANR (2007-2010). The number of audio documents available on the Internet is highly increasing. New methods for browsing, retrieving or classifying have to be proposed to users. The growing Music Information Retrieval research community identifies and explicates the problems induced by these new methods. One of the key problems of this research area is the estimation of the musical similarity between audio data. In this project, we focus on this difficult and vast problem.

- Supervisor of the PEPS project Self-similarity of Music (2008-2009) granted by CNRS). This project aim at studying the properties of self-similarity in musical pieces by automatically identifying repetitions in sequences.

2.4 Teaching Activities and Student Supervision

2.4.1 Teaching Activities

Since 2002, I have taught at different levels from the first year ("Licence d'Informatique") to the last year at University ("Master d'Informatique" and "Master de BioInformatique").

Between 2003 and 2007, I was furthermore director of the Bio-informatics speciality of the Master in Computer Science (2003-2007).

2.4.2 Supervision of MsC Theses and Ph.D.

Since I joined the LaBRI, I supervised the works of many Master (DEA) trainees or 3rd year engineer school students.

Master Theses in Computer Science

Omer Nguema (University of Bordeaux 1) achieved is DEA in computer science under my full supervision in 2003. His thesis aimed at developing generalisation of comparison methods between sequences to define a distance between quotiented

sequences. A multi-scale model of sequences has been proposed by Omer, and then he has proposed algorithms allowing to compute a distance between such models. These methods have been implemented into the OpenAlea software.

Master Theses in Computer Science, speciality in BioInformatic

Omer Nguema's work has been used as a basis of **Sylvain Demey**'s (University of Bordeaux 1) master thesis in 2004. The goal of his thesis was to apply the previous methods to the comparison of biological events on axis of hybrid apple trees. This analysis allowed us to identify the differences between the methods used for comparing vegetative sequences. This work has been done under my full supervision and in collaboration with Evelyne Costes from the National Agronomic Research Institute (INRA) in Montpellier.

Master Theses in Computer Science, speciality in Models and Algorithms

Aïda Ouangraoua (the National Engineering School in Electrical, Electronic and Computer Science of Bordeaux) achieved her Master in Computer Science under my full supervision in 2004. RNA secondary structures comparison is part of the comparative approach that aims to infer the functions of RNA molecules from the similarities between the structures of these molecules. In this context, comparing two RNA secondary structures can be reduced to the comparison of ordered labeled tree-graphs representation of the structures. In this part of her research, she was interested in the development of efficient tree-to-tree editing algorithms for the comparison of ordered labeled tree-graphs dedicated to the comparison of RNA secondary structures.

Ione Ituarte, (Engineering School of the University of Mondragon - Spain) has done her final training of engineering school under my full supervision from November 2005 to June 2006. Ione has developed an interface for the comparison module between plants and RNA. Her software has been integrated into the OpenAlea software and is now available for the biologists. She also integrated a new method allowing to retrieve a subtree within a tree-graph at a distance d. Her work has been presented to the JOBIM conference [NC38].

Under Pierre Hanna supervision (25%) and mine (75%) **Pierre Schweitzer** (University of Bordeaux 1) ended his master thesis in 2008. His thesis has been achieved in the context of Simbals project and consisted in the automatic identification of repetitions in monophonic melodies. During his master thesis, Pierre has developed an original heuristic that allows to retrieve approximate repetitions in strings.

Ph.D. Thesis in Computer Science

Aïda Ouangraoua continued the works she had started during her master thesis, and defended her Ph.D. thesis in Computer Science in december 2007. Trees are

an important data structure in several fields: botanic, molecular biology, etc. Particularly, in most botanical applications, trees are used to represent the topological structure of plants architecture. The concept of architecture plays an important role in the understanding of the development and the growth of plants. Thus, the comparison of plant architectures has many applications in various fields of botany such as agronomy, horticulture, agriculture or forestry. Some initial works proposed during Aïda's master thesis, aimed at exploring tree comparison algorithms, their applications in botany and their extensions. The algorithms studied are based on the generalization of the methods for computing similarity between sequences. Their general principle consists in building a target tree by applying various structural operations, called edit operations, to an initial tree. In her thesis, she proposed a unification of the various tree edit methods by introducing the class of "semi-ordered trees" which involves unordered and ordered trees. Furthermore, she introduced new extensions of edit algorithms by taking account of constraints dedicated to the comparison of plant architectures or RNA secondary structures. For instance, multi-scales aspects, semi-order or order relations between the components of the structures are now taken into account. Aïda Ouangraoua is, since September 2009, full time researcher at the French National Institute for Research in Computer Science and Control (INRIA).

In October 2008, **Anne-Laure Gaillard** started a Ph.D. thesis under my full direction on the evaluation of self-similarity properties in plants. She focused her work on the methods to approximately compress trees.

2.4.3 Ph.D. Committees

I was one of the examiners of the Ph.D. thesis committee of Laurent Tichit which took place in December 2004 at the University of Bordeaux 1.

I was furthermore, as her supervisor, member of the Ph.D. thesis committee of Aïda Ouangraoua, which took place in December 2007 at the University of Bordeaux 1.

2.5 Publications

International Journals

[J1] P. Ferraro, C. Godin, P. Prusinkiewicz, Quantifying the degree of self-nestedness of trees. application to the structural analysis of plants, IEEE/ACM Transactions on Computation Biology and Bioinformatics in press (2009) 16. 13

[J2] P. Ferraro, J. Allali, P. Hanna, M. Robine, C. Illiopoulos, Toward a general framework for polyphonic comparison, Fundamenta Informaticae 97 (2009) 331–346.

[J3] A. Ouangraoua, P. Ferraro, A constrained edit distance algorithm between semi-ordered trees, Theoretical Computer Science 410 (8-10) (2009) 837–846. 13

[J4] A. Ouangraoua, P. Ferraro, A new constrained edit distance between quotiented ordered trees, Journal of Discrete Algorithms 7 (1) (2009) 78–89, doi:10.1016/j.jda.2008.05.001. 13

[J5] V. Segura, A. Ouangraoua, P. Ferraro, E. Costes, Comparison of tree architecture using tree edit distances: application to two-year-old apple hybrids, Euphytica 161 (2008) 155–164. 13

[J6] P. Hanna, P. Ferraro, M. Robine, On optimizing the editing algorithms for evaluating similarity between monophonic musical sequences, Journal of New Music Research 36 (4) (2007) 267–279. 14

[J7] A. Ouangraoua, P. Ferraro, S. Dulucq, L. Tichit, Local similarity between quotiented ordered trees, Journal of Discrete Algorithms 5 (1) (2007) 23–35. 13

[J8] P. Ferraro, C. Godin, P. Prusinkiewicz, Toward a quantification of self-similarity in plants, Fractals 13 (2) (2005) 1–25. 13

[J9] P. Ferraro, C. Godin, An edit distance between quotiented graphs, Algorithmica 36 (2003) 1–39. 13

[J10] P. Ferraro, C. Godin, Optimal mappings with minimum number of connected componanents in tree-to-tree comparison problems, Journal of Algorithms 48 (2003) 385–406. 13

[J11] P. Ferraro, C. Godin, A distance measure between plant architectures, Annals of Forest Science 57 (2000) 445–461. 13

Submission to International Journals

[SJ12] A. Ouangraoua, V. Segura, E. Costes, P. Ferraro, Methods to evaluate similarity between plant architectures based on the comparison of their tree structured representation, BMC Systems Biology submitted (2008) 8. 13, 19

Publications in International Conferences (with review)

[C13] J. Allali, P. Antoniou, P. Ferraro, C. Iliopoulos, S. Michalakopoulos, Overlay problems for music and combinatorics, in: Proceedings of the 15th International Conference on Auditory Display, Copenhagen, Dn, 2009, pp. 138–143. 14, 20

[C14] P. Antoniou, J. Allali, C. Iliopoulos, P. Ferraro, Validation and Decomposition of Partially Occluded Images with Holes, in: Proceedings of the Prague Strincology Conference Prague Stringology Conference, Prague Tchèque, République, 2009, p. to appear.

[C15] P. Ferraro, J. Allali, P. Hanna, M. Robine, Extending Alignment Algorithm for Polyphonic Comparison, in: Proceedings of the International Computer Music Modeling and Retrieval Conference International Computer Music Modeling and Retrieval Conference (CMMR), Danemark, 2009, pp. 206–209.

[C16] P. Ferraro, P. Hanna, L. Imbert, T. Izard, Accelerating Query-by-Humming on GPU, in: 10th International Society for Music Information Retrieval Conference, Kobe Japon, 2009, p. to appear, 6.

[C17] M. Robine, P. Hanna, T. Rocher, P. Ferraro, Structured Representation of Harmony for Music Retrieval, in: Proceedings of the International Computer Music Conference International Computer Music Conference (ICMC), Canada, 2009, p. to appear.

[C18] P. Hanna, M. Robine, P. Ferraro, Visualisation of musical structure by applying improved editing algorithms, in: Proceedings of the International Computer Music Conference (ICMC), Belfast, Northern Ireland, 2008, p. to appear.

[C19] P. Hanna, M. Robine, P. Ferraro, J. Allali, Improvements of alignment algorithms for polyphonic music retrieval, in: Computer Music Modeling and Retrieval 2008, Copenhague, 2008, pp. 244–251. 14, 20

[C20] P. Ferraro, P. Hanna, Optimizations of local edition for evaluating similarity between monophonic musical sequences, in: RIAO 2007: Proceedings of the 8th International Conference on Information Retrieval - Large-Scale Semantic Access to Content (Text, Image, Video and Sound), Pittsburgh, PA, USA, 2007.
URL http://www.riao.org

[C21] C. Godin, P. Ferrarro, Self-similar analysis of plant architecture reveals hierachical classes of meristem states, in: 5th International Workshop on Functional Structural Plant Models, Napier, Napier, NZ, 2007, pp. 481–484.

[C22] M. Robine, P. Hanna, P. Ferraro, J. Allali, Adaptation of string matching algorithms for identification of near-duplicate music documents, in: Proceedings of the International SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN), Amsterdam, Netherlands, 2007, pp. 37–43. 14, 20

[C23] J. Allali, P. Ferraro, P. Hanna, C. Iliopoulos, Local transpositions in alignment of polyphonic musical sequences, in: N. Ziviani, R. Baeza-Yates (Eds.), 14th String Processing and Information Retrieval Symposium, Vol. 4726 of Lecture Notes in Computer Science, Springer, 2007, pp. 26–38. 71, 61

[C24] S. Dufour-Kowalski, C. Pradal, N. Dones, P. Barbier de Reuille, F. Boudon, J. Chopard, D. DaSilva, J. Durand, P. Ferraro, C. Fournier, Y. Guédon, A. Ouangraoua, C. Smith, S. Stoma, F. Theveny, H. Sinoquet, C. Godin, Openalea: An open-source platform for the integration of heterogeneous fspm components, in: 5th International Workshop on Functional Structural Plant Models, Napier, NZ, 2007, pp. 361–362.

[C25] P. Hanna, P. Ferraro, Polyphonic music retrieval by local edition of quotiented sequences, in: Proceedings of the 5th International Workshop on Content-Based Multimedia Indexing (CBMI), Bordeaux, France, 2007, pp. 61–68.

[C26] A. Ouangraoua, P. Ferraro, Plant architecture comparison methods : a review of existing algorithms, in: 5th International Workshop on Functional Structural Plant Models, Napier, NZ, 2007, pp. 471–474. 13, 19, 50

[C27] M. Robine, P. Hanna, P. Ferraro, Music similarity: Improvements of edit-based algorithms by considering music theory, in: Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR), Augsburg, Germany, 2007, pp. 135–141.

[C28] V. Segura, A. Ouangraoua, P. Ferraro, E. Costes, Comparison of tree architecture using a tree edit distance: application to 2-year-old apple hybrids, in: XIII EUCARPIA Biometrics in Plant Breeding Section Meeting, 2006, p. 21.

[C29] P. Ferraro, C. Godin, P. Prusinkiewicz, A structural method for assessing self-similarity in plants, in: Fourth International Workshop on Functional-Structural Trees Models, Montpellier, France, 2004, pp. 56–61.

[C30] M. Nikolski, P. Ferraro, P. Durrens, M. Aigle, Saccharomyces siliceus, in: First International Workshop on Systems Biology of Yeast, St Louis, 2003.

[C31] D. Auber, M. Delest, J.-P. Domenger, P. Ferraro, R. Strandh, EVAT : Environment for visualization and analysis of trees, in: IEEE Symposition on Information Visualisation Contest, Vol. www.cs.umd.edu/hcil/iv03contest/, 2003, pp. 124–126.

[C32] P. Ferraro, P. Prusinkiewicz, L. Münderman, C. Godin, L-systems and MTGs: Integrating simulation and formal analysis of architectural plant models, in: 16th European Simulation Multiconference, Modelling and Simulation, Darmstadt, 2002, pp. 418–423.

[C33] P. Ferraro, P. Prusinkiewicz, L. Münderman, C. Godin, Integration of a simulating system to formal analysis of architectural plant models, in: Third International Workshop on Functional-Structural Trees Models, Montréal, QB, Canada, 2001, pp. 24–26.

[C34] P. Ferraro, C. Godin, An algorithm for comparing unordered tree graphs based on a minimum cost mapping with a minimal connectivity, in: Third International Conference on Orders, Algorithms and Applications, Montpellier (France), 1999.

[C35] P. Ferraro, C. Godin, A distance measure between plant architectures based on the comparison of their topological structures, in: Second International Workshop on Functional-Structural Trees Models, Clermont-Ferrand (France), 1998.

Publications in National Journals

[NJ36] P. Hanna, P. Ferraro, M. Robine, J. Allali, Recherche de Documents Musicaux par Similarité Mélodique, Document numérique 11 (3-4) (2008) 107–125.

Publications in National Conferences (with review)

[NC37] J. Allali, Y. d'Aubenton Carafa, C. Chauve, A. Denise, C. Drevet, P. Ferraro, D. Gautheret, C. Herrbach, F. Leclerc, A. de Monte, A. Ouangraoua, M.-F. Sagot, C. Saule, M. Termier, C. Thermes, H. Touzet, Benchmarking rna secondary structure comparison algorithms, in: Actes des Journés Ouvertes de Biologie, Informatique et Mathḿatiques - JOBIM'08, 2008, pp. 67–68. 57

[NC38] I. Ituarte, A. Ouangraoua, P. Ferraro, PyTreeMatch : a python based software for comparing RNA Secondary Structure, in: JOBIM, Bordeaux, 2006, p. 127. 17, 21

[NC39] P. Ferraro, L. Tichit, S. Dulucq, Local similarity between trees, in: 5èmes Journées Ouvertes Biologie Informatique Mathématiques, Montréal, Canada, 2004, p. 19.

[NC40] A. Ouangraoua, P. Ferraro, Distance multi-échelles entre structures secondaires d'ARNs, in: 5èmes Journées Ouvertes Biologie Informatique Mathématiques, Montréal, Canada, 2004, p. 81.

[NC41] P. Ferraro, C. Godin, P. Prusinkiewicz, Une méthode structurelle pour évaluer les propriétés d'autosimilarité des plantes, in: Colloque Autosimilarité et Applications (Self Similarity And Applications Workshop), Clermont-Ferrand (France), 2002.

Technical Report and Others

[R42] P. Ferraro, P. Hanna, MIREX symbolic music similarity, in: I. M. I. R. S. E. Laboratory (Ed.), MIREX 2006, the Second Annual Music Information Retrieval Evaluation eXchange, 2006, pp. 90–91.

[R43] P. Ferraro, A. Ouangraoua, Local mapping between unordered trees, Tech. Rep. RR-105, LaBRI (dec 2005).
URL http://w5.labri.fr/publications/combalgo/2005/FO05

[R44] P. Ferraro, Rapport de recherche post-doctoral 2001-2002, Tech. Rep. 2, Cirad (2002).

[R45] C. Godin, Y. Guédon, S. Bellouti, C. Nouguier, P. Ferraro, N. Dones, B. Adam, Introduction and reference manual, AMAPmod version 1.5, Tech. rep., Cirad (2002).

Plant Modeling

3

As I introduced earlier, plant architecture analysis has focused my research for a long period in my researcher life. I present hereafter an overview of my works on this topic.

Numerous societal challenges involve plants as sources of diverse products for humans, such as food, medicines, fibbers, woods, carburant, *etc.* Agronomic systems that aim at controlling and optimizing plant production in order to maximize gains for human uses, have benefit from large progresses obtained during plant domestication and breeding processes, and thanks to the improvement of agronomic practices. Numbers of these progresses have been based on the progressive modification of plant structures. In particular, during plant domestication and breeding processes, plant structures have been progressively reduced by the selection of plants with low branching and small size that are adapted to high density plantation systems. This evolution has been particularly rapid during the 20th century, after the discovery of dwarf mutants and the subsequent "green revolution".

Today, beyond traditional agronomic approaches, which attempt to optimise biomass production as a function of different treatments, new interests are expressed by the society in ecology, sustainable agronomy and environmental changes. The challenge for the coming years will be to maintain plant productivity and food supply for humanity despite a progressive degradation of the environmental conditions. The increase of high temperature and drought periods during summer times will combine with an increase in demographic pressure on lands, this situation leading to an increasing number of constraints that agronomists and growers will have to face. In this context, plant plasticity in responses to fluctuating environmental conditions represents a key question which will mobilize numerous researches in applied biology and agronomy, and will necessitate revisiting genetic and agronomic issues with innovative approaches.

Most of the approaches that are developed in agronomy and quantitative genetics are based on the decomposition of plant phenotype in elementary variables, easily related to plant productivity. Such variables are often the plant height, trunk diameter or volume, plant biomass, leaf area, *etc.* These approaches have been very successful in estimating the genetic gain or the yield improvements, without entering into the complexity of plant phenotype. However, to describe and understand plant responses to environmental conditions, more detailed information would allow biologists to identify the pertinent variables and development stages where these answers can be captured. New approaches need thus to be explored in order to study plant structure plasticity. Due to the complexity of plant structures, the associated scientific problems need a new multi-disciplinary scientific approach in which not only the production of plants is controlled but also the way in which plants elaborate this production during their lifetimes.

3.1 Formal Representations of Plants

The architectural development of a plant is the result of many elementary morphological events distributed over time and space. Whatever the size, shape and complexity of the resulting architecture, only a few categories of homologous botanical entities can be identified in the caulinar shoot system of an adult plant [Barthélémy 1991a]. These types of botanical entities can be ordered from the most elementary which is the node or more precisely the set: {node, internode, leaf(ves) and axillary bud(s)} referred to as metamer [White 1979], to more macroscopic entities such as the axis (*i.e.* succession of metamers built up by a single meristem).

3.1.1 Sequence Decomposition

Plant development is the result of two basic mechanisms: the apical growth process and the branching process. Each of these two basic mechanisms corresponds to a particular meristem activity. An apical meristem is defined as a set of embryonic cells located in the apical part of each axis. These particular cells are always able to divide and thus to generate various tissues (pith, epiderm, vascular bundles, *etc.*) or

organs (leaf, flower, *etc.*). The apical meristem activity also generates small lateral sets of embryonic cells, the axillary meristems, which may give rise to axillary or offspring shoots. In the case of the apical growth process, a succession of metamers is built up by a single apical meristem. If no resting periods are observed during the elongation, the growth is qualified as continuous and leads to the edification of an axis made up of a succession of metamers. If the apical growth takes the form of a succession of elongation and resting periods (with annual periodicity for temperate species), the growth is qualified as rhythmic. This temporal organization translates into a spatial organization which may be identified by morphological markers *e.g.* scale leaves and short internodes corresponding to slowdowns or stops of growth [Caraglio 1997]. In this way, different types of macroscopic botanical entities may be defined such as growth units (portion of leafy axis, *i.e.* succession of metamers built up between two resting phases or annual shoots (portion of a leafy axis built up over a single year for temperate species, and which may be composed of one or several successive growth units [Godin 1998]. In the case of the branching process, an offspring entity is built up by an axillary meristem. This offspring entity, which may be either a flower resulting from the transformation of this meristem or a shoot, will be globally referred to as an axillary production. The entities borne on the successive nodes of a given *parent* entity may exhibit a remarkable structure.

Both the apical growth and the branching processes induce sequential structures at the node succession scale along growth units (sequence of internode lengths or sequence of axillary productions). Hence, the node constitutes a natural index parameter from which sequences can be built [Guédon 2003]. Depending on the rhythmic or continuous character of apical growth, these sequences may correspond to growth units, annual shoots or axis. It should be noted that the year of a shoot or the rank of a growth unit are also valid index parameters from which more macroscopic sequences can be built.

3.1.2 Tree-Graph Models of Plant

Plant architectures are also described as tree-graphs [Godin 1998]) by considering a plant as a set of botanical elementary entities. Depending on the scale of decomposition (Fig. 3.1) several sets of entities are considered (set of axes, growth units or internodes). At a given scale, the plant architecture is then represented by a directed graph $T = (V, E)$ where V is a set of vertices representing the plant entities at this scale and E is a set of edges describing the connections between entities, each edge being represented by an ordered pair of vertices such that there exists an edge (x, y) in E if and only if the entity corresponding to y is linked to the parent entity corresponding to x. For any edge (x, y) in E, x is called a parent of y and conversely y is a child of x. In a plant, since each entity is physically attached to at most one parent entity except the root entity which has no parent entity and is linked to the plant root, then the topological structure is represented by a tree-graph.

In order to account for different natures of connections between botanical entities, complementary definitions can be added to a tree-graph representation of a plant

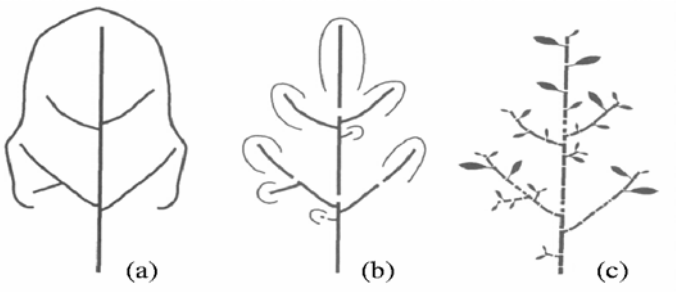

Figure 3.1: 3 scales of decomposition of the same plant (a) axes, (b) growth units and (c) internodes.

architecture. In the plant, a parent entity can be connected to several children entities and if no difference is taken into account for the natures of connections between parents and their children then no order is considered on the set of children entities of a given parent entity. In this case, the plant architecture is modeled by an unordered tree-graph [Ferraro 2000]. However, two natures of connections giving rise to two types of children entities can be distinguished according to the growing and branching processes in plants. An entity is a successor entity of its parent if it is on the same axis as his parent otherwise it is a born entity. Each parent entity can then have at most one successor entity and several born entities and thus a semi-order relation between the set of children entities of a parent entity can be defined : the successor entity is the first in the set while the born entities are all equivalent. This phenomenon is observed for instance in whorl plants, or can result from sampling procedure when a large number of plants have to be described [Segura 2006, Segura 2008]. In these cases, the plant architecture is represented by semi-ordered tree-graphs [Ouangraoua 2009a]. In the particular case where each entity has at most one child, the set of plant entities can be totally ordered at any scale of decomposition and the plant architecture can be represented by ordered tree-graphs (Fig. 3.2).

Furthermore, to take account of the multiscale nature of plant structures [Godin 1998], plants are often represented by quotiented trees or multiscale tree-graphs.

3.1.3 Formal Definitions

In the following of the report, we will use specific notations and definitions presented hereafter.

The set of children of a vertex is denoted by $child(v)$ which is of size n_v and $\deg(G) = \max_{v \in V}\{n_v\}$. For every k in $\{1, .., n_v\}$, v_k denotes a child of v. If x is any vertex of tree $T[v]$, $F[x]$ denotes the *forest rooted in* x, *i.e.* obtained from $T[x]$ by removing the root x and all the edges incident with x.

Formally, an ordered tree is a pair $(T, S_{\preceq})$ where T is a rooted tree and $S_{\preceq} = \{\preceq_x$

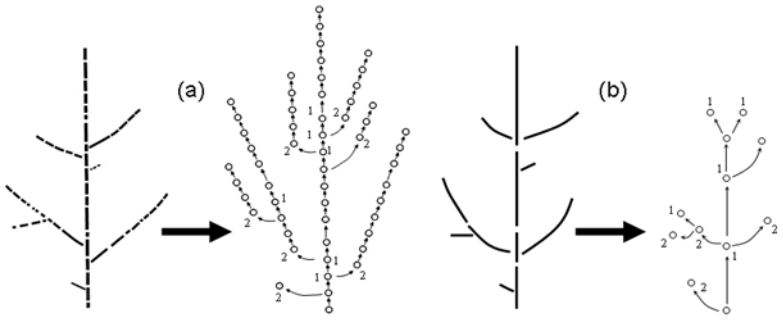

Figure 3.2: Schematic representations of a plant architecture (left) and its corresponding tree-graph (right); (a) Ordered tree-graph representation at internode level; (b) Semi-ordered tree-graph at growth unit level. The order between children is indicated when there is more than one child.

, $x \in T\}$ is a set of total order relations such that for any vertex x of T, $\preceq_x$ is a total order relation on $\mathrm{C}(x)$. The set of total order relations $S_\preceq$ and the ancestor-descendant relation $\leq$ on the vertices of T induce a total order relation on the set of vertices of T (namely *prefix* or *suffix* order relations). An unordered tree is a rooted tree T such that the only significant relation between the vertices of T is the ancestor-descendant relation (all the children of a vertex of T are equivalent). We have proposed to unify these both data structures in a single one called semi-ordered trees [Ouangraoua 2009a]. A *semi-order relation* [Preparata 1973] $\sqsubseteq$ on a set V is a reflexive and transitive binary relation on V. $\sqsubseteq$ is a *total semi-order relation* if for any pair (x, y) of elements of V, $x \sqsubseteq y$ or $y \sqsubseteq x$, otherwise $\sqsubseteq$ is called *partial semi-order relation.*

Definition 1 (Semi-ordered tree) *A semi-ordered tree is a pair $(T, S_\sqsubseteq)$ where T is a rooted tree and $S_\sqsubseteq = \{\sqsubseteq_x,\ x \in T\}$ is a set of total semi-order relations such that for any vertex x of T, $\sqsubseteq_x$ is a total semi-order relation on $C(x)$.*

Note that an unordered tree T is a semi-ordered tree $(T, S_\sqsubseteq)$ such that for any vertex x of T, $\sqsubseteq_x$ is an equivalence relation, *i.e* for any pair (x_1, x_2) in $\mathrm{C}(x)^2$, $x_1 \sqsubseteq_x x_2$ and $x_2 \sqsubseteq_x x_1$ (all the children of x are equivalent). Similarly, an ordered tree T is a semi-ordered tree $(T, S_\sqsubseteq)$ such that for any vertex x of T, $\sqsubseteq_x$ is a total order relation on $\mathrm{C}(x)$.

Let $(T, S_\sqsubseteq)$ be a semi-ordered tree, the set of semi-order relations $S_\sqsubseteq$ and the ancestor-descendant relation $\leq$ on the vertices of T induce an order relation denoted by $\sqsubseteq$ on the set of vertices of T. Let consider two vertices x and y, x will be smaller than y according to $\sqsubseteq$ if and only if x is an ancestor of y or there exists two ancestor s and t of respectively x and y, having the same parent u such that s is smaller than t according to $\sqsubseteq_u$.

Formally, for any vertex u of T and $(s, t) \in \mathrm{C}(u)^2$, the relation $s \sqsubseteq_u t$ and $t \not\sqsubseteq_u s$ is denoted by $s \sqsubset_u t$. The order relation $\sqsubseteq$ is such that for any pair (x, y) of vertices

of T, x and y verify $x \sqsubseteq y$ if and only if x is ancestor of y or there is a pair (s,t) of vertices in $\mathrm{C}(x \wedge y)$ such that $s \leq x$ and $t \leq y$ and $s \sqsubset_{x \wedge y} t$:

$$\forall (x,y) \in V^2 \ , \ x \sqsubseteq y \Leftrightarrow \left\{ \begin{array}{l} x \leq y \text{ or,} \\ \exists \ (s,t) \in \mathrm{C}(x \wedge y)^2 \mid s \leq x \text{ and } t \leq y \text{ and } s \sqsubset_{x \wedge y} t. \end{array} \right.$$

Note that if T is an unordered tree then $\sqsubseteq$ is simply the ancestor-descendant relation and if T is an ordered tree then $\sqsubseteq$ is a total order relation on V (namely the prefix order relation), in any other case $\sqsubseteq$ is a partial order relation.

Let V be a set and $\sqsubseteq$ a total semi-order relation on V, $\sqsubseteq$ induces an *equivalence relation* (*i.e.* reflexive, transitive and symetric) on V denoted by $\equiv$ and defined by:

$$\forall (x,y) \in V^2, \ x \equiv y \Leftrightarrow x \sqsubseteq y \text{ and } y \sqsubseteq x.$$

The *class of equivalence* of an element x in V is denoted by $[x] = \{y \in V \mid x \equiv y\}$ and the set of equivalence classes of the elements of V (defining a partition of V) is denoted by $V_{\equiv} = \{[x] \mid x \in V\}$. In the same way, the total semi-order relation $\sqsubseteq$ induces a *total order relation* $\preceq$ (*i.e.* reflexive, transitive and antisymetric) on $V_{\equiv}$ defined by:

$$\forall ([x],[y]) \in V_{\equiv}^2, \ [x] \preceq [y] \Leftrightarrow x \sqsubseteq y.$$

A quotiented tree is a tree with an equivalence relation defined on the set of vertices, and such that the resulting quotient graph is also a tree. A quotiented tree can thus be considered as an autosimilar structure represented by trees on two different scales.

More precisely, a quotiented graph H is a 3-uple (G, W, π) where $G = (V, E)$ is a directed graph called the *support* of H, W is a set of vertices and π is a surjective mapping from V to W. For any vertex x in V, the vertex $\pi(x)$ is called the *complex* of x and reciprocally x is a *component* of $\pi(x)$. $\pi^{-1}(z)$ denotes the set of components of a vertex z of W and if x is a vertex of V, $\Pi(x)$ denotes the set $\pi^{-1}(\pi(x))$ of components of $\pi(x)$. The size of $\Pi(x)$ is denoted by $|\Pi(x)|$ and $\deg_{\pi}(H) = \max_{x \in V}\{|\Pi(x)|\}$. The function π induces a partition Π_H on V: $\Pi_H = \{\pi^{-1}(z) | \quad z \in W\}$. The *quotient graph* $\mathcal{Q}(H)$ associated with H is the graph (W, E_{π}) such that:

$$\forall (x,y) \in E, \quad (\pi(x), \pi(y)) \in E_{\pi} \Leftrightarrow \pi(x) \neq \pi(y)$$

The multiscale representation of plants is based on the notion of quotiented and multiscale tree-graphs. Quotiented graphs whose support and quotient graphs are trees are called *quotiented trees*. Let $H = (G, W, \pi)$ be a quotiented graph with support graph $G = (V, E)$ which is either a tree or a forest. Let $x \in V$, then $H[x]$ denotes the quotiented graph $(G[x], W[\pi(x)], \pi_{/x})$ where $G[x]$ is the sub-tree or a forest of G rooted in x, $W[\pi(x)]$ is the set of vertices of the sub-tree of $\mathcal{Q}(H)$ rooted in $\pi(x)$ and $\pi_{/x}$ is the restriction of π to $V[x]$. If $G[x]$ is a tree, $H[x]$ is a quotiented tree.

A multi-scale tree-graph is then a recursive definition of a quotiented tree graph.

3.2 Plant Comparison

The increasingly important role played by plant architecture in the structure/function modeling of plants generates a need for new investigational tools. Generic tools have already been developed to visualize plant architecture in 3-dimensions [de Reffye 1989, Prusinkiewicz 1990], to model the growth of plant architecture, *e.g.* [Kurt 1994, Mech 1996, LeDizès 1997], to measure plant architecture [Godin 1997, Hanan 1997, Sinoquet 1997b], and to explore and to analyze the plant [Godin 1997].

To compare two plants, a first approach consists of summarizing each individual by a small number of synthetic and global variables (*e.g.* fruit production, crown size, *etc.*). The similarity of two individuals is then reduced to the similarity between these synthetic variables. In forestry for instance, wood production and quality are usually assessed by measuring variables such as stem diameter, crown volume, branching density, etc. Comparing different wood qualities thus amounts to comparing these global variables. This defines global comparison methods in which the topological organization of plant entities is not taken into account.

On the other hand, domains exist in which plant topological structure plays an important role. In forestry for example, refining wood quality criteria leads foresters to consider more detailed descriptions of tree crowns, taking for instance into account the spatial distribution of branches along the stems or branch geometry (*e.g.* [Kuppers 1985]). Similarly, in horticulture, determining the fruiting position in the tree crown leads to a better understanding of the fruiting habits and production parameters (*e.g.* [Costes 1998]). In such cases, the notion of distance between individuals would naturally take into account the topological and spatial organization of plant entities. This defines analytical comparison methods which are based on a piece-by-piece comparison of plants [Miclet 1984].

Since in most applications, descriptions of plant architecture usually rely on a tree graph representation of topological structures, during my research, I have proposed several algorithms with bounded complexity to compute a distance between tree graphs. This distance, based on algorithms proposed in theoretical computer science [Zhang 1993, Zhang 1996], are defined as the minimum cost of the sequence of elementary edit operations needed to transform one tree graph into the other.

3.2.1 A General Framework to Compare Unordered Trees

A considerable amount of work has been performed on comparison algorithms for problems that can be modeled as data sequences [Stanley 1986]: in molecular biology [Sobel 1986, Kececioglu 1995], in speech or text recognition [Levinson 1978], in code error correction [Tanaka 1986], in plant modeling [Guédon 1998, Guédon 2001]. In the early seventies, Wagner and Fisher [Wagner 1974] presented an algorithm which computes the distance between two strings of characters as the minimum cost sequence of elementary operations needed to transform one string into the other. In order to define a distance between rooted tree graphs, Taï [Tai 1979], Selkow

[Selkow 1977] and Lu [Lu 1979], [Zhang 1989] proposed a generalization of the Wagner and Fisher algorithm with application in different fields [Noetzel 1983]. All the tree graphs discussed in these papers are ordered, meaning that the sets of sons of any vertex are ordered sets. These algorithms cannot always be applied directly to the problem of plant comparison since tree graphs used to represent plant topology are unordered [Godin 1998] or semi-ordered [Ouangraoua 2009a]. However, Zhang [Zhang 1993, Zhang 1996] proposed an algorithm in theoretical computer science for computing a distance between unordered rooted tree graphs based on Lu's method, by introducing a new hypothesis in the tree-graph transform. The following section briefly describes the main principle of the algorithm.

A distance measure between two trees T_1 and T_2 is defined by considering the minimum cost of elementary operations needed to transform T_1 into T_2. Three kinds of elementary operations, called edit operations are considered: changing one vertex into another (note that this may change labels), deleting (*i.e.* making the sons of a vertex v become the sons of the father of v and then removing v from T_1) or inserting one vertex (*i.e.* the symmetric operation on T_2). In order to transform one tree graph into the other, all the vertices of T_1 and T_2 must be affected by at least one edit operation.

A cost function, called local distance, is defined for each edit operations s. The local distance assigns a non-negative real number $\gamma(s)$ to s:

- $\gamma(s) = d(v_1, v_2)$ if s changes the vertex v_1 into the vertex v_2,
- $\gamma(s) = d_{del}(v_1, v_2)$ if s deletes the vertex v_1, and,
- $\gamma(s) = d_{ins}(v_1, v_2)$ if s inserts the vertex v_2.

Let S be a sequence of n edit operations $(s_1, s_2, \ldots, s_n)$ which transform one tree graph T_1 into another one T_2. The cost $\gamma(E)$ of a sequence of edit operations is then defined by summing up the cost of the edit operations that compose E. The dissimilarity measure $D(T_1, T_2)$ from a tree graph T_1 to a tree graph T_2 is then measured as the minimum cost of a sequences in S:

$$S(T_1, T_2) = \max_{E \in \mathcal{E}} \{\sigma(E)\},$$

where $\mathcal{E}$ represents the set of the sequences of edit operations transforming T_1 into T_2. Likewise, we can extend this notion to the similarity between forests $S(F_1, F_2)$.

Zhang showed that the dissimilarity measure is a distance [Zhang 1996, Zhang 1993]. According to this definition, Zhang then proposed an algorithm with bounded complexity for solving this optimization problem which consists of finding a valid matching function with minimum cost. This algorithm is based on the following recursive equations.

Theorem 1 *[Zhang 1996, Zhang 1993] Let T_1 and $T2$ be two tree-graphs respectively rooted in v and w, then $D(T_1[v], T_2[w])$ and $D(F_1[v], F_2[w])$ can be computed recursively:*

1. *initialisation:*

$$\begin{array}{lcllcl} D(\theta,\theta) & = & 0 & & & \\ D(F_1[v],\theta) & = & \sum_{v_k \in son[v]} D(T_1[v_k],\theta), & D(T_1[v],\theta) & = & D(F_1[v],\theta) + d(v,\lambda) \\ D(\theta,F_2[w]) & = & \sum_{w_k \in son[w]} D(\theta,T_2[w_k]), & D(\theta,T_2[w]) & = & D(\theta,F_2[w]) + d(\lambda,w) \end{array}$$

2. *Distance between trees:*

$$D(T_1[v],T_2[w]) = \min \begin{cases} D(\theta,T_2[w]) + \min_{w_k \in son[w]} \{D(T_1[v],T_2[w_k]) - D(\theta,T_2[w_k])\} \\ D(T_1[v],\theta) + \min_{v_k \in son[v]} \{D(T_1[v_k],T_2[w]) - D(T_1[v_k],\theta)\} \\ D(F_1[v],F_2[w]) + d(v,w) \end{cases}$$

3. *Distance between forests:*

$$D(F_1[v],F_2[w]) = \min \begin{cases} D(\theta,F_2[w]) + \min_{w_k \in son[w]} \{D(F_1[v],F_2[w_k]) - D(\theta,F_2[w_k])\} \\ D(F_1[v],\theta) + \min_{v_k \in son[v]} \{D(F_1[v_k],F_2[w]) - D(F_1[v_k],\theta)\} \\ \min_{M \in \mathcal{R}(v,w)} \{\gamma(M)\} \end{cases}$$

Zhang [Zhang 1993] models the computation of $\min_{M \in \mathcal{R}(v,w)} \{\gamma(M)\}$ as a problem of minimum cost maximum flow, which mainly determines the overall complexity of the final algorithm. The complexity of this algorithm is:

$$O\left(|T_1| \times |T_2| \times (\deg(T_1) + \deg(T_2)) \times \log_2 (\deg(T_1) + \deg(T_2))\right)$$

3.2.2 Mapping between semi-ordered trees

In order to take into account of the more general representation of plant architecture, we have developped a generalisation of the previous method to compare semi-ordered tree graphs [Ouangraoua 2009a].

Now, we consider optimal sequences of edit operations from T_1 to T_2 such that the corresponding mapping associates a vertex of T_1 to at most one vertex of T_2 and reciprocally and preserves the relations defined on the vertices of T_1 and T_2, namely the ancestor-descendant relation and the semi-order relations. Preserving the semi-order relations $\sqsubseteq_x$ is equivalent to the preservation of the order relation $\sqsubseteq$ induced on the vertices of T_1 and T_2.

A *valid mapping* from $T_1 = (V_1, E_1)$ to $T_2 = (V_2, E_2)$ is a set M of ordered pairs of vertices (x_1, x_2) with $x_1 \in V_1$ and $x_2 \in V_2$ such that for any pairs (x_1, x_2) and (y_1, y_2) in M:

$$\begin{array}{lll} x_1 = y_1 & \Leftrightarrow x_2 = y_2 & \text{(one-to-one)} \\ x_1 \leq y_1 & \Leftrightarrow x_2 \leq y_2 & \text{(ancestor-descendant preservation)} \\ x_1 \sqsubseteq y_1 & \Leftrightarrow x_2 \sqsubseteq y_2 & \text{(semi-order preservation)} \end{array}$$

For any pair (x_1, x_2) in M, x_1 and x_2 are said image of each other and are denoted by $M(x_1) = x_2$ and $M(x_2) = x_1$. If there is no confusion, $(x_1, x_2) \in M$ is denoted by $x_1 \in M$ and $x_2 \in M$. Let F_1 (resp. F_2) be a subforest of T_1 (resp. T_2), the image of F_1 (resp. F_2) denoted by $M(F_1)$ (resp. $M(F_2)$) is the subforest of T_2 (resp. T_1) containing all the images of vertices of F_1 (resp. F_2). The concept of mapping is extended to forests and sets of vertices not necessarily connected.

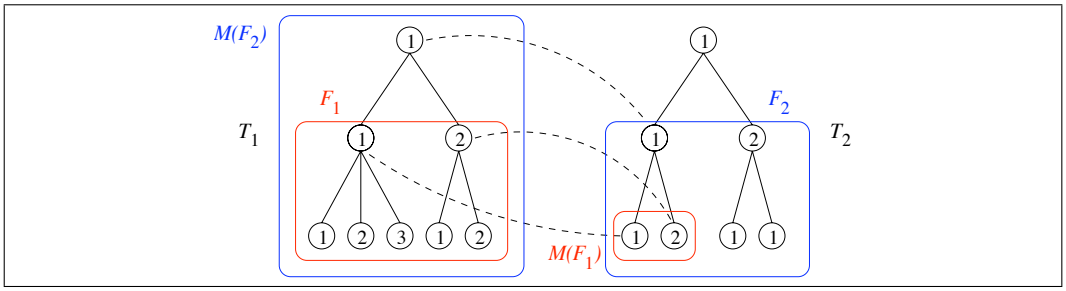

Figure 3.3: A valid mapping from a semi-ordered tree T_1 to a semi-ordered tree T_2 and the images $M(F_1)$ and $M(F_2)$ of subforests F_1 and F_2 of T_1 and T_2 respectively. The vertices of T_1 et T_2 image of each other are linked by dotted lines.

Let M be a valid mapping from T_1 to T_2, let I (resp. J) be the set of vertices of T_1 (resp. T_2) which do not appear in a pair of M, the cost of M is defined by:

$$\gamma(M) = \sum_{(x,y)\in M} \gamma(x,y) + \sum_{x\in I} \gamma(x,\lambda) + \sum_{y\in J} \gamma(\lambda,y).$$

The *edit distance* between T_1 and T_2 is the minimum cost of a valid mapping from T_1 to T_2. Computing the edit distance between two semi-ordered trees is equivalent to the computation of the edit distance between unordered trees which is a MAX-SNP-hard problem [Zhang 1994].

3.2.3 Constrained edit distance

Since the computation of the edit distance between two semi-ordered trees is a MAX-SNP-hard problem, we consider a constrained version of the problem: the computation of the *constrained edit distance* introduced by Zhang [Zhang 1996] to compare unordered trees.

A *constrained mapping* from T_1 to T_2 is a valid mapping (M, T_1, T_2) such that for any pairs (x_1, x_2), (y_1, y_2) and (z_1, z_2) in M:

$$(x_1 \wedge y_1) \leq z_1 \quad \Leftrightarrow (x_2 \wedge y_2) \leq z_2 \quad \text{(structure preservation)}$$

Examples of mappings between semi-ordered trees are given in Figure 3.4. Let x (resp. y) be a vertex of T_1 (resp. T_2), let $X = \{x_1, \ldots, x_n\}$ (resp. $Y = \{y_1, \ldots, y_m\}$) be a subset of $\mathrm{C}(x)$ (resp. $\mathrm{C}(y)$), the set of constrained mappings from $T_1[x]$ to $T_2[y]$ (resp. $F_1[X]$ to $F_2[Y]$) is denoted by $\mathcal{M}_C(T_1[x], T_2[y])$ (resp. $\mathcal{M}_C(F_1[X], F_2[Y])$).

Definition 2 (Constrained edit distance) *The constrained edit distance between T_1 and T_2 is the minimum cost of a constrained mapping from T_1 to T_2:*

$$D_C(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_C(T_1, T_2)\}.$$

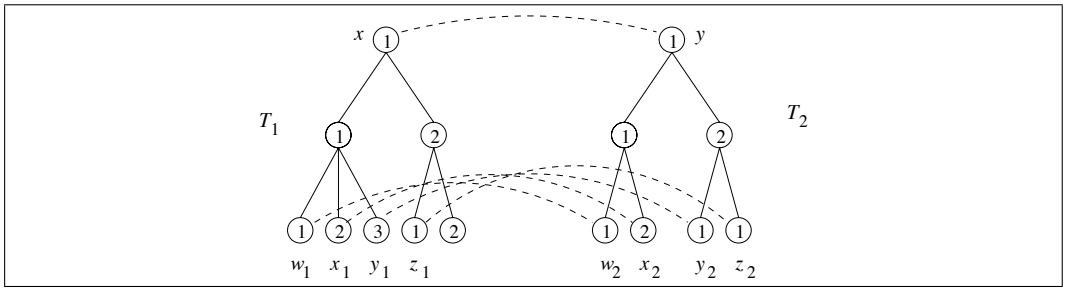

Figure 3.4: Examples of mappings between two semi-ordered trees T_1 and T_2. $M = \{(x, y), (w_1, w_2), (x_1, x_2), (y_1, y_2), (z_1, z_2)\}$ is a not valid mapping from T_1 to T_2: $y_1 \sqsubseteq z_1$ whereas $y_2 \not\sqsubseteq z_2$. $M \backslash \{(z_1, z_2)\}$ is a valid but not constrained mapping: $(w_1 \wedge x_1) \leq y_1$ whereas $(w_2 \wedge x_2) \not\leq y_2$. $M \backslash \{(y_1, y_2), (z_1, z_2)\}$ is a constrained mapping from T_1 to T_2.

Zhang [Zhang 1996] has described a dynamic programming algorithm to compute the constrained edit distance between two unordered trees using reductions of some subproblems to *minimum cost maximum flow problems* [Tarjan 1983]. Similarly, we proposed [Ouangraoua 2009a] a reduction of the problem of computing the constrained edit distance between semi-ordered trees to minimum cost maximum flow problems. The following recurrence formulas form the basis of the algorithm for the computation of the constrained edit distance between two semi-ordered trees.

Lemma 2 (Constrained edit distance between subtrees) *Let x, y be two vertices of T_1 and T_2 respectively, the constrained edit distance between $T_1[x]$ and $T_2[y]$ is:*

$$D_C(T_1[x], T_2[y]) = \min \begin{cases} D_C(F_1[x], F_2[y]) + \gamma(x, y) \\ \min_{y_k \in C(y)} \{D_C(T_1[x], T_2[y_k]) - D_C(\emptyset, T_2[y_k])\} + D_C(\emptyset, T_2[y]) \\ \min_{x_k \in C(x)} \{D_C(T_1[x_k], T_2[y]) - D_C(T_1[x_k], \emptyset)\} + D_C(T_1[x], \emptyset) \end{cases}$$

3.2.4 Global Comparison Between Unordered Quotiented Trees

For the case of the multiscale representation of plant architecture, we have introduced new edit distances between unordered quotiented trees [Ferraro 2003a, Ferraro 2003b] based on a comparison of support graph and edit operations that preserves equivalence relations. We have also proposed a symmetric approach by comparing quotiented trees at the more macroscopic scale [Ouangraoua 2009a]. Basically, quotiented trees refer to trees whose nodes are also trees. Edit score related to quotient vertices is thus defined as an edit score computation between the support subtrees of these vertices.

In the case of unordered trees, in order to find an optimal mapping between two unordered trees, as proposed in [Zhang 1996], the definition of a valid mapping M is sligthly modified to respect the following property:

Let $T_1[v] = (V_1[v], E_1[v])$ and $T_2[v] = (V_2[v], E_2[v])$ be two *unordered* trees, a *valid* mapping M from $T_1[v]$ to $T_2[w]$ is a set of ordered pairs of vertices $(x, y) \in V_1[v] \times$

$V_2[w]$ satisfying the constraints:

$$\begin{aligned} &\forall (x_1,x_2),(y_1,y_2),(z_1,z_2) \in M \ : \\ &\qquad x_1 = y_1 \quad \Leftrightarrow x_2 = y_2 \qquad (3.1) \\ &\qquad x_1 \leq y_1 \quad \Leftrightarrow x_2 \leq y_2 \qquad (3.2) \\ &x_1 \wedge y_1 < z_1 \quad \Leftrightarrow x_2 \wedge y_2 < z_2 \qquad (3.3) \end{aligned}$$

where $x_1 \wedge y_1$ represents the least common ancestor of x and y.

Let us now consider two quotiented trees $G_1 = (T_1, W_1, \pi_1)$ and $G_2 = (T_2, W_2, \pi_2)$ (if no confusion is possible π_1 and π_2 are denoted by π). Let M be a mapping from $T_1[v]$ to $T_2[w]$. M induces a mapping from tree $\mathcal{Q}(G_1)$ to tree $\mathcal{Q}(G_2)$, called *the quotient mapping*, denoted by $\mathcal{Q}(M)$, composed of pairs of vertices in $W_1 \times W_2$ and defined as:

$$(a,b) \in \mathcal{Q}(M) \Leftrightarrow \exists (z,t) \in M \text{ such that } \left\{ \begin{array}{rcl} \pi(z) & = & a \\ \pi(t) & = & b \end{array} \right.$$

We are then able to define a valid mapping between two quotiented trees as:

Definition 3 (valid mappings on quotiented trees) *Let $G_1[v] = (T_1, W_1, \pi_1)$ and $G_2[w] = (T_2, W_2, \pi_2)$ be two quotiented trees, a* valid mapping *M from $G_1[v]$ to $G_2[w]$ is a valid mapping from $T_1[v]$ to $T_2[w]$ such that $\mathcal{Q}(M)$ is also a valid mapping from $\mathcal{Q}(G_1)$ to $\mathcal{Q}(G_2)$.*

The set of valid mappings from $T_1[\pi(v)]$ to $T_2[\pi(w)]$ is denoted by $\mathcal{T}(\pi(v), \pi(w))$. Thus by definition M is a valid mapping from $G_1[v]$ to $G_2[w]$ if, and only if, M is in $\mathcal{T}(v,w)$ and $\mathcal{Q}(M)$ is in $\mathcal{T}(\pi(v), \pi(w))$. The set of valid mappings from $G_1[v]$ to $G_2[w]$ is denoted by $\mathcal{G}(v,w)$.

A dissimilarity measure between quotiented trees is then defined by the following optimization problem.

Problem 3 *Finding $\Gamma_{v,w}(M)$ minimum, such that M is a valid mapping from $G_1[v]$ to $G_2[w]$ satisfying definition 3:*

$$D(G_1[v], G_2[w]) = \min_{M \in \mathcal{G}(v,w)} \{\Gamma_{v,w}(M)\}$$

We have then proposed in [Ferraro 2003a] a polynomial algorithm (not described in this report) to compute this distance and solve problem 3.

3.3 Applications to Plant Analysis

These theoretical works would be useless if they were not motivated by biological applications. Among several applications (*cf.* section 3.4), I propose to present two of them leading my latest research. They intend to propose a better understanding of the repetiveness in plant development.

3.3.1 Compression of Plants

Plants are branching living organisms that develop throughout their lifetimes. Organs are created by small embryogenetic regions at the tip of each axis, called apical meristems (or simply meristems). During plant ontogeny, meristems develop branching structures that show remarkable organizations, made up of many similar organs at different scales: leaves, shoots, axes and branching systems of different sizes [Hallé 1978a, Harper 1986, Barthélémy 1989, Barthélémy 1991a, Room 1994]. Important progresses in the understanding of these growth processes have been made in the last decades by studying and quantifying real plants and their development under various environmental conditions. Two complementary approaches are being used. In simulation approaches, developmental models are built in order to reproduce the essence of the plant development with a few parameters in a simulation model, see [Prusinkiewicz 1998, Prusinkiewicz 2004, Deussen 2005] for reviews. On the other hand, in descriptive approaches, quantitative analyses of observed plant structures are tentatively used to reveal regularities or gradients hidden in the complex organization of plant structures, see [Godin 2005] for a review. In the recent years, both the simulation and the descriptive approaches are being combined to obtain more accurate models of plant development and assess them quantitatively against real data, e.g. [Renton 2006, Mündermann 2005, Guo 2006]. In the descriptive approach, a lot of techniques have been developed for analyzing distributions of events in the plant structure (e.g. [Godin 1997, Sinoquet 1997a]) or sequences of events along a branch or a meristem trajectory (e.g. [Guédon 2001, Guédon 2003]). Comparatively less attention has been paid to the development of methods for directly characterizing tree-like structures, e.g. [Ferraro 2000, Durand 2005, Ferraro 2005]. However, the natural organization of plants is primarily observed at the level of branching systems which qualitatively show strong internal similarities between their own parts. In many cases, this repetition of quasi-identical structures is accompanied with the impression that the branching systems are nested one into the others. Although these phenomena have been empirically described by botanists for decades, [Troll 1937, Arber 1950, Frijters 1976b, Barthélémy 1997], no algorithmic approach was developed yet to address this problem of recognizing similar, possibly nested, patterns in plant structures. We have aimed to develop such a computational framework and to illustrate its application to plant architecture analysis.

Characterizing the nested structure of rooted trees. Plant structures are usually represented by either ordered or unordered rooted trees [Prusinkiewicz 1990, Godin 1998] and a number of algorithms have been developed in computer science on such trees that have connections to our problem.

A first set of approaches makes it possible to compare quantitatively the structure of two trees. They are based on the use of an *edit-distance approach*, in which a metric is defined that reflects the minimum number of elementary edit operations necessary to transform one tree into the other. These algorithms solve different tree-to-tree comparison problems, such as defining a metric between trees, finding whether a tree is included into another one [Kilpeläinen 1995], finding the consensus tree between two trees (*ie.* the minimal tree that contains both) [Wang 1997], or the maximum

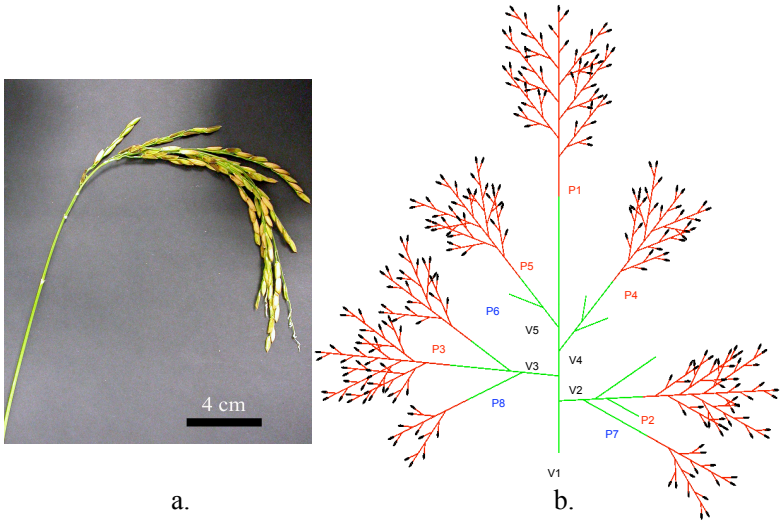

Figure 3.5: a) photo of a rice panicle (courtesy of Y. Caraglio). b) corresponding topological structure (reconstructed with the AMAPmod/VPlants open software [Godin 1998]).

common subtree [Wang 1999, Ouangraoua 2007], *etc.* Usually, to answer each question, a whole family of algorithms is developed to account for different characteristics of either the input trees (ordering of nodes, labelling) or the comparison problem (constraints on the valid edit operations, etc.). In the context of plant modeling, based on an original algorithm proposed by Zhang in 1993 [Zhang 1993, Zhang 1996], we studied in a previous work how to use and adapt such algorithms to compare plant architectures from a structural point of view [Ferraro 2000]. However, all these studies concentrate on the comparison between two different trees and usually pay no attention to characterizing the internal structure of a tree.

In a different spirit, the problem of studying internal repetitions of structures in a tree has been addressed by eliminating the structural redundancy appearing in trees (or in graphs). For this, similar parts in a tree are condensed, resulting in a *directed acyclic graph* (DAG). Such an approach was used in different domains. Based on a pioneering work by Akers [Akers 1978], Bryant [Bryant 1986] introduced one of the very first uses of such DAGs to represent efficiently the graph of Boolean functions. In this application, these function graphs are actually binary, ordered DAGs (each node of such a DAG has exactly two children, with a first and a second child). An algorithm is depicted that reduces these function DAGs to canonical DAGs from which all the structural redundancy of the initial DAG has been removed. It is closely related to the algorithm described in [Aho 1974] for testing whether two trees are isomorphic. DAG representations of trees are also much used in computer graphics where the process of condensing a tree into a graph is called *object instancing* [Hart 1991]. This process, first used by Sutherland in 1963 [Sutherland 1963],

makes it possible to share nodes representing scene objects and thus avoids the unnecessary duplication of different instances of the same graphical object. This efficient scheme is now commonly implemented as scene graphs in computer graphics applications. It allows to manipulate efficiently very huge scenes and was notably applied to the rendering of complex fractal scenes [Hart 1991, Hart 1992] and plant scenes, e.g. [Kay 1986, Hart 1991, Soler 2003].

Finally, our problem is also connected to the notion of structural self-similarity in trees. While self-similarity usually refers to purely geometric properties of objects (parts of an object are *geometrically* similar to the entire object up to a scaling factor), structural self-similarity attempts to capture an equivalent idea for structures and graphs. Different approaches of structural self-similarities have been tentatively proposed. Authors defined self-similarity by analysing either global branching parameters of trees e.g. [Tokunaga 1978, Viennot 1989, Peckam 1995, Zamir 2002] or topological structural properties of graphs [Kron 2004, Prusinkiewicz 2004]. Interestingly, in the context of studying efficient subtree isomorphisms, Greenlaw [Greenlaw 1996] introduced the definition of *nested trees*.

As such, nested trees are not strictly self-similar, but they have an internal recursive structure that makes them a closely related notion. Formally, self-nested trees are such that all their subtrees of a given height are isomorphic. We have called such trees *self-nested trees* to insist on their recursive structure and on their proximity to the notion of structural self-similarity. We derived from this notion the possibility to compress unordered trees without loss of information as more compact DAGs and showed that self-nested trees are those trees that can be compressed as linear DAGs. Based on concepts coming from these three areas, we have introduced in [Ferraro 2009a] a new algorithmic measure to quantify the degree of self-nestedness of trees. A compression of a rice panicle is presented in Figures 3.5 and 3.6.

Structural self-similarity was also introduced in the context of plant modelling by Prusinkiewicz [Prusinkiewicz 2004] based on botanical insights of Arber [Arber 1950]. This definition relies on the use of L-system rules, and was shown to grasp the essence of the pattern recursion through scales included in the idea of self-similarity. Subsequently, we have proposed an alternative approach to structural self-similarity in plants [Ferraro 2005], using edit-distance metrics to find recursively similarities between the high order branches of a plant and its trunk.

3.3.2 Self-Similarity in plants

As introduced in the previous section, repetitive patterns are readily noticeable in the growth and structure of many living organisms. In particular, modular organization is an essential element of the development and structure of plants[Hallé 1978b, White 1979, Barthélémy 1991b]. This is essential to the understanding of plant biology, and plays an important role in the formal analysis [Godin 1998] and simulation [de Reffye 1989, Prusinkiewicz 1990] of plants.

In some cases, the modules are arranged into compound, recursively nested, fractal-like structures, with similar patterns appearing at different scales [Mandelbrot 1983]

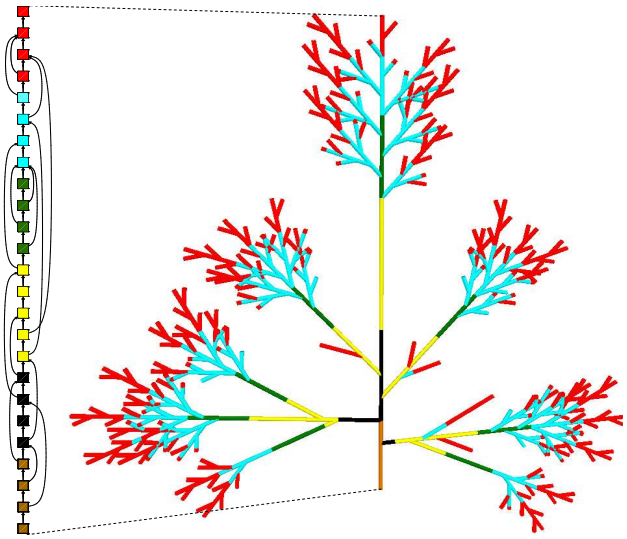

Figure 3.6: a. An element of the nearest embeded self-similar tree T^*. Backward projection of the differentiation states inferred from T^* onto the initial tree structure of the panicle.

(Fig. 3.7). In botany, an early study of such structures is due to Arber [Arber 1950]. Troll [Troll 1964] defined a compound inflorescence as a system consisting of the main florescence and paracladia that repeat the structure of the main florescence. This concept was later formalized by Fritjers and Lindemayer [Frijters 1976a], who defined paracladia as "branches which repeat the florescence of the main axis and which on their turn can give rise to paracladia of their own." Prusinkiewicz *et al.* [Prusinkiewicz 2001] introduced a related concept of branch mapping, according to which, "given two branches of the same order, the shorter branch is identical [...] to the top portion of the longer branch." Mündermann [Mündermann 2003] further studied these concepts as a basis for constructing three-dimensional plant models. More recently, Prusinkiewicz [Prusinkiewicz 2004] introduced the notion of topological self-similarity, which relates plant self-similarity to the theory of L-systems [Lindenmayer 1968, Herman 1975, Prusinkiewicz 1990].

In comparison to inflorescences, self-similar organization is less obvious in trees, which develop over a longer period of time, and therefore are more prone to the influences of the environment. Nevertheless, the branching systems of trees also result from repetitive processes, in which various meristems follow similar sequences of states and produce similar structures as a result. These sequences have been characterized by biologists in such terms as *age state* [Gatsuk 1980], *morphogenetic program* [Nozeran 1984], and *physiological age* [Barthélémy 1997]. This last notion made it possible, for example, to exploit similarities in the development and structure of *Zelkova serrata* (Japanese elm) in the construction of a simulation model [de Reffye 1989], where the sequence of physiological ages of a typical meristem in the plant served as a template (called the *reference axis*) for all plant meristems.

Figure 3.7: Examples of plants showing remarkably self-similar branching structures: A fern leaf, a compound inflorescence (lilac), and a romanesco broccoli.

Self-similarity, like symmetry, is not easily quantifiable: it is usually considered to be either absent or present. Real plants, however, may be self-similar only to some degree. We have proposed [Ferraro 2005] a definition of self-similarity in branching structures, and describe a procedure for quantifying it in measured or simulated plants. This procedure is based on a method for comparing tree-like structures and extended to the problem of comparing plants [Ferraro 2000].

3.4 Paper Presentation

3.4.1 Tree-Graph Comparison

In the papers presented in this section we have proposed dynamic programming algorithms to compare trees using a constrained edit distance. We have introduced in [Ferraro 2003a] and [Ferraro 2003b] first steps for the comparison of multi-scale tree-graphs.

[Ferraro 2003a] **Pascal Ferraro** and Christophe Godin. *An edit distance between quotiented graphs.* Algorithmica, vol. 36, pages 1–39, 2003.

[Ferraro 2003b] **Pascal Ferraro** and Christophe Godin. *Optimal mappings with minimum number of connected componanents in tree-to-tree comparison problems.* Journal of Algorithms, vol. 48, pages 385–406, 2003.

In [Ouangraoua 2009a], we propose a formal definition of a new class of trees called semi-ordered trees and a polynomial dynamic programming algorithm to compute a constrained edit distance between such trees. The core of the method relies on a similar approach to compare unordered [Zhang 1996] and ordered trees [Zhang 1989]. The method was then applied to evaluate the similarity between architectures of apple trees [Segura 2008].

[Ouangraoua 2009a] Aïda Ouangraoua and **Pascal Ferraro**. *A constrained edit distance algorithm between semi-ordered trees.* Theoretical Computer Science, vol. 410, no. 8-10, pages 837–846, 2009.

3.4.2 Applications

The following papers focuses on the methods dedicated to the comparison between plant architectures.

A first analysis has been proposed in [Ferraro 2000].

[Ferraro 2000] **Pascal Ferraro** and Christophe Godin. *A distance measure between plant architectures.* Annals of Forest Science, vol. 57, pages 445–461, 2000.

[Segura 2008] and [Ouangraoua 2008] present the last techniques available to compare plant architectures depending on their formal representation. Their relative advantages / disadvantages are presented through biological examples. The application of both global and local comparison methods showed that distances between trees were similar with most of the methods, except the end-space free method. Several edit distance algorithms for the comparison of plant architectures are presently available on an open-source platform: OpenAlea. We have shown that for a given formal tree-graph representation, both global and local comparison methods can be applied depending on the biological context and goal. However, the methods are more or less appropriate depending on the heterogeneity of the topological size of the compared plants; the traits that must be taken into account.

[Segura 2008] Vincent Segura, Aïda Ouangraoua, **Pascal Ferraro** and Evelyne Costes. *Comparison of tree architecture using tree edit distances: application to two-year-old apple hybrids.* Euphytica, vol. 161, pages 155–164, 2008.

[Ouangraoua 2008] Aida Ouangraoua, Vincent Segura, Evelyne Costes and **Pascal Ferraro**. *Methods to evaluate similarity between plant architectures based on the comparison of their tree structured representation.* BMC Systems Biology, vol. submitted, page 8, 2008.

3.4.3 Self-similarity in plants

Self-similarity of plants has attracted the attention of biologists for at least 50 years, yet its formal treatment is rare, and no measure for quantifying the *degree* of self-similarity currently exists. We have proposed in [Ferraro 2005] a formal definition and measures of self-similarity, tailored to branching plant structures based on the comparison between trees.

[Ferraro 2005] **Pascal Ferraro**, Christophe Godin and Przemylaw Prusinkiewicz. *Toward a quantification of self-similarity in plants.* Fractals, vol. 13, no. 2, pages 1–25, 2005.

The article [Ferraro 2009] introduces and develops the notion of self-nestedness of trees. Self-nested trees are such that all their subtrees of a given height are isomorphic. We show that these trees present remarkable compression properties, with high compression rates. In order to measure how far a tree is from being a self-nested tree, we then study how to quantify the degree of self-nestedness of any tree.

[Ferraro 2009] **Pascal Ferraro**, Christophe Godin and Przemylaw Prusinkiewicz. *Quantifying the degree of self-nestedness of trees. Application to the structural analysis of plants.* IEEE/ACM Transactions on Computation Biology and Bioinformatics, vol. in press, page 16, 2009.

RNA Secondary Structures 4

4.1 Context

During the last few years, our knowledge about the role of RNA molecules in the cell life process has been considerably modified. Notably, it has been stated that small non coding RNAs (such as small nuclear RNAs, small nuclear RNAs, microRNAs, siRNAs, ...) play a very important role in nearly all aspects of gene regulation. Additionally, about 60% of the mouse genome (thus probably of most mammal genomes) is transcribed into RNA, while less than 5% is translated into proteins. This gives place to a large universe of "RNA dark matter" that is worth being explored (See *e.g.* [sci 2005, Mendes Soares 2006]).

Therefore, there is a crucial need for efficient computer-based methods for helping to handle the structural modelling and functional assignment of RNAs. From a historical perspective, many computational methods have been proposed to deal with the RNA folding problem. But there is still no reference tool to compare RNA molecules. The (pairwise or multiple) structural comparison of RNA molecules is among the most important problems in this area. The pairwise RNA comparison problem consists both in computing a distance between two RNA molecules that

reflects their evolutionary distance, and in giving a correspondence between the structural elements of the two molecules As for sequences, structural comparison can be either global or local. Notably, fast and biologically relevant pairwise comparison tools are urgently needed in order to classify and to annotate the huge amount of new non coding RNAs found by from genomic-scale analyses. The multiple RNA comparison problem consists in extending the pairwise comparison problem to a full family of homologous RNA sequences. This allows, for example, to determine the shared structural features of a given functional family, or to detect a very conserved, thus probably biologically active, common structural site.

The biological function of a RNA molecule is very closely related to its spatial structure (tertiary structure), and two rather different RNA sequences may lead to very similar structures. This is why the classical sequence comparison tools are not relevant for RNA molecules. From the algorithmic point of view, the pairwise structure comparison problem has been a very active research area during the last 20 years. Briefly, the problem is as follows : The RNA structure is modelled as a graph whose nodes are the nucleotides and whose edges are the chemical bonds between them. A set of basic operations on these graphs is given, with a score for each operation. Then, there are two major ways to compare two structures S_1 and S_2 : edition and alignment. The edition problem consists in finding a best-scoring sequence of operations which changes S_1 into S_2. The alignment problem consists in finding a "superstructure" S_3 which contains S_1 and S_2 as substructures, in such a way to maximise the sum of the edition scores between S_1 an S_3 and between S_2 and S_3. The algorithmic complexity of these problems mainly depends on:

- the model of RNA structure that is taken into account. Briefly, the graph which represents the whole structure is called the tertiary structure of the molecule. A secondary structure is a partial representation of the tertiary structure where only edges corresponding to the strongest chemical bounds (Watson-Crick pairs, and possibly wobble pairs) are present. If a secondary structure does not contain crossing edges, so-called pseudo-knots, then it can be represented by a rooted ordered tree.

- the set of operations that are allowed. A set of biologically relevant edit operations on bases and base-pairs in RNA structures have been defined in [Jiang 2002]: the classical substitution, insertion, and suppression operations, plus some very important operations for RNA, such as pairing or unpairing nucleotides.

In its full generality, the RNA structure comparison problem is difficult: comparing tertiary structures with the above complete set of operations has been proved to be NP-complete [Jiang 2002] for the edition problem, and for the alignment problem [Blin 2006]. Moreover, Blin *et al.* [Blin 2007] have proved that the edition problem remains NP-complete even for two secondary structures without pseudo-knots. Meanwhile, during the last 20 years much effort has been done on designing polynomial comparison methods for secondary structures without pseudo-knots, either by heuristics approaches, or by restrict-

ing the set of allowed operations, or by fixing constraints on their scores (see [Vauchaussade de Chaumont 1985, Zhang 1989, Jiang 1994, Höchsmann 2003] among others). Some heuristics have also been developed for pseudo-knotted structures (such as [Jiang 1994, Collins 2000]). Based on some of these works, a few programs are publicly available : RNAdistance in the Vienna RNA Package (`http://www.tbi.univie.ac.at/ ivo/RNA/RNAdistance.html`), RNAmatch and RNA-align (`http://www.csd.uwo.ca/ kzhang/rna/`), RNAForester (`http://bibiserv.techfak.uni-bielefeld.de/rnaforester/`).

The multiple comparison problem has been less studied, probably because of its greater complexity. Regarding the theoretical complexity, the problem is NP-complete for all reasonable edit models. From a practical point of view, few multiple structural alignment heuristics have been developed, *e.g.* [Wang 2004, Höchsmann 2004, Siebert 2005].

Up to now, RNA comparison software are rarely used by biologists. And when they are, a long and thorough subsequent work must be done "by hand" in order to get precise results. This is due partly to the fact that most of these tools are rather recent. And all these approaches suffer from drawbacks, notably:

- The operations are defined at the nucleotide or base-pair level only, even though it is known that larger structural elements must be taken into account [Kachouri 2005, Allali 2005b].
- Even at the nucleotide and base-pair level, none of the algorithms takes into account the whole set of relevant operations with independent scores.
- The scores of the operations are fixed in a somewhat arbitrary way (although some work has been done recently on statistical approaches for computing them on biological data, see *e.g.* [Klein 2003].

Recently, a number of progresses in this area have been done by the teams in the present project. In particular, my researches have focused on original approaches allowing to better take biological properties into account. In this context, we have developed new algorithms, and even a new pairwise comparison software [Ouangraoua 2007, Ituarte 2006, Ouangraoua 2009b].

4.2 Formal Representations of RNA Secondary Structures

In the field of RNA secondary structure comparison, we are led back to the comparison of ordered labeled trees. A tree is said labeled if a label is associated to each node. A tree is said ordered if the edges incident to a given node are ordered cyclically. These are therefore trees for which the left,to,right order among the sibling nodes is significant.

The RNA structure can hence be decomposed in a way where the labels of the nodes represent:

1. either structural elements (sequences of paired bases, hairpins, loop, bulges, stems, ...),
2. or nucleic acids (A, C, G, U), Watson-Crick pairs, Wobble pairs, etc.

In that way, considering only the structural patterns, this tree representation can be rough (situation (1)) or refined (situation (2)). Below are represented:

- a RNA secondary structure on the left, the dots representing the nucleic acids,
- a relatively rough tree representation of this structure on the center where the following labels are used: a node labeled S(tem) represent a sequence of paired bases, a node labeled B a bulge, a node labeled I an inside loop, a node labeled M a multi-branch loop and a node labeled H an hairpin.
- a coding of the same structure on the right: "coding" in the way that the tree (with a good labeling of its nodes) makes it possible to come back to the secondary structure.

Both tree models of a RNA secondary structure represent information in the molecule at two distinct scales and are gathered in a single model: a quotiented tree (Fig. 4.2).

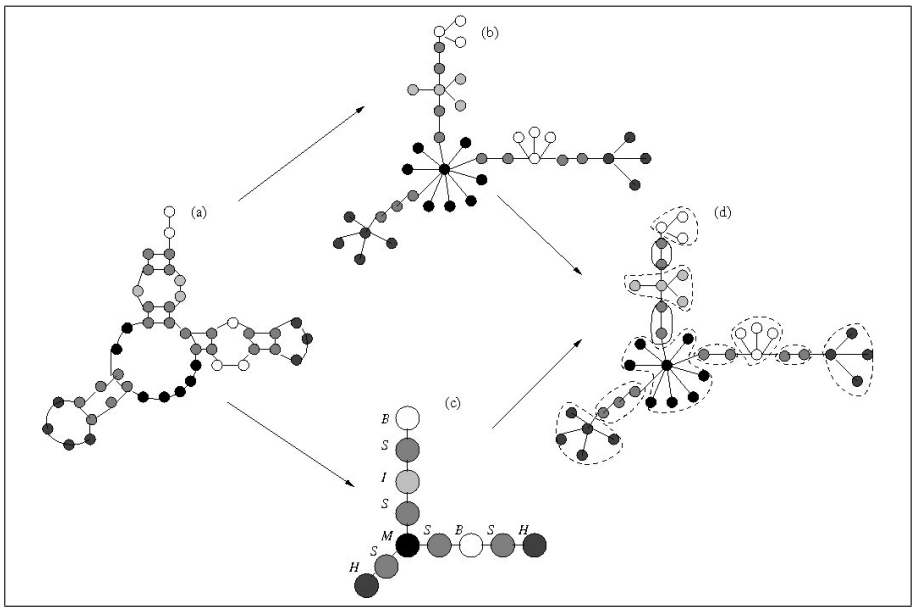

Figure 4.1: The RNA secondary structure (a) is modeled by a microscopic tree (b) and a macroscopic one (c). Both trees are gathered to obtain the quotiented tree representation (d)

RNA secondary structures used to be compared using either microscopic [Zhang 1989] or macroscopic [Shapiro 1990] tree representations. A quotiented

comparison [Ouangraoua 2007, Ouangraoua 2009b] will take into account structural informations at both scales.

4.3 RNA Secondary Structures Comparison

Ordered trees are commonly used to model the topological structure of biological data (*e.g.* plants [Godin 1998], RNA secondary structures [Shapiro 1990]) and then to evaluate similarity between these data. Over the last 30 years, many ordered labeled tree comparison algorithms have been developed [Selkow 1977, Tai 1979, Zhang 1989]. These algorithms were introduced by Selkow [Selkow 1977] and are based on the notion of edit distance or alignment score, which have since become classic approaches in the frame of sequence comparison [Wagner 1974]. Both notions are equivalent in the case of sequence comparison, but lead to two different problems in the case of tree comparison. We have focused in [Ouangraoua 2007, Ouangraoua 2009b] on multi-scale extensions of Zhang and Shasha's algorithm [Zhang 1989] which is the standard when dealing with ordered labeled tree comparison. These extensions are currently used to evaluate the similarity between two multiscale representations of RNA secondary structures.

To take into account the multi-scale nature of RNA secondary structures, two different approaches have been recently proposed. Allali and Sagot [Allali 2005a] introduced a new data structure, called MiGaL for Multiple Graph Layers, composed of various graphs linked together by relations of abstraction/refinement. This structure is used for representing information that can be described at different levels of abstraction, each level corresponding to *a graph* (and not necessarily a tree). The comparison of two MiGaL is then performed as a step-by-step comparison starting with the most *abstract level*. The result of the comparison at a given step is then communicated to the next step using a special colouring scheme. MiGaLs are used for comparing RNA secondary structures that may be seen at different levels of detail, going from the sequence of nucleotides, single or paired with another to participate in a helix, to the network of multiple loops.

In the same way, in [Ouangraoua 2007] we have proposed a top-down approach to compare RNA secondary structures based on two levels of abstraction, each level being *an ordered tree*. This algorithm is an extension of Zhang and Shasha's algorithm [Zhang 1989] to locally compare quotiented ordered trees [Ouangraoua 2007]. A quotiented ordered tree [Godin 1998] is an ordered tree with an equivalence relation defined on the set of vertices, such that the resulting quotient graph is also an ordered tree. It can thus be considered as a self-similar structure represented by trees on two different scales: basically, quotiented trees refer to trees (the macroscopic scale) whose vertices are also trees (the microscopic scale). Our approach [Ouangraoua 2007] then consists of comparing trees at the macroscopic scale by defining the edit cost related to macroscopic vertices as the edit distance computation between sub-trees of these vertices. However, during the comparison, equivalence relations on vertices are not preserved.

On the other hand, for plant modeling purposes, in [Ferraro 2003a] (see section 3) we have proposed a bottom-up solution to the comparison of *unordered* quotiented trees which preserves equivalence relations. Based on this approach, and by introducing new notations, we present hereafter an edit distance between quotiented ordered trees which can be used to compute an optimal sequence of edit operations at the most *microscopic scale* and *preserving equivalence relations on tree vertices.*

4.3.1 Comparison between Ordered Trees

Let T be an ordered tree, referring to the notations of [Shapiro 1990, Ouangraoua 2007] and presented in the previous Chapter, let $x_1 < x_2 < \ldots < x_k$ be the vertices of $T[x_k]$, $F[x_1 \ldots x_i]$ is the forest consisting in the subtrees of $T[x_k]$ restricted to the vertices $x_1, x_2, \ldots, x_i$. Particularly, $F[x_1 \ldots x_k]$ is the whole tree $T[x_k]$. By convention, if $x_k < x_1$ then $F[x_1 \ldots x_k]$ represents θ the empty tree.

Let F_1 and F_2 be two ordered forests and let x_1, y_1 and x_2, y_2 be vertices of F_1 and F_2 respectively. Zhang *et al.* [Zhang 1989] proposed a general recurrence formula for computing the similarity score $S(F_1[x_1 \ldots y_1], F_2[x_2 \ldots y_2])$ between ordered forest:

$$S(F_1[x_1 \ldots y_1], F_2[x_2 \ldots y_2]) = max \begin{cases} S(F_1[x_1 \ldots y_1 - 1], F_2[x_2 \ldots y_2]) + s(\alpha(y_1), \lambda) \\ S(F_1[x_1 \ldots y_1], F_2[x_2 \ldots y_2 - 1]) + s(\lambda, \alpha(y_2)) \\ S(F_1[x_1 \ldots l(y_1) - 1], F_2[x_2 \ldots l(y_2) - 1]) \\ + \; S(F_1[l(y_1) \ldots y_1 - 1], F_2[l(y_2) \ldots y_2 - 1]) \\ \quad + s(\alpha(y_1), \alpha(y_2)) \end{cases} \tag{4.1}$$

Note, if y_1 (resp. y_2) is an ancestor of x_1 (resp. x_2), then $F_1[x_1 \ldots y_1]$ (resp. $F_1[x_2 \ldots y_2]$) is a tree and $F_1[x_1 \ldots l(y_1) - 1]$ (resp. $F_2[x_2 \ldots l(y_2) - 1]$) is the empty tree.

4.3.2 Local similarity

In many cases trees share only a limited region of similarity. This may be a common domain or simply a short region of recognizable similarity. This case is dealt with by so-called *local mapping* in an algorithm developed by Smith and Waterman [Smith 1981] to evaluate local similarity between strings. Local similarity aims at identifying the best pair of regions, one from each string, such that the optimal (global) similarity of these two regions is the best possible. This relies on a scoring scheme that maximizes a similarity score because otherwise an empty sequence of edit operations would always yield the smallest score. Naively, the algorithm to compute the local similarity would need to inspect every pair of regions and apply a global comparison algorithm to it. The decisive idea of Smith and Waterman was to find for any prefix of the sequences a suffix with a maximal score. We have proposed a generalization of this algorithm to evaluate local similarity between ordered and quotiented trees [Ouangraoua 2007].

A first algorithm for finding similar regions in trees has been previously proposed by Höshmann *et al.* [Höchsmann 2003]. It build upon the tree alignment algorithm for ordered trees given by Jiang *et al.* [Jiang 1994]. This algorithm is used for

evaluating local similarity in RNA secondary structures. However, since edition of trees and tree alignments are different concepts and lead to different algorithms, this method is not discussed in this Chapter. Note that in [Wang 1998, Wang 1999], Wang *et al.* solved a similar problem consisting of finding the largest approximately common substructures in ordered labeled trees. Given two trees T_1 and T_2 and an additional parameter δ, their algorithm determine the largest sub-trees U_1 and U_2 of respectively T_1 and T_2 whose distance is at most δ. It is based on the computation of the global edit distance between two trees, and is therefore a minimization problem. Our variation does not use any additional parameter and thus cannot be solved as a minimization problem.

4.3.2.1 Local comparison between ordered trees

The computation of a local similarity allows to detect local conserved areas between both trees. The solution of such a problem is based on the notion of prefix mapping between trees.

Definition 4 *Let T be a tree rooted in r, any* partial *sub-tree of T rooted in r is called* a prefix of T *or* a T-prefix. *By convention, the empty tree θ is a T-prefix.*

Note that a particular prefix of T rooted in r is $T[r]$ itself. Let T_1 and T_2 be two trees and let x_1 and x_2 be two vertices of T_1 and T_2, the set of $T_1[x_1]$-prefixes and $T_2[x_2]$-prefixes are respectively denoted by $\mathcal{T}_1[x_1]$ and $\mathcal{T}_2[x_2]$.

A similar definition can be proposed for a forest:

Definition 5 *Let F be a forest made of n trees $T_1, \ldots, T_n$ respectively rooted in $r_1, r_2, \ldots, r_n$. A* F-prefix *is a sub-forest of F made of any prefixes of $T_1, \ldots, T_n$.*

The local prefix mapping problem for a given pair x_1, x_2 of vertices is to find a (possibly empty) prefix ρ_1 of $T_1[x_1]$ and a (possibly empty) prefix ρ_2 of $T_2[x_2]$ (Figure 4.2.a) such that the score of the optimal sequence of edit operations transforming ρ_1 into ρ_2 is the maximum over all scores of sequences of edit operations between prefixes of $T_1[x_1]$ and $T_2[x_2]$.

The score of the sequence solving the optimal local prefix mapping problem (called local score) for a given pair x_1, x_2 of vertices is denoted by $LS(T_1[x_1], T_2[x_2])$:

$$LS(T_1[x_1], T_2[x_2]) = max\{S(\rho_1, \rho_2), (\rho_1, \rho_2) \in \mathcal{T}_1[x_1] \times \mathcal{T}_2[x_2]\}.$$

Note that a local prefix problem between two forests $F_1[x_1 \ldots y_1]$ and $F_2[x_2 \ldots y_2]$ is similarly defined as:

$$LS(F_1[x_1 \ldots y_1], F_2[x_2 \ldots y_2]) = max\{S_F(\rho_1, \rho_2), (\rho_1, \rho_2) \in \mathcal{F}_1[x_1 \ldots y_1] \times \mathcal{F}_2[x_2 \ldots y_2]\}.$$

where $\mathcal{F}_1[x_1 \ldots y_1]$ and $\mathcal{F}_2[x_2 \ldots y_2]$ represent respectively the set of $F_1[x_1 \ldots y_1]$-prefixes and $F_2[x_2 \ldots y_2]$-prefixes (Fig. 4.2.b).

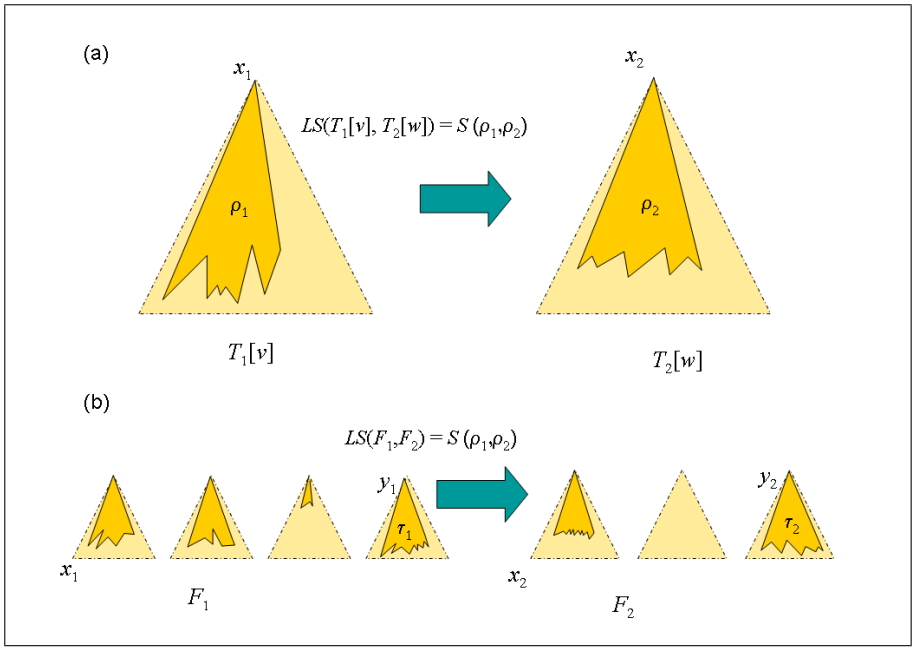

Figure 4.2: Local prefix definition for (a) two trees and (b) two forests

Local similarity between two trees is then defined as the score of the best pair of local prefixes in trees T_1 and T_2:

Theorem 4

$$LS(T_1, T_2) = max\{LS(T_1[x_1], T_2[x_2]), (x_1, x_2) \in V_1 \times V_2\}.$$

So, in order to evaluate local similarity, the algorithm needs to find maximum similarity between prefixes of $T_1[x_1]$ and $T_2[x_2]$, for any pair of vertices (x_1, x_2) of $V_1 \times V_2$, and then to determine the best pair of vertices x_1^{Max}, x_2^{Max} of T_1 and T_2.

The recurrence formula for the computation of the local score is then given by the following proposition [Ouangraoua 2007]:

Proposition 5 *Let T_1 and T_2 be two trees and let x_1, y_1 and x_2, y_2 be vertices of T_1 and T_2 respectively, with $x_1 < y_1$ and $x_2 < y_2$:*

$$LS(F_1[x_1 \dots y_1], F_2[x_2 \dots y_2]) = \max \begin{cases} LS(F_1[x_1 \dots l(y_1) - 1], F_2[x_2 \dots l(y_2) - 1]) \\ LS(F_1[x_1 \dots y_1], F_2[x_2 \dots l(y_2) - 1]) \\ LS(F_1[x_1 \dots l(y_1) - 1], F_2[x_2 \dots y_2]) \\ LS(F_1[x_1 \dots l(y_1) - 1], F_2[x_2 \dots l(y_2) - 1]) \\ \quad + LS(F_1(l(y_1) \dots y_1 - 1], F_2[l(y_2) \dots y_2 - 1]) \\ \quad + s(\alpha(y_1), \alpha(y_2)) \\ LS(F_1[x_1 \dots y_1], F_2[x_2 \dots y_2 - 1]) + s(\lambda, \alpha(y_2)) \\ LS(F_1[x_1 \dots y_1 - 1], F_2[x_2 \dots y_2]) + s(\alpha(y_1), \lambda) \end{cases} \quad (4.2)$$

The complexity of this algorithm [Ouangraoua 2007] is the same as Zhang-Shasha's algorithm and is bounded by $O(|T_1| \times |T_2| \times min(h(T_1), l(T_1)) \times min(h(T_2), l(T_2)))$

where, for any i in $\{1, 2\}$, $h(T_i)$ and $l(T_i)$ represent respectively the height and the number of leaves of the tree T_i. The average complexity of the algorithm is on the order of $|T_1|^{3/2} \times |T_2|^{3/2}$ ([Dulucq 2003]). The space complexity is in $O(|T_1| \times |T_2|)$.

This recurrence formula is not totally equivalent to Smith and Waterman's computation [Smith 1981]. Let us consider the problem of local similarity between sequences as a tree edit problem. Any sequence with a length n could be represented following two different graphs, *i.e.* as a graph with n vertices and such that any vertex has only one child except one, the leaf, or as a graph made of a root and exactly $n-1$ children. In the first case, since any vertex (except the leaf) has only one child, the local score $LS(F_1[x_1 \dots l(y_1) - 1], F_2[x_2 \dots l(y_2) - 1])$ is always equal to zero. Similarly, three first line of equation 4.2 are equivalent to zero. Then the computation of local similarity between trees leads to the same equivalence relation than Smith and Waterman's one [Smith 1981]. However, since order relations are not taken into account to define the notion of optimal T-prefix, the second case does not lead to the same result. To get Smith and Waterman result, a sequence of symbol should be represented using the first model.

4.3.3 Comparison Between Ordered Quotiented Trees

4.3.3.1 Valid Mapping Between Quotiented Ordered Trees

The tree-to-tree correction problem consists in determining the distance between two trees measured by the optimal sequence of edit operations (namely substitution, insertion and deletion) needed to transform one tree into the other.

As introduced in the previous Chapter, mapping between two ordered trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ is usually associated to each sequence of edit operations transforming T_1 into T_2. A *mapping* [Tai 1979] from T_1 to T_2 is a triple (M, T_1, T_2) where M is a set of ordered pairs of vertices (x, y) with $x \in V_1$ and $y \in V_2$ where the following properties hold for any two pairs (x_1, x_2) and (y_1, y_2) in M:

$$x_1 = y_1 \quad \Leftrightarrow \quad x_2 = y_2, \tag{4.3}$$

$$x_1 \sqsubset y_1 \quad \Leftrightarrow \quad x_2 \sqsubset y_2, \tag{4.4}$$

$$x_1 < y_1 \quad \Leftrightarrow \quad x_2 < y_2. \tag{4.5}$$

Let M be a mapping from T_1 to T_2 and let $\overline{M_1}$ (resp. $\overline{M_2}$) be the set of vertices of T_1 (resp. T_2) which do not appear in a pair of M. The cost of M is then defined by:

$$\gamma(M) = \sum_{(x_1,x_2)\in M} \gamma(x_1, x_2) + \sum_{x_1 \in \overline{M_1}} \gamma(x_1, \lambda) + \sum_{x_2 \in \overline{M_2}} \gamma(\lambda, x_2).$$

The notion of mapping is extended to quotiented ordered trees. Let $Q_1 = (T_1, W_1, \pi)$ and $Q_2 = (T_2, W_2, \pi)$ be two quotiented ordered trees. Let (M, T_1, T_2) be a mapping from T_1 to T_2. The mapping induced from M on the quotient trees $\pi(T_1)$ and $\pi(T_2)$ is denoted by $(\pi(M), \pi(T_1), \pi(T_2))$ where $\pi(M)$ is the set of all pairs of vertices (X_1, X_2) of $\pi(T_1)$ and $\pi(T_2)$ such that:

$$(X_1, X_2) \in \pi(M) \Leftrightarrow \exists (x_1, x_2) \in M \ s.t. \ \pi(x_1) = X_1 \ and \ \pi(x_2) = X_2.$$

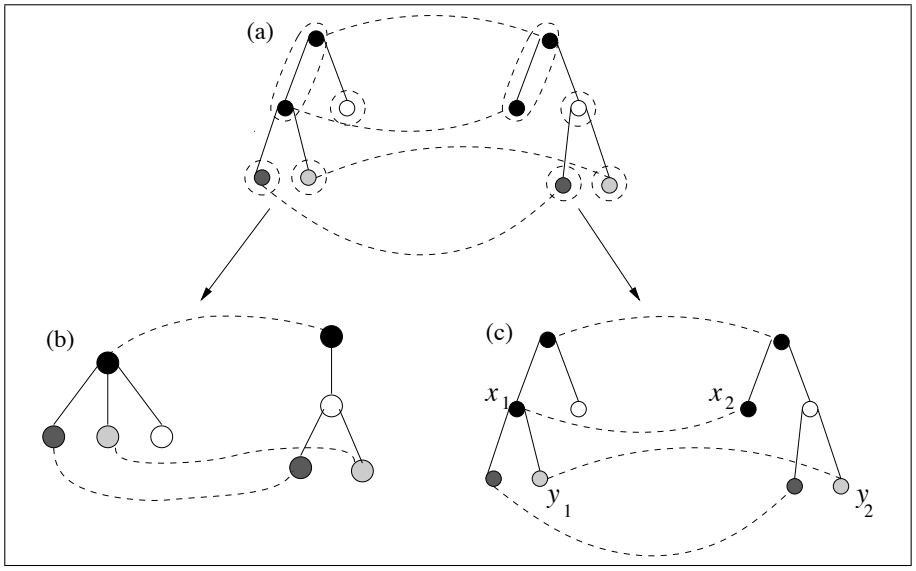

Figure 4.3: (a) An optimal mapping obtained from Ouangraoua *et al.*'s algorithm [Ouangraoua 2007]. Vertex mappings are represented by dotted lines. Using this algorithm the quotient vertices are mapped (b) which induces a mapping on the support graph (c). Since $x_1 \sqsubset y_1$ and y_2 is not a descendant of x_2, the mapping on the support graph does not preserve the ancestor-descendant relationship.

This definition means that the pair (X_1, X_2) belongs to the mapping induced by M on the quotiented tree if and only if there exists at least one vertex in X_1 and one in X_2 such that they are both mapped. However, with such a definition a quotiented vertex could belong to several pairs of the induced mapping. In this case the mapping induced by M does not respect constraint (1).

In [Ouangraoua 2007] we proposed an algorithm to compute the edit distance between two quotiented ordered trees. Basically, a quotiented tree is a tree such that vertices are also trees and then an edit score related to quotient vertices is defined as an edit score between the support sub-trees of these vertices. However, this method does not allow to compute an optimal mapping satisfying constraints (1), (2) and (3) simultaneously at both scales. Indeed, as mentioned previously in some cases, the minimal cost mapping does not respect ancestor-descendant and/or order relationships in support trees (Fig. 4.3). We have then proposed to extend the definition of mappings in order to preserve these constraints at every scale.

A *valid mapping* from Q_1 to Q_2 is thus a mapping from T_1 to T_2 such that the corresponding induced mapping on quotient trees is also a mapping.

Definition 6 (Valid mapping) *A valid mapping between two quotiented ordered trees $Q_1 = (T_1, W_1, \pi)$ and $Q_2 = (T_2, W_2, \pi)$ is a triple (M, Q_1, Q_2) such that M is a mapping from T_1 to T_2 and $\pi(M)$ is a mapping from $\pi(T_1)$ to $\pi(T_2)$. For any pairs (x_1, x_2) and (y_1, y_2) in M, (x_1, x_2) and (y_1, y_2) verify relations (1), (2), (3)*

and:

$$\pi(x_1) = \pi(y_1) \quad \Leftrightarrow \quad \pi(x_2) = \pi(y_2), \tag{4.6}$$
$$\pi(x_1) \sqsubset \pi(y_1) \quad \Leftrightarrow \quad \pi(x_2) \sqsubset \pi(y_2), \tag{4.7}$$
$$\pi(x_1) < \pi(y_1) \quad \Leftrightarrow \quad \pi(x_2) < \pi(y_2). \tag{4.8}$$

The set of valid mappings (or simply mappings) from Q_1 to Q_2 is denoted by $\mathcal{M}(Q_1, Q_2)$. If there is no confusion, we denote (M, Q_1, Q_2) by M and any pair $(x_1, x_2) \in M$ is denoted by $x_1 \in M$ and $x_2 \in M$. In addition, the set of vertices of $\pi(T_1)$ (resp. $\pi(T_2)$) which do not appear in a pair of $\pi(M)$ is denoted by $\overline{\pi(M_1)}$ (resp. $\overline{\pi(M_2)}$).

Since a quotiented mapping is a valid mapping between ordered trees, the cost of a quotiented mapping is properly defined, thus:

Definition 7 (Edit distance) *The distance between Q_1 and Q_2 is defined by:*

$$D(Q_1, Q_2) = \min\{\gamma(M) \mid M \in \mathcal{M}(Q_1, Q_2)\}.$$

These definitions are directly extended to quotiented ordered forests.

In [Ouangraoua 2007] we have proposed a top-down approach to compare RNA secondary structures based on two levels of abstraction, each level being an ordered tree. This algorithm is an extension of Zhang and Shasha's algorithm to locally compare quotiented ordered trees, presented in the previous section. The following introduce a bottom-up approach which has been presented in [Ouangraoua 2009b].

4.3.4 Edit Distance Computation

This section presents recurrence relations computing an optimal valid mapping between quotiented ordered trees which *preserves one-to-one, ancestor-descendant and order relationships on tree vertices* at both scales. The solution proposed is slightly different from the one we proposed in [Ferraro 2003a] (*cf.* Chapter 3 of this report) to measure the distance between quotiented unordered trees. In their solution, the distance is computed by studying optimal mappings at the support scale and then by checking if these mappings conform to the constraints at the quotient scale. This approach leads us to evaluate a huge number of combinatoric cases. In [Ouangraoua 2007], we have also proposed a symmetric approach by firstly constraining the mapping at the most macroscopic scale (*i.e.* the quotient scale) and then computing the final optimal mapping at the support scale.

Let $Q_1 = (T_1, W_1, \pi)$ and $Q_2 = (T_2, W_2, \pi)$ be two quotiented ordered trees, similarly to the computation of a distance between ordered trees [Zhang 1989], we propose to compute the distance between Q_1 and Q_2 using the dynamic programming principle. To take into account the new constraints on valid mappings, we introduced a constrained edit distance that reminds if two quotient vertices are mapped.

Definition 8 (Constrained edit distance) *Let C_1 and C_2 be two quotient vertices of Q_1 and Q_2 respectively, the* constrained edit distance $D_C(Q_1, Q_2, C_1, C_2)$ *between Q_1 and Q_2 is the minimal cost of a valid mapping M from Q_1 to Q_2 such that the quotient vertices C_1 and C_2 are mapped by M:*

$$D_C(Q_1, Q_2, C_1, C_2) = \min\{\gamma(M) \mid M \in \mathcal{M}(Q_1, Q_2) \text{ and } (C_1, C_2) \in \pi(M)\}.$$

The constrained edit distance is extended in order to indicate if a quotient vertex is not mapped:

$$\begin{aligned} D_C(Q_1, Q_2, \emptyset, C_2) &= \min\{\gamma(M) \mid M \in \mathcal{M}(Q_1, Q_2) \text{ and } C_2 \in \overline{\pi(M_2)}\}, \\ D_C(Q_1, Q_2, C_1, \emptyset) &= \min\{\gamma(M) \mid M \in \mathcal{M}(Q_1, Q_2) \text{ and } C_1 \in \overline{\pi(M_1)}.\} \end{aligned}$$

By extension, we will use the following notation:

$$D_C(Q_1, Q_2, \emptyset, \emptyset) = D(Q_1, Q_2).$$

This constrained edit distance between quotiented ordered trees (or forests) will be used to recursively compute the distance between Q_1 and Q_2. The algorithm, similarly to [Zhang 1989], considers the distance between two quotiented ordered forests in its intermediate steps. Let x_1 and y_1 (resp. x_2 and y_2) be two vertices of Q_1 (resp. Q_2) such that $L(x_1) \leq y_1 \leq x_1$ and $L(x_2) \leq y_2 \leq x_2$, at a step of the computation we are interested in computing the distance $D(Q_1[L(x_1) \ldots y_1], Q_2[L(x_2) \ldots y_2])$ between two quotiented ordered forests $Q_1[L(x_1) \ldots y_1]$ and $Q_2[L(x_2) \ldots y_2])$ of Q_1 and Q_2 respectively.

Proposition 6 *The distance δ between the quotiented ordered forests $Q_1[L(x_1) \ldots y_1]$ and $Q_2[L(x_2) \ldots y_2]$ is recursively given by:*

$$\delta = \min \begin{cases} \gamma(\lambda, y_2) + D(Q_1[L(x_1) \ldots y_1], Q_2[L(x_2) \ldots y_2 - 1]) \\ \gamma(y_1, \lambda) + D(Q_1[L(x_1) \ldots y_1 - 1], Q_2[L(x_2) \ldots y_2]) \\ \gamma(y_1, y_2) + D_C(Q_1[L(y_1) \ldots y_1 - 1], Q_2[L(y_2) \ldots y_2 - 1], \pi(y_1), \pi(y_2)) \\ \quad + D_C(Q_1[m(L(x_1), y_1) \ldots L(y_1) - 1], Q_2[m(L(x_2), y_2) \ldots L(y_2) - 1], \pi(y_1), \pi(y_2)) \\ \quad + D(Q_1[L(x_1) \ldots m(L(x_1), y_1) - 1], Q_2[L(x_2) \ldots m(L(x_2), y_2) - 1]) \end{cases}$$

Proposition 6 leads to compute the constrained edit distance between two quotiented ordered forests $Q_1[L(x_1) \ldots y_1]$ and $Q_2[L(x_2) \ldots y_2]$, such that $L(x_1) \leq y_1 \leq x_1$ and $L(x_2) \leq y_2 \leq x_2$ and the quotient vertices C_1 and C_2 are mapped on each other. However, this computation is only needed only if C_1 is an ancestor of $\pi(L(x_1) \wedge y_1)$ and C_2 is an ancestor of $\pi(L(x_2) \wedge y_2)$. The recurrence formulae are detailed in [Ouangraoua 2009b].

The computation of the constrained edit distance is based on a similar scheme to the computation of the distance between ordered trees [Zhang 1989]. The time complexity of the computation is then bounded by $O(|T_1| \times \min\{l(T_1), p(T_1)\} \times |T_2| \times \min\{l(T_2), p(T_2)\}$ where $p(T_i)$ and $l(T_i)$ are respectively the depth and the number of leaves of the tree T_i , $i \in \{1, 2\}$. We can remark that the solution proposed is independent of the size of the quotient graphs $\pi(T_1)$ and $\pi(T_2)$. The size complexity of the algorithm is also bounded by $O(|T_1| \times \min\{l(T_1), p(T_1)\} \times |T_2| \times \min\{l(T_2), p(T_2)\}$ since we need to memorise all the intermediate computed distances in the dynamic programming table D.

4.4 Evaluation of Comparison Methods

We have proposed in the context of Brasero ANR project a first evaluation of all available programs for RNA secondary structure pairwise comparison [Allali 2008]. Our benchmark is based on the *real-life* problem of being able to distinguish sequences of non coding RNAs of a given family from other sequences. Typically, this problem occurs when one aims to find, in a genome or a set of sequences, non-coding RNAs that are similar to some given RNAs from a same family. Another application is the automatic classification of a whole set of putative non-coding RNAs into different functional families. We studied the ability of each program to distinguish alignments within a same family from alignments between two different families. Of course, the quality of a comparison depends on the ability of the folding tool to find the correct secondary structure. Thus the comparison programs were tested in presence of *real-life* noise. We compared six tools: RNAforester [Höchsmann 2003], MiGaL [Allali 2005a], TreeMatching [Ouangraoua 2007], gardenia [Blin 2006], NestedAlign [Blin 2007], and RNAStrAT [Guignon 2005]. These tools use different models of secondary structure. We also included BLAST into the comparison.

RNAforester is an ordered trees local/global alignment algorithm. It uses a special tree encoding that allows to break nucleotide pairings under certain conditions. MiGaL uses a multi-level representation of the secondary structure composed by four layers coded by rooted ordered trees. The layers model different structural levels from multiloop network to the sequence of nucleotides composing the RNA. The algorithm is an adapted edit distance successively applied to each layer.

TreeMatching is based on a quotiented tree representation of the secondary structure which is an auto-similar structure composed of two rooted ordered trees on two different scales (nucleotides and structural elements). The core of the method relies on the comparison of both scales simultaneously: it computes an edit distance between quotiented trees at the macroscopic scale using edit costs defined as edit distances between sub-trees at the microscopic scale. gardenia and NestdAlign use an arc-annotated based representation, that allows for complex edit operations, such as arc-breaking or arc-altering. They allow local and global alignment features. gardenia notably allows affine gap scores while NestedAlign implements an original local alignment algorithm.

RNAStrAT performs the comparison in two steps. First, it compares stems of the two structures using an alignment algorithm with complex edit operations. Then it finds an optimal mapping between the different stems.

We applied the following benchmarking protocol on several families of structural RNAs (tRNA, SRP, ...). For each family, we used about five references. These references were secondary structures obtained in reputed databases. We then took on average 75 sequences of RNAs known to belong to that family (true events). We added 200 sequences randomly picked inside viral genomes (false events). All sequences (true and false) were folded using MFold (keeping optimal and suboptimal) and RNAshapes. This defines the data set. For each tool, we compared each

structure of a same sequence of the data set with all structures of the reference set. We stored the best score obtained. Finally, we sorted the sequences of the data set according to the score obtained. This gave us an array of true and false events. Using this array, we drew the ROC curve, that is the sensibility ($\frac{nb\ of\ true\ positives}{nb\ true\ events}$) and specificity ($\frac{nb\ of\ false\ negatives}{nb\ of\ false\ events}$) curve. We also represented the distribution of the scores and the computation times.

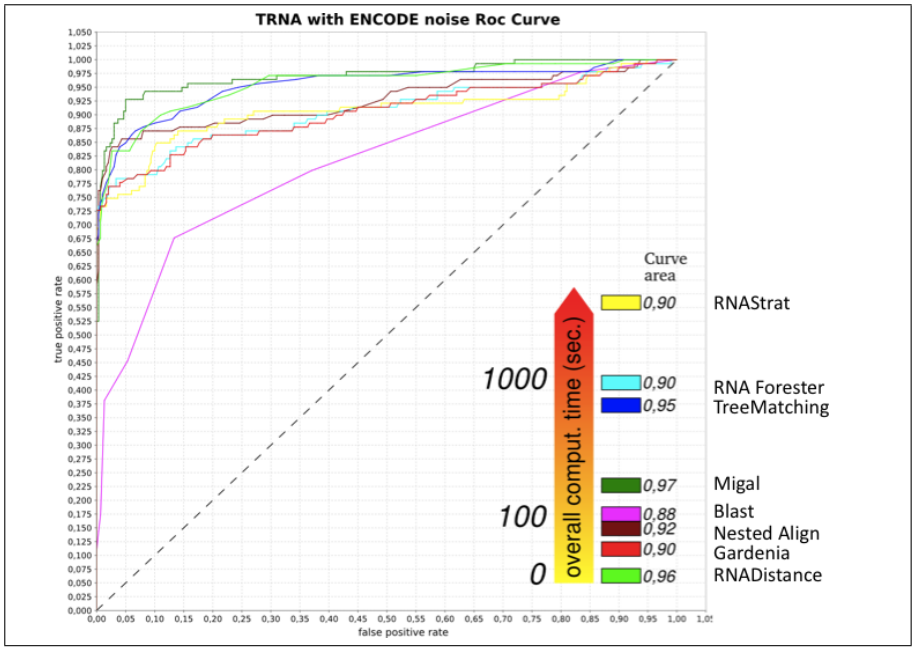

Figure 4.4: Results obtained on tRNA with ENCODE for the different methods studied.

With this work, we propose to help a user to decide which tool is the most adapted to his/her situation (time performance, sequence length, ...). This benchmark is the first one of a set of benchmarks on tools for comparing pairwise RNA secondary structure called BRASERO.

4.5 Presentation of the papers

4.5.1 Tree-Graph Comparison

In the papers presented in this section, we proposed dynamic programming algorithms to evaluate local similarity between ordered quotiented trees using different constrained edit scoring scheme. We have described extensions to globally and locally compare two quotiented trees. This last method allows to find the region in each tree with the highest similarity. Algorithms are been used in genomic analysis to evaluate variability between RNA secondary structures.

[Ouangraoua 2007] Aïda Ouangraoua, Pascal Ferraro, Serge Dulucq and Laurent Tichit. *Local similarity between quotiented ordered trees.* Journal of Discrete Algorithms, vol. 5, no. 1, pages 23–35, 2007.

[Ouangraoua 2009b] Aïda Ouangraoua and Pascal Ferraro. *A new constrained edit distance between quotiented ordered trees.* Journal of Discrete Algorithms, vol. 7, no. 1, pages 78–89, 2009. doi:10.1016/j.jda.2008.05.001.

4.5.2 Applications

In the last ten years, several tools have been proposed for RNA secondary structure pairwise comparison. These tools use different models (ordered tree or forest, arc annotated sequence, multi-level tree) and methods (edit distance, alignment). We have presented in [Allali 2008] a first benchmark for comparing these tools.

[Allali 2008] Julien Allali, Yves d'Aubenton Carafa, Cédric Chauve, Alain Denise, Cédric Drevet, **P. Ferraro**, Daniel Gautheret, Claire Herrbach, Fabrice Leclerc, Antoine de Monte, Aïda Ouangraoua, Marie-France Sagot, Cédric Saule, Michel Termier, Claude Thermes, Hélène Touzet, *Benchmarking RNA secondary structure comparison algorithms*, in: Actes des Journés Ouvertes Biologie, Informatique et Mathématiques - JOBIM'08, 2008, pp. 67–68.

Symbolic Music Analysis 5

Although music is apparently not directly related to tree structured data, I have spent the last few years developing new methods to analyse symbolic music.

The number of audio documents available on the Internet is highly increasing. New methods for browsing, retrieving or classifying have to be proposed to users. The growing Music Information Retrieval research community identifies and explicates

the problems induced by these new methods. One of the key problem of this research area is the estimation of the musical similarity between audio data.

Indeed, one of the main goal of music retrieval systems is to find musical pieces in large databases given a description or an example. These systems compute a numeric score on how well a query match each piece of the database and rank the music pieces according to this score. Computing such a degree of resemblance between two pieces of music is a difficult problem. Three methodologies have been proposed [Orio 2006]. Approaches based on index terms generally consider N-grams techniques [Doraisamy 2003, Uitdenbogerd 2002], which count the number of common distinct terms between the query and a potential answer. Geometric algorithms [Ukkonen 2003, Typke 2004, Typke 2008] consider geometric representations of music and compute distances between objects. Techniques based on string matching are generally more accurate than the previous two, as they can take into account errors in the query or in the pieces of music of the database. This property is of major importance in the context of music retrieval systems since audio analysis always induces approximations. Moreover, some music retrieval application require specific robustness. Query-by-Humming (QbH), a music retrieval system where the input query is a user-hummed melody, is a very good example. Since the sung query can be transposed, played faster or slower, without degrading the melody, retrieval systems have to be both transposition and tempo invariant. Edit distance algorithms, mainly developed in the context of DNA sequence recognition, have been adapted in the context of music similarity. These algorithms, based on the dynamic programming principle, are generalisations of a local sequence alignment method proposed by Smith and Waterman [Smith 1981] in the early 80's. Applications relying on local alignment are numerous and include cover detection [Serrá 2008], melody retrieval [Mongeau 1990], Query-by-Humming [Dannenberg 2007], Query-by-Tapping [Hanna 2009], structural analysis [Hanna 2008], comparison of chord progressions [Bello 2007], *etc.* Local alignment approaches usually provide very accurate results as shown at the recent editions of the Music Information Retrieval Evaluation eXchange (MIREX) [Downie 2008].

As we have seen in previous chapters, measuring similarity between sequences is a well-known problem in computer science which has applications in many fields [Gusfield 1997] such as computational biology, or in text processing, optical character recognition, image and signal processing, error correction, pattern recognition, etc. However, musical sequences are characterized by specific properties. Developing efficient and accurate algorithms implies to take into account areas such as sound perception and music theory. This emerging research area thus requires interdisciplinary collaborations between experts in the field of audio signal processing, algorithmic, pattern recognition, computer music, music theory, *etc.*

In this chapter, we will see a representation of music pieces as sequences and quotiented sequences (*ie.* trees) of characters and then as trees. I will introduce some specific algorithms to compare these sequences and trees in the context of music analysis or to identify their structures.

5.1 Music Representations

In the following, we only consider symbolically encoded pieces of music. Symbolic pieces of music are defined by musical events, such as beginnings or endings of notes. Each note is then defined by a few attributes: pitch, duration or onset time. Such representations allow the application of algorithms adapted from the string-matching domain: N-grams based algorithms [Uitdenbogerd 2002, Doraisamy 2003] or alignment between pieces [Smith 1981], for instance. Since it easily allows the consideration of *elements of music theory* such as tonality, passing notes, or strong and weak beats [Hanna 2007] this kind of methods has been experimented as one of the most accurate in monophonic context [Hanna 2007]. *Monophonic music* is assumed to be composed of only one dominant melody. In a stricter sense, it implies that no more than one note is sounded at any given time. In music theory, *polyphony* is a texture consisting of two or more independent melodic voices. A music with one dominant melodic voice accompanied by chords is often called *homophony*. However, in the following, we will consider homophony as a part of polyphonic music [Uitdenbogerd 1998]. Thus, in the polyphonic context, more than one note can sound at a given time. We will see in this chapter that string-matching methods can be generalized to the polyphony and lead to accurate results.

5.1.1 Representation of Monophonic Musical Pieces as Sequences

Monophonic musical pieces can be represented by trees of pitches [Rizo 2002]. This representation implies a hierarchy relying on bars induced by the time signature of the score notation. However different trees can represent the same melody (same sequence of pitches and durations). For example, two melodies with two different time signatures are represented by two different musical scores. In this case, these two melodies sound similar but are represented in a different way.

Following Mongeau and Sankoff's model [Mongeau 1990], any monophonic score can be represented as a sequence of ordered pairs with the pitch of the note as the first component and its length as the second. Thus, the sequence

```
(B4 B4 r4 C4 G4 E2 A2 G8)
```

represents the example illustrated in Fig. 5.1.

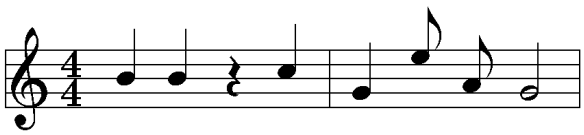

Figure 5.1: Example of monophonic melody.

Several alphabets of characters and set of number have been proposed to represent pitches and durations [Uitdenbogerd 2002, Lemström 2000]. We present only a few ones that we think are the most pertinent in this context.

The melodic contour indicates the variation between successive notes. Only three values are possible: Up, Down, Same. Therefore, the sequence corresponding to Fig. 5.1 is:

$$SUDUDD.$$

The absolute pitch simply indicates the exact pitch (MIDI notation). For example, the melody of Fig. 5.1 is represented by:

$$71, 71, 72, 67, 76, 69, 67.$$

In order to reduce the vocabulary, this exact pitch can be represented by their modulo-12 values. The melodic contour can also be taken into account by using positive values when the melody moves up and negative values when it moves down. The *directed modulo-12 absolute pitch* sequence corresponding to the melody represented by Fig. 5.1 is:

$$11, 11, +0, -7, +4, -9, -7.$$

In the context of query by humming applications, this representation present the huge disadvantage to be not transposition invariant.

At the contrary of the *absolute* pitch representations, the *interval* and *key relative* representations are transposition invariant. The *exact interval* representation is simply the number of semitones between two successive notes. The *exact interval* sequence corresponding to Fig. 5.1 is:

$$0, 1, 5, 9, 7, 2.$$

This representation can also be limited with modulo-12. Information about melodic direction can also be indicated:

$$0, +1, -5, +9, -7, -2.$$

The *key relative* representations indicate the difference in semitones between notes and the key of the melody. In the case of Fig. 5.1, the key signature corresponds to `C` major. Therefore the associated sequence is:

$$11, 11, 0, 7, 4, 9, 7.$$

This representation can also be limited according to modulo-12 and the information about melodic contour can be indicated:

$$11, 11, +0, -7, +4, -9, -7.$$

The limitations of the *key relative* representation is closely linked to the choice of the key. The correct key has to be known in order to compute the correct representation.

Concerning the note durations, the same representations are possible. The duration contour (Shorter, same, Longer) indicates the general variation of duration between successive notes. Therefore the duration representation of the melody of Fig. 5.1 is:

$$ssssSsL.$$

The *absolute representation* simply indicates the length of the note in sixteenth notes:

$$4, 4, 4, 4, 4, 2, 2, 8.$$

It is important to note that this representation is not tempo invariant, while the *relative* representation is tempo invariant. The difference of durations between successive notes can be expressed as duration subtraction:

$$0, 0, 0, 0, 2, 0, 6$$

or duration ratio:

$$1, 1, 1, 1, \frac{1}{2}, 1, 4.$$

5.1.2 Representation of Polyphonic Musical Pieces

In order to compare polyphonic music, existing works generally require reduction of a polyphonic piece [Uitdenbogerd 2002]. Certain methods [Madsen 2008] reduce polyphonic music as a set of separate tracks. We have chosen to not use any information about the voice lines, since they could be missing in the case of polyphonic music obtained by transcription from audio, for example. Other monophonic reductions only consider the note with the highest pitch in the polyphony, based on two assumptions: firstly the query is the main theme or the melody of the musical piece searched, and secondly the highest pitches always form the melody. It is rarely the case in an orchestra, where the polyphony could not be reduced to just the voice of the Western concert flute for example. In order to avoid such assumption, we propose here to study algorithms that take into account all the notes of a polyphonic musical piece. One way would be to consider all the distinct monophonic lines induced by the polyphony. But this naive approach may imply a combinatoric explosion in the cases of large scores [Lemström 2006].

To take into account the polyphonic nature of musical sequences, we thus propose to use a quotiented sequence representation. Formally, a quotiented sequence is a sequence graph with an equivalence relation defined on the set of vertices, and such that the resulting quotient graph is also a sequence.

Definition 9 *A* quotiented sequence *is a 3-tuple* $Q = (S, W, \pi)$ *where* S *is a sequence called the support of* Q*,* W *is a set of vertices and* π *a surjective application from* V*, the set of vertices of* S *to* W*.*

The quotient graph $\pi(S)$ associated with Q is (W, E_π) such that: $\forall (x, y) \in E, (\pi(x), \pi(y)) \in E_\pi \Leftrightarrow \pi(x) \neq \pi(y)$. By definition, in a quotiented sequence, quotient graph and support sequence are both sequences. A quotiented sequence can thus be considered as a self-similar structure represented by sequences on two different scales.

A quotiented sequence can also be seen as a tree structure in which the leaves represent the notes of the musical piece and fathers of leaves represent chords. In

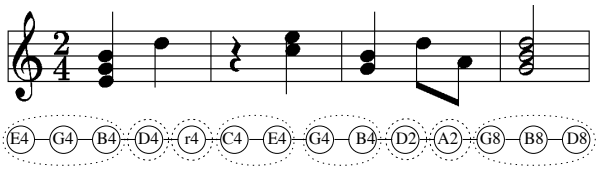

Figure 5.2: An example of a polyphonic musical score and its related quotiented sequence.

the context of polyphonic music, notes that occur at the same time are grouped to form a quotiented sequence $Q = (S, W, \pi)$ (Figure 5.2) where S is the suite of notes, W the suite of chords and π the application that maps a set of notes to each chord. Each vertex of the quotiented sequence is labelled by the pitch and the duration of each note.

5.1.3 Harmony Tree

Many works have already proposed to structure the representation of harmony. Thus, methods based on the theory from Schenker [Schenker 1935] or Lerdahl and Jackendoff [Lerdahl 1985] induce a tree, with rule-based reduction algorithms. We proposed [Robine 2009] to represent the harmony of a musical piece with an ordered tree, according to the time, on five levels illustrated in Fig. 5.3. In our case this order is given by the time. Let us define the five levels of the harmony tree:

1. The main tonality of a musical piece is the root of the structure.
2. The second level is the sequence of local tonalities, *i.e.* the successive modulations, each of them being linked to the root.
3. The third level is the chord sequence, as the harmonic background of a jazz standard could be notated, for example. Each chord of this level contributes to a tonality and is therefore linked to a local tonality above.
4. The fourth level contains the sequence of noteChords, that is a vertical reduction of a polyphony. Several noteChords could sound during one chord, they are therefore all linked to a father chord.
5. The fifth and last level contains the sequence of the notes of a musical piece. Each note is a part of a note-Chord.

The comparison of harmonic trees has been presented in [Robine 2009]. The consideration of a tree structure, instead of a list of sequences to represent musical pieces, requires techniques for tree comparison. The technique used to compare these trees is similar to the ones presented in the previous chapter and more precisely based on Selkow's algorithm [Selkow 1977]. This algorithm won't be presented here.

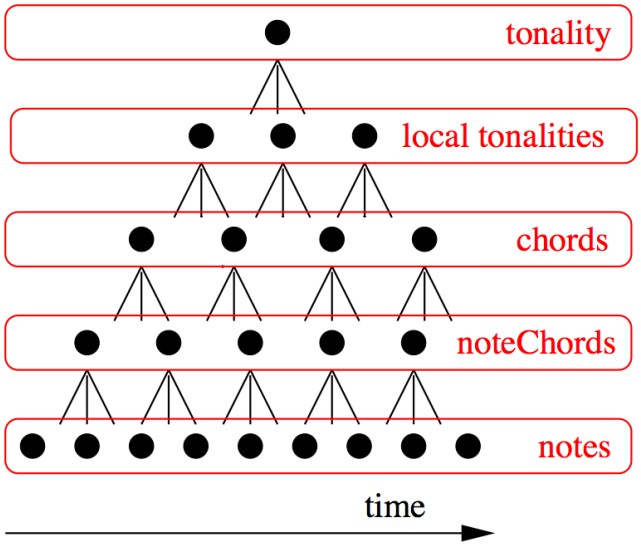

Figure 5.3: Ordered tree for harmony representation.

5.2 Aligning two Pieces of Music

5.2.1 General Sequence Alignment

In the early seventies, [Needleman 1970] and then [Wagner 1974] proposed algorithms which compute a similarity measure between two strings of symbols as the maximum score sequence of elementary operations needed to transform one of the strings into the other. Given two strings of symbols S_1 and S_2 of respective lengths $|S_1|$ and $|S_2|$, a set of elementary operators on strings, called edit operations, and a score associated to each edit operation, a score between these two strings is defined as the score of the sequence of edit operations that transforms S_1 into S_2 with maximum score. This similarity measure makes use of the dynamic programming principle to achieve an algorithm with quadratic complexity, *i.e.* in $O(|S_1| \times |S_2|)$.

Let us consider only the three edit operations that are usually used to compare musical sequences: substitution, deletion and insertion. Let denote x_i the i^{th} character of S_1 and y_j the j^{th} character of S_2. Let e be an edit operation, a score s is assigned to each edit operation as follows:

- if e substitutes x_i (the i^{th} character of S_1) into y_j (the j^{th} character of S_2) then $s(e) = s(x_i, y_j)$
- if e deletes x_i then $s(e) = s(x_i, \lambda)$
- if e inserts y_j then $s(e) = s(\lambda, y_j)$.

The score s is extended to a sequence of edit operation $E = (e_1, e_2, \ldots, e_n)$ by letting $s(E) = \sum_{k=1}^{n} s(e_k)$. This makes it possible to define a score $\sigma(S_1, S_2)$ between

sequences S_1 and S_2 as the maximum score of edit operation sequences transforming S_1 into S_2, namely:

$$\sigma(S_1, S_2) = \max_{E \in \mathcal{E}} \{s(E)\}$$

where $\mathcal{E}$ represents the set of sequences of edit operations transforming S_1 into S_2.

To simplify the problem of finding the edit distance between two strings, Wagner and Fisher [Wagner 1974] and Needleman and Wunsh [Needleman 1970] introduce the notion of alignment (or trace). *Sequence alignment* refers to a method that allows the computation of a one-to-one mapping between symbols of two sequences that respects symbol order. Scores are associated with each symbol pair and each symbol not in a pair. Sequence alignment aims at finding a mapping between symbols that maximizes the sum of scores. Here is a formal definition of the problem:

Definition 10 (Sequence Alignment) *Let S_1 and S_2 be two sequences over an alphabet Σ, whose respective lengths are $|S_1|$ and $|S_2|$. A valid alignment A from S_1 to S_2 is a set of ordered pairs of integers (i, j) satisfying:*

1. *$1 \leq i \leq |S_1|$ and $1 \leq j \leq |S_2|$,*
2. *for any distinct pairs (i_1, j_1) and (i_2, j_2) in A:*
   (a) *$i_1 \neq i_2$ if and only if $j_1 \neq j_2$,*
   (b) *$i_1 < i_2$ if and only if $j_1 < j_2$.*

Condition 1 ensures that only character positions of the respective strings are involved in the alignment. Condition 2(a) ensures that each character position of either string is aligned with at most one character position of the other string; condition 2(b) ensures that the order between character positions is maintained in the alignment. Let I and J be the sets of positions in S_1 and S_2 respectively not involved in any pair in A.

Let s be an arbitrary scoring function which assigns to each pair (a, b) of $\Sigma \cup \{\varepsilon\} \times \Sigma \cup \{\varepsilon\}$ a real number $s(a, b)$. Let A be an alignment from S_1 to S_2, the score $S(A)$ associated with the alignment A is defined by:

$$S(A) = \sum_{(i,j) \in A} s(x_i, y_j) + \sum_{i \in I} s(x_i, \varepsilon) + \sum_{j \in J} score(\varepsilon, y_j).$$

Finally, the alignment score $\sigma(S_1, S_2)$ between S_1 and S_2 is defined as:

$$\sigma(S_1, S_2) = Max(S(A) \ for \ all \ valid \ A \ between \ S_1 \ and \ S_2).$$

It can be shown that this alignment score is equal to the edit score defined previously [Wagner 1974].

A pair of two identical symbols in the alignment A is called a *match*. A pair of two different symbols is called a *mismatch*. Symbols not involved in the alignment are

called *gaps*. Thus the score of A is the cost of the match and mismatch pairs, added to the cost of gaps.

The alignment A from S_1 to S_2 is usually represented by two sequences S'_1 and S'_2 of the same length over the alphabet $\Sigma \cup \{-\}$. S_1 and S_2 are respectively obtained by suppressing the symbols $-$ from respectively S'_1 and S'_2. Let us consider the following alignment from the word $MUSIC$ to the word $QUICK$:

position:	0	1	2	3	4	5
	M	U	S	I	C	-
	Q	U	-	I	C	K
score:	-1	2	-1	2	2	-1

In this example, we have a mismatch in position 0, a match in positions 1, 3 and 4 and two gaps in positions 2 and 5. If the score for a match is $+2$ and -1 for a mismatch or a gap then the alignment score is 3.

The alignment score $\sigma(S_1, S_2)$ from S_1 to S_2 may be computed rapidly (in time proportional to $|S_1| \times |S_2|$ by recurrence). The recurrence equation from $1 \leq i \leq |S_1|$ and $1 \leq j \leq |S_2|$ is (see Fig. 5.5-left):

$$\sigma(S_1[x_i], S_2[y_j]) = Max \begin{cases} \sigma(S_1[x_{i-1}], S_2[y_j]) & + & s(x_i, \varepsilon) \\ \sigma(S_1[x_i], S_2[y_{j-1}]) & + & s(\varepsilon, y_j) \\ \sigma(S_1[x_{i-1}], S_2[y_{j-1}]) & + & s(x_i, y_j) \end{cases} \tag{5.1}$$

where $S_1[x_i]$ (resp. $S_2[y_j]$) represent the substring $x_0 x_1 \dots x_i$ of S_1 (resp. $y_0 y_1 \dots y_i$ of S_2). In the following, we will also use the notation $S_1[x_s, x_e]$ to designate the substring $x_s \dots x_e$ of S_1.

The computation can be done by computing a table M of size $(|S_1|+1) \times (|S_2|+1)$ where the value $M[i][j]$ represents $\sigma(S_1[x_i], S_2[y_j])$. The first row (resp. column) contains scores of the alignment between $S_1[x_i]$ (resp. $S_2[y_j]$) and ε. The time complexity of this algorithm is $O(|S_1| \times |S_2|)$ in time and $O(min(|S_1|, |S_2|))$ in memory as we can make the computation using only one row or column [Smith 1981].

5.2.2 Local Alignment

In many applications, two strings may not be highly similar in their entirety but may contain *regions that are highly similar*. This is particularly true when long stretches of anonymous sequences are compared, since only some internal sections of those strings may be related. In this case, the task is to find and extract a pair of regions, one from each of the two given strings, that exhibits high similarity. This is called *local alignment* or *local similarity problem* [Smith 1981] and is defined as : given two strings S_1 and S_2, find sub-strings ρ_1 and ρ_2 of S_1 and S_2, respectively, whose similarity is maximum over all pairs of sub-strings from S_1 and S_2.

The computation of a local similarity allows us to detect local conserved areas between both sequences. The solution of such a problem is based on the notion of

suffix mapping between sequences. The local suffix mapping problem for a given pair x_i, y_j of symbols is to find a (possibly empty) suffix ρ_1 of the sub-sequence $S_1[x_i]$ (defined from the first symbol of string S_1 to x_i) and a (possibly empty) suffix ρ_2 of the sub-sequence $S_2[y_j]$ of S_2 such that the score of the optimal sequence of edit operations transforming ρ_1 into ρ_2 is the maximum over all scores of sequences of edit operations between suffixes of $S_1[x_i]$ and $S_2[y_j]$.

The score of the sequence solving the optimal local suffix mapping problem (called local score) for a given pair x_i, y_j of symbols is denoted by $\mathrm{LS}(x_i, y_j)$:

$$\mathrm{LS}(x_i, y_j) = \max\{\sigma(\rho_1, \rho_2), (\rho_1, \rho_2) \text{ suffixes of } S_1 \text{ and } S_2\}.$$

Local similarity between two sequences is then defined as the score of the best pair of local suffixes in trees S_1 and S_2:

$$\mathrm{LS}(S_1, S_2) = \max\{\mathrm{LS}(x_i, y_j), (x_i, y_j) \in S_1 \times S_2\}.$$

So, in order to evaluate local similarity, the algorithm needs to find maximum similarity between suffixes of $S_1[x_i]$ and $S_2[y_j]$, for any pair (x_i, y_j) of $S_1 \times S_2$, and then to determine the best pair $x_1^{\max}$, $y_2^{\max}$ of S_1 and S_2.

Since we can always choose an empty suffix, $\mathrm{LS}(x_i, \theta) = 0$ and $\mathrm{LS}(\theta, y_j) = 0$, where θ is an empty sequence. And finally, for any (x_i, y_j), the proper recurrence for $\mathrm{LS}(x_i, y_j)$ is:

$$\mathrm{LS}(x_i, y_j) = \max \begin{cases} 0 \\ \mathrm{LS}(x_{i-1}, y_j) + s(x_i, \lambda) \\ \mathrm{LS}(x_i, y_{j-1}) + s(\lambda, y_j) \\ \mathrm{LS}(x_{i-1}, y_{j-1}) + s(x_i, y_j) \end{cases}$$

where x_{i-1} and y_{j-1} respectively represent symbols before x_i and y_j in sequences S_1 and S_2. Note that if the query sequence S_1 has only one symbol x, then the local score between S_1 and S_2 is obtained from an empty sequence (*ie.* there is no matching) or from a unique matching between x and the most similar symbol of S_2.

Dynamic programming algorithms can compute the optimal alignment (either global or local) and the corresponding score in time $\mathcal{O}(|S_1| \times |S_2|)$ and memory $\mathcal{O}(\min\{|S_1|, |S_2|\})$ (see [Needleman 1970, Wagner 1974, Smith 1981] for details).

These two former algorithms we designed mainly for bioinformatic applications. However in music context they would need some adaptations.

5.2.3 Local Transposition

Musical queries produced by human beings can, not only be totally transposed, but can also be composed of several parts that are independently transposed. For example, if the original musical piece is composed of dierent harmonic voices, the user may sing dierent successive parts with dierent keys. In the same way, pieces of popular music are sometimes composed of dierent choruses sung based on dierent

tonic. A sung query may imitate these characteristics. Moreover, errors in singing or humming may occur, especially for users that are not trained to perfectly control their voice like professional singers. From a musical point of view, sudden tonal changes are disturbing. However, if these changes last during a long period, they may not disturb listeners. Figure 5.4 shows an example of query having two local transpositions. The two pieces in Figure 5.4 sound very similar, although the two re-

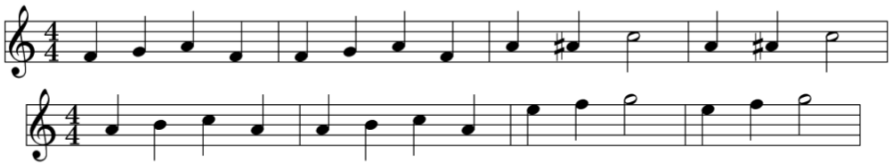

Figure 5.4: Example of a monophonic query not transposed (top) and a monophonic query characterized by two local transpositions (bottom).

sulting sequences are very dierent. We have addressed this problem in [Allali 2007]. The solution (not developed here) requires to compute multiple score matrices simultaneously, one for each possible transposition value. The time complexity is $\mathcal{O}(\Delta \times |S_1| \times |S_2|)$, where Δ is the number of local transposition allowed during the comparison (for practical applications, Δ is set up to 12). Our experiments show that the local transposition algorithm provides a much better results, in particular for Query-by-Humming (QbH) applications.

5.2.4 An Extended Set of Edit Operations

When comparing monophonic pieces of music, one limitation of the alignment approach is that it only allows one-to-one association. Actually, in musical pieces, a single note in one sequence may sometimes be split into two or more notes in the second sequence. To avoid this limitation, Mongeau and Sankoff [Mongeau 1990] introduced a new operation, called *merge*, allowing the replacement of several characters by a single character and the replacement of one character by several. The motivation for this new operation, is the fact that a whole note can be replaced by four quarter notes of the same pitch, which is not very costly (in term of perception).

The definition of an alignment is then modified as follows:

Definition 11 *Let S_1 and S_2 be two sequences over an alphabet Σ, whose respective lengths are $|S_1|$ and $|S_2|$. A valid alignment A from S_1 to S_2 is a set of pairs of ordered pairs of integers $((i_s, i_e), (j_s, j_e))$ satisfying:*

1. $1 \leq i_s, i_e \leq |S_1|$ *and* $1 \leq j_s, j_e \leq |S_2|$*; and* $i_e > i_s$ *and* $j_e > j_s$*,*
2. $i_e \neq i_s + 1 \Rightarrow j_e = j_s + 1$ *and* $j_e \neq j_s + 1 \Rightarrow i_e = i_s + 1$*,*
3. *if* $((i'_s, i'_e), (j'_s, j'_e))$ *and* $((i'_s, i'_e), (j'_s, j'_e))$ *are any two distinct elements of* A*:*
   (a) i'_s *cannot be equal to* i_s*,*

(b) if $i'_s < i_s$ *then* $i'_e \leq i_s$ *and* $j'_e \leq j_s$,
(c) if $i'_s > i_s$ *then* $i'_s \geq i_e$ *and* $j'_s \geq j_e$.

Condition 1 ensures that only character positions of the respective strings are involved by the alignment; Condition 2 ensures that a sequence of two non consecutive character positions in a string is associated with at most one character position; Conditions 3 ensure that the order between character position is maintained in the alignment.

To take into account the merge operation in the recurrence 5.1, the following cases must be added to the recurrence formula (see Fig. 5.5-right):

$$\begin{array}{lll} 2 & \leq k \leq i: & \sigma(S_1[x_{i-k}], S_2[y_{j-1}] + s(S_1[x_{i-k}], y_j), \\ 2 & \leq l \leq j: & \sigma(S_1[x_{i-1}], S_2[y_{j-l}]) + s(x_i, S_2[y_{j-l}, y_j]). \end{array} \tag{5.2}$$

where $score(S_1[x_{i-k}], y_j)$ and $score(x_i, S_2[y_{j-l}])$ are the predefined weights associated to the merge operations.

The time complexity of the algorithm with the merge operation is $O((|S_1| \times |S_2|)(|S_1| + |S_2|))$. In practice, the number of consecutive merged notes is bounded by a constant L which leads to a complexity of $O(|S_1| \times |S_2| \times L)$ [Mongeau 1990].

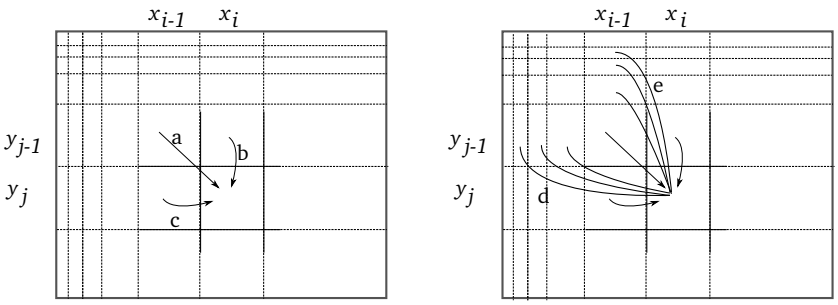

Figure 5.5: On the left, the dynamic programming table for the alignment of S_1 and S_2. The arrows a, b and c represent the three cases of recurrence 5.1 for the computation of $M[i+1][j+1]$. On the right, the computations added by Mongeau and Sankoff algorithm (Formula 5.2). Arrows d correspond to merge in S_1, arrows e correspond to merge in S_2.

So far, to define an accurate algorithm, a scoring function must be defined. A naive scoring scheme like constant operation scores or a complex scoring scheme can produce significantly different results [Hanna 2007]. Setting these parameters is tricky and generally leads to a scoring table that contains the scores between each possible pitch difference. Indeed, substituting one pitch with another one has more or less influence on the general melody. As introduced in [Mongeau 1990], scores are thus determined according to consonance. In particular, interval scores decrease for Western tonal music in order of decreasing dissonance: unison, fifth, third, sixth, fourth, seventh and second.

In the monophonic case, a good scoring scheme between two notes x_i and y_j is often reached by using a function $score_p$ on pitches (its values are coded into a table) and a function $score_d$ on durations, such as:

$$score(x_i, y_j) = \alpha \times score_p(pitch(x_i), pitch(y_j)) \\ +\beta \times score_d(duration(x_i), duration(y_j)).$$

For a merge operation the weight $score(S_1[x_{i-k}, x_i], y_j)$ is computed similarly as:

$$score(S_1[x_k, x_i], y_j) = \alpha \times \sum_{l=k}^{i} score_p(pitch(x_l), pitch(y_j)) \\ +\beta \times score_d(\sum_{l=k}^{i} duration(x_l), duration(y_j)).$$

Usually, the cost associated to a gap only depends on the note duration.

5.2.5 Polyphonic Case

A lot of problems arise, when dealing with polyphonic music alignment. Actually, the definition of an alignment in the polyphonic case is not a straightforward application of the monophonic comparison.

Since many notes may be played at the same time, relative encoding cannot be used. Thus, polyphonic music can be represented by a sequence of sets of notes. A set can contain a single note or a chord. A direct consequence is that transpositions cannot be treated by the relative encoding.

Furthermore, as presented before, setting up a scoring scheme in the monophonic case (*i.e.* fixing a score for two notes) is a difficult problem. This task becomes harder when comparing two chords. Indeed, in one octave there are 12 possible pitch values for a chord made of a single note (in practical applications, it is common to work only on one octave), then 12×11 for two note chords $\ldots \binom{12}{p}$ for p note chords, which means the scoring scheme will be represented by a matrix of size $2^{12} \times 2^{12}$. Moreover, complex note rearrangements may occur between two similar

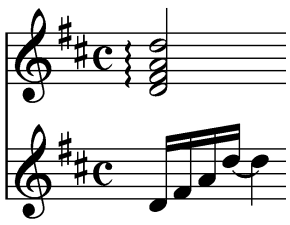

Figure 5.6: Arpeggiated chord above, with its interpretation below. The notes are played successively from the lowest to the highest pitch, and not simultaneously.

polyphonic pieces, and temporal deviations may appear between a score and its

interpretation. Fig. 5.6 shows such an example: in the notation score, the chord has to be arpeggiated, and the related interpretation is transcribed as successive notes. In this case, a comparison system has to detect the similarity between the chord indicated in the musical score and the successive notes interpreted.

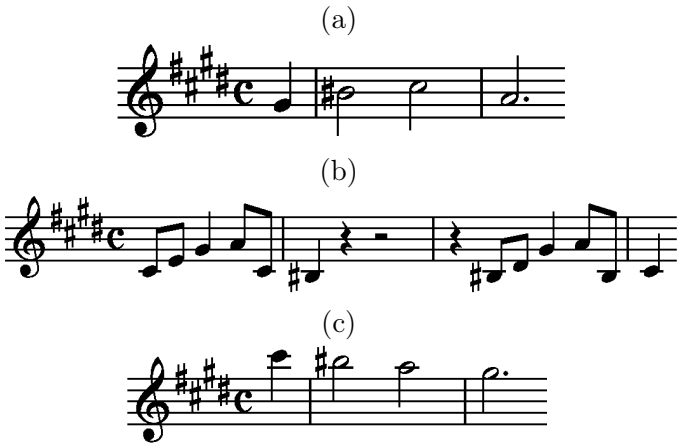

Figure 5.7: Similarity despite permutations (a) Main motif of the 14^{th} string quartet in C# minor opus 131 by Beethoven (1st movement, bar 1). The motif is composed by 4 notes (sequence (1 2 3 4)). (b) First theme of the 7^{th} movement, bar 1. The 4 last notes of the two groups are permuted notes of the main motif, sequence (1 4 3 2) and (1 4 2 3) (c) Second theme of the 7^{th} movement, bar 21. The 4 notes are again a permutation of the main motif, sequence (3 2 4 1).

More generally, we have to deal with notes/chords merging and local rearrangements. For example, composers may choose to change the order of the notes in a melodic motif during a musical piece. Fig. 5.7 shows three excerpts from a piece by Beethoven: the second (b) and third (c) excerpts correspond to a rearrangement of the main motif (a) with swapped notes.

5.3 Polyphonic Alignment

I present hereafter a general method to align two polyphonic musical pieces. The main characteristic of this method is that it is based on the scoring scheme between monophonic music pieces introduced in the previous section.

5.3.1 Chord Comparison

In many cases, an arbitrary order is given to the notes composing the chords of a musical sequence. To avoid this arbitrary choice, one can consider chords as sets. The cost for substituting one chord by another one leads to the problem of computing the best permutation between both chords. Fig. 5.8 shows an example of two cadences that sound similar, but that can be estimated as very dissimilar

because of the different order of the notes in the chords. To avoid this sort of problem, we suggest that chords should be considered as unordered sets and the best permutation should be found. This optimization method allows the estimation of a high similarity between these two sequences of chords.

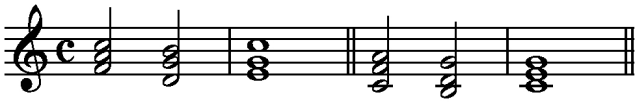

Figure 5.8: Similarity between inverted chords. These successive two perfect cadences in C major are similar despite the different order of the notes in the chords composing the cadences.

This optimization problem is actually a maximum score maximum bipartite matching problem and can be modeled as a weighted maximum matching algorithm [Edmonds 1972]. A similar approach has been proposed in the case of a geometric representation of musical pieces [Typke 2004].

Let C_1 and C_2 be two chords of size n and m. We denote by $score_{bpg}(C_1, C_2)$ the score between these two chords. To compute $score_{bpg}(C_1, C_2)$ as the maximum score maximum bipartite matching problem, we consider the following graph $G(v, w) = (V, E)$ (Fig. 5.9):

1. *vertex set* : $V = \{s, t, e_1, e_2\} \cup \{s_1^1, s_1^2, \dots s_1^n\} \cup \{s_2^1, s_2^2, \dots s_2^m\}$, where s is the source, S_1 is the sink, $\{s_1^1, s_1^2, \dots s_1^n\}$ and $\{s_2^1, s_2^2, \dots s_2^m\}$ are the notes of the chords C_1 and C_2 and e_1, e_2 represent ε;

2. *edge set* : (s, s_1^k), (s, e_1), (e_2, t), (s_2^l, t) with a score 0, (s_1^k, s_2^l) with score $score(s_1^k, s_2^l)$, and (s_1^k, e) with score $score(s_1^k, \varepsilon)$. All the edges have a capacity of 1 except (e, t) which capacity is $n - m$.

G is then a graph whose edges are labeled with integer capacities, non-negative scores in $\mathbb{R}$, and the maximum flow $f^* = n + m$. The score of the maximum flow is actually the score $score_{bpg}(C_1, C_2)$ and the complexity of computing local score is due to this maximum score maximum flow computation. This problem can be solved by the Edmonds and Karp's algorithm [Edmonds 1972] improved by Tarjan [Tarjan 1983] whose complexity is $O(|E||f^*| \log_2(|V|))$. For our graph, the maximum flow is $f^* = n+m$, the edge number is $|E| = n \times m + 2n + 2m + 3$ and the vertex number is $|V| = n + m + 4$. Finally the complexity of the score computation between two chords is bounded by $O(n^3 \times \log_2(n))$ where n represents the maximum number of notes in a chord.

In conclusion, computing alignment between two strings S_1 and S_2 leads to a total time complexity of $O(|S_1| \times |S_2| \times C^3 \times \log_2(C))$ where C is the maximum number of notes in a chord in S_1 or S_2. In practical applications the parameter C is generally bounded by 4.

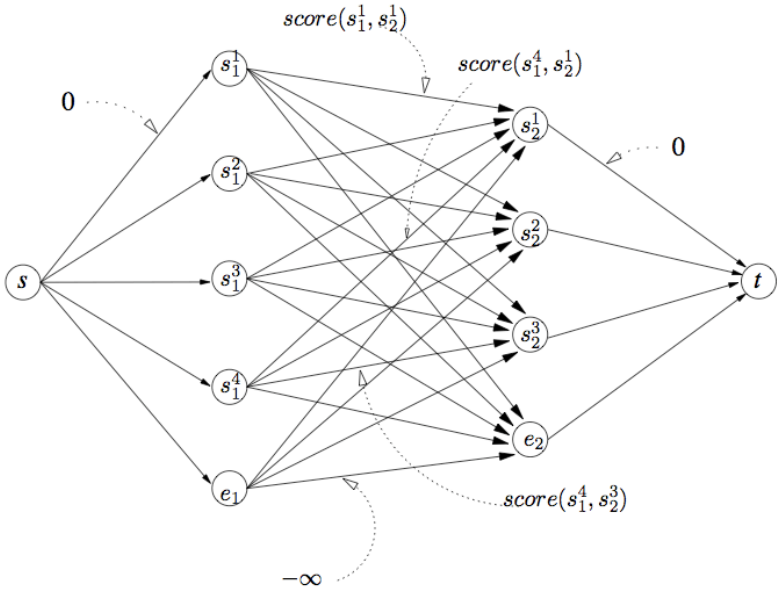

Figure 5.9: Resolution of the optimal permutation as a maximum score flow problem.

5.3.2 Extending Mongeau-Sankoff Operations

As we already stated, an accurate algorithm for music alignment must take into account both local rearrangements and merging operations. We thus propose allowing the merging of sub-sequences in both S_2 and S_1 simultaneously.

Music alignment between two sequences is then extended as follows:

Definition 12 (Extended Music Alignment) *Given two sequences S_1 and S_2 of sets of symbols over Σ. A valid extended alignment X^a between S_1 and S_2 is a set of pairs of sub-sequences of S_1 and S_2, that is $X^a = \{((i_s, i_e), (j_s, j_e))\}$ and respects the following for all $((i_s, i_e), (j_s, j_e)) \in X^a$:*

1. $0 \leq i_s < i_e \leq |S_1|$ *and* $0 \leq j_s < j_e \leq |S_2|$,
2. $\forall((i'_s, i'_e), (j'_s, j'_e)), ((i'_s, i'_e), (j'_s, j'_e)) \in X^a$ *such that* $i'_s \neq i_s$:
   (a) *if* $i'_s < i_s$ *then* $i_e \leq i_s$ *and* $j'_e \leq j_s$,
   (b) *if* $i'_s > i_s$ *then* $i'_s \geq i_e$ *and* $j'_s \geq j_e$.

We define the set G_1 (resp. G_2) as the set of positions of S_1 (resp. S_2) not involved in X^a, that is

$$G_1 = \{1, \ldots, |S_1|\} \backslash \bigcup_{((i_s,i_e),(j_s,j_e)) \in X^a} \{i_s \ldots i_e\}.$$

The score associated with X^a is defined as follow:

$$S(X^a) = \sum_{((i_s,i_e),(j_s,j_e))\in X^a} score(S_1[x_s,x_e],S_2[y_s,y_e]) + \sum_{i\in G_1} s(x_i,\varepsilon) + \sum_{j\in G_2} s(\varepsilon,y_j).$$

The adaptation of the alignment algorithm is straightforward. The recurrence 5.1 is modified by adding the following cases:

$$\begin{aligned} \forall\ 2 \quad & \leq k \leq i, 2 \leq l \leq j : \\ & \sigma(S_1[x_{i-k}], S_2[y_{j-l}]) + score(S_1[x_{i-k}, x_i], S_2[y_{j-l}, y_j]). \end{aligned} \tag{5.3}$$

In the dynamic programming table, the value of a given position $M[i][j]$ is obtained from any case $M[k][l]$ for $1 \leq k \leq i$, $1 \leq l \leq j$. We thus need to consider the computation of the value $score(S_1[x_{i-k}, x_i], S_2[y_{j-l}, y_j])$ (*ie.* the cost of aligning the sequence of chords $S_1[x_{i-k}, x_i]$ and the sequence of chords $S_2[y_{j-l}, y_j]$).

For the comparison of two polyphonic sequences S_1 and S_2, we propose to define a scoring scheme based on the on the scoring scheme presented in Section 5.2.4. However, in this case, we are dealing with sequences of chords instead of single chords and we thus need a way to encode several chords into a single one.

Let us consider the sequence of chords $S_1[x_{i_s}, x_{i_e}]$ that must be encoded (and that will be compared to a sequence of chords $S_2[y_{j_s}, y_{j_e}]$ in S_2). The pitch of $S_1[x_{i_s}, x_{i_e}]$ is then defined as the set T_1^p of all the different pitches that are present in this sequence. The duration of this sequence of chords is equal to T_1^d the duration elapsed from the beginning of x_{i_s} to the end of x_{i_e}.

For example, in Fig. 5.2, the sequence of 4 chords:

$$(\{G4, B4\}, \{D2\}, \{A2\}, \{G8, B8, D8\})$$

is encoded by the chord

$$(\{A, B, D, G\})$$

with a duration of 16.

Finally, a sequence of chords is only represented by a single set of unordered pitches and a single duration. To compare these chord representations the approach described in Section 5.3.1 and $score_p$ to weight the edges of the bipartite graph are used. Then, the score between two sequences of chords is given by:

$$score(S_1[x_{i_s}, x_{i_e}], S_2[y_{j_s}, y_{j_e}]) = \alpha \times score_{bpg}(T_1^p, T_2^p) + \beta \times score_d(T_1^d, T_2^d)$$

Now let us consider the computation of the dynamic programming table M. We propose to illustrate the computation of a case $M[i][j]$ of this table through the example in Tab. 5.1. The score in position X is obtained either from f which implies the computation of scores $score_{bpg}(\{A\}, \{C, G\})$ and $score_d(2, 2)$ or:

- from e which implies the computation of $score_{bpg}(\{A\}, \{A, C, G\})$ and $score_d(2, 3)$.

		{A1}	{C2,G2}	{C1,E1,B1}
	...	...	...	...
{G4,B4 }	...	a	b	
{D2}	...	c	d	
{A2}	...	e	f	
{G8,D8,B8}	...			X

Table 5.1: Example of merged notes that must be considered for one step (X) of the alignment algorithm.

- from d which implies the computation of $score_{bpg}(\{A, D\}, \{C, G\})$ and $score_d(4, 2)$.
- from c which implies the computation of $score_{bpg}(\{A, D\}, \{A, C, G\})$ and $score_d(4, 3)$.
- from b which implies the computation of $score_{bpg}(\{A, B, D, G\}, \{C, G\})$ and $score_d(8, 2)$.
- from a which implies the computation of $score_{bpg}(\{A, B, D, G\}, \{A, C, G\})$ and $score_d(8, 3)$.

One can observe that from one computation of $score_{bpg}$ to another one, we just add vertices in the bipartite graph. So, it is not necessary to recompute $score_{bpg}$ from scratch. Toroslu and Üçoluk give in [Toroslu 2007] an incremental algorithm to compute the assignment problem in $O(V^2)$ where V is the number of vertices in the bipartite graph. Using this algorithm in our approach the time complexity of the computation of all possible merges for the case i, j is bounded by $O(\sum_{i=1}^{C} i^2) = O(C^3)$ where C is number of different pitches in $S_1[x_i i]$ and $S_2[y_j]$. The time complexity of the alignment becomes $O(|t|^2 \times |q|^2 \times C^3)$ where C is the number of different pitches in S_1 and S_2.

5.3.3 An Illustrative Example

In order to illustrate the algorithm, we propose to compare the following sequence of chords: t =({G4}, {B4},{D2,A2},{G8},{D8},{B8}) (*cf.* Fig. 5.2) and an arpeggiated version of this sequence: S_2 =({G4,B4}, {D2},{A2},{G8,D8,B8}). The distance between S_1 and S_2 is computed using the following scoring scheme. Let us consider two notes a and b:

- $score(a, a) = 2$,
- $score(a, b) = 1$ if $a \neq b$,
- $score(a, \varepsilon) = score(\varepsilon, b) = -1$,

	ε	{G4}	{B4}	{D2,A2}	{G8}	{D8}	{B8}
ε	0	-1	-2	-4	-5	-6	-7
{G4,B4 }	-2	1	4	2	1	0	-1
{D2}	-3	0	3	5	4	3	2
{A2}	-4	-1	2	8	7	6	5
{G8,D8,B8}	-7	-4	-1	5	8	11	14

Table 5.2: Example of a dynamic programming table for the comparison between the sequence of chords ({G4},{B4},{D2,A2},{G8},{D8},{B8}) and ({G4,B4},{D2},{A2},{G8,D8,B8})

In this example, we suppose the score between two notes does not depend on the pitch and the duration of the notes. The dynamic programing table computation is represented in Tab. 5.2. We then can observe, for instance, that the score between the subsets of notes ({G4},{B4},{D2,A2}) and ({G4,B4},{D2},{A2}), is obtained from the merge of chords ({G4},{B4}) and ({G4,B4}) on one hand and on the other hand from the merge of the chords ({D2,A2}) and ({D2},{A2}). This score (*i.e.* 8) is actually computed by summing the score of the permutation between {D2,A2} and {D2,A2} (*i.e.* 4) and the score between the chords ({G4},{B4}) and ({G4,B4}) (*i.e.* 4).

The final score between the two global sequences of chords S_1 and S_2 can be obtained either by merging all the chords of S_1 to the chords of S_2 (which corresponds to the cost of the optimal bipartite matching between ({G4,B4,D2,A2,G8,D8,B8}) and ({G4,B4,D2,A2,G8,D8,B8}); or from the cost described previously (between ({G4},{B4},{D2,A2}) and ({G4,B4},{D2},{A2}) for which the cost is 8) plus the cost of merging ({G8,D8,B8}) and ({G8},{D8},{B8}).

5.4 Structure Identification

Automatic analysis of the structure of musical pieces is one major open problem of the Music Information Retrieval research area. This analysis may be very useful for very different applications such as music comparison, classification or visualisation. Moreover, once the general structure is known, it is then easier to focus on each part to detect finer variations (local variations), or the repetitions of short patterns such as sequences of chords. Visualising the structure of a musical piece may also help the listener to understand the artistic intentions of the composer. Indeed, repetition is an important tool for composing music (hopefully often with variation and contrast), it is thus one of the main properties of music.

At different levels, music can be analyzed as a self-similar structure. A general structure (for example the ABA form of a minuet) is often composed of a few parts (exposition of a theme, development, *etc.*), which may be also composed of repetitions of patterns. For example [Pollack 2001], The Beatles song “Strawberry

Fields Forever" has the structure as follow:

$$I\ C\ V\ C\ V\ C\ V\ C\ O$$

where I is the Intro, V, is a Verse, C is a Chorus and O represents an Outro. The automatic extraction of these structures at different levels is a very difficult problem. It generally relies on the estimation of similarity between the different parts of a given musical piece. We have proposed in [Hanna 2008] to adapt existing methods for the estimation of the melodic similarity to compute similarity matrices that exhibit repetitions into a musical piece.

One of the main methods for the visualisation of the musical structure of audio music relies on the computation of a matrix [Foote 1999] denoted $M(i, j)$, which indicates the similarity between frames i and j of the same musical piece. Frames represent a small time interval of music. The similarity is generally indicated by a score s: the higher the score $s(i, j)$, the more similar the frames i and j. An image is then computed according to this matrix by defining each pixel color $p(i, j)$ to the corresponding similarity score $s(i, j)$. Since the similarity $s(i, i)$ is maximum, the diagonal $p(i, i)$ of such image is easily identifiable. Dark colors are associated with high similarity scores (the opposite choice is made in [Foote 1999] for example). The main diagonal then appear black on a mainly white background. Such images clearly allow the visualisation of the structure of musical pieces. Dark zones indicate similar regions. In particular, dark diagonal lines (different from the main diagonal) will exhibit repeated parts such as theme, chorus, verse, *etc.* Figure 5.10 shows an example of such a self-similarity matrix.

The musical structure is manually annotated and corresponds to dark diagonals. Existing approaches essentially consider audio signals. Features are extracted directly from audio samples. The first main features applied were the Mel-frequency cepstral coefficients (MFCC) [Foote 1999]. These coefficients are estimated from the smoothed spectrum of the audio signal and primarily describe the timbre properties of the sound. Considering only these coefficients implies limitations of structural analysis systems. Indeed, many popular music pieces are composed of one repeated chorus, played successively with different instruments : for example, the first chorus can be played by a singer with a single piano whereas the second chorus can be repeated with many other instruments (drums, guitar, *etc.*). However these different parts cannot be analyzed as repetitions of the same part. This important limitation leads researchers to consider other features describing in a better way tonal information, since the harmonic information is one of the main information (tonality, chords, etc.) that induces the structure of music. In [Bartsch 2001], chroma-based representations (chromagram or pitch class) are applied for encoding musical structure. Experiments showed that musical structure analysis is improved by using such representations. Other researches investigate the possibility to take into account local musical variations such as tempo deviations or key transpositions [Müller 2007].

Following this way, we have proposed in [Hanna 2008] to investigate methods for musical structure analysis that enable the detection of repetitions even with local

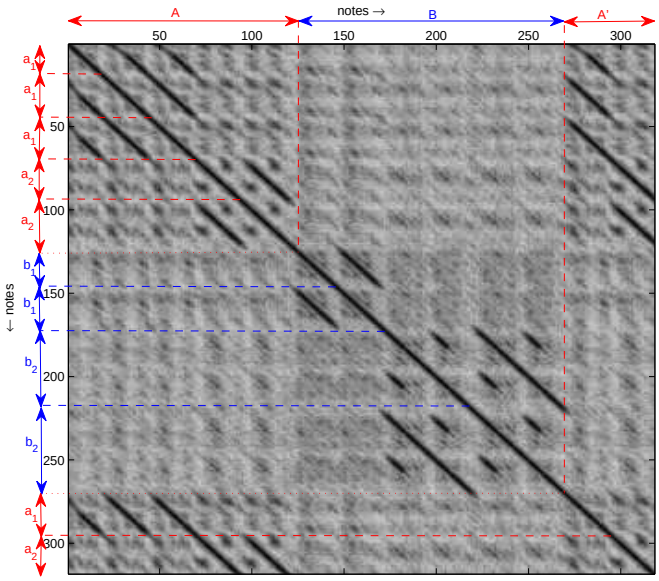

Figure 5.10: Visualisation of musical structure of the minuet part of the Water Music Suite No.1 in F (Haendel) by a self-similarity matrix computed with the method described in [Hanna 2008]. The structure has been manually annotated.

musical variations. Some recent techniques are very accurate when the different parts of the structure are repeated with few variations. If it is generally the case in the context of the Western popular music, musically similar segments may also exhibit significant variations of tonality (global or local transpositions) or tempo for example. We aim at developing a system that takes into account such variations. This investigation requires to develop systems that consider elements of musical theory, in particular information associated to harmonic content.

5.5 Elements of Implementation

The alignment algorithms mentioned above are powerful and optimal: they will always find the best alignment. However, they are also very time consuming. This drawback considerably limits their use for musical applications. For biological applications, heuristics such as BLAST and FASTA can be used to speed-up local sequence alignment while allowing for multiple regions of local similarity. These heuristics are valuable, but they may fail to report hits or may report false positives. In order to get more accurate results faster implementations are therefore of primary importance.

Graphics processing units (GPUs) have recently received lots of attention thanks to their extensive computing resources. Not only are the latest generations of GPUs very powerful graphic engines, they can also be used for general purpose computa-

tion (GPGPU) [Owens 2008]. With the recent evolutions of GPUs' architecture into a unified, highly parallel programmable processor, and the development of programming tools and high-level programming languages such as NVIDIA's CUDA, GPUs have become a very attractive, low-cost alternative to the traditional microprocessors for computationally demanding applications that can be expressed as data-parallel computations, *i.e.* the same program is executed on many data elements in parallel. We have proposed an implementation [Ferraro 2009b] of our variant of Smith-Waterman based on local transposition which illustrates the advantages of recent graphic cards as computation platforms. We proposed to experiment this implementation with a QbH application. This type of parallelism is well suited to the problem of QbH on very large scale music databases, although it also brings new challenges regarding memory operations and computational resource allocations.

5.5.1 GPU Architecture

Both AMD and NVIDIA build architectures with unified, massively parallel programmable units, which allow programmers to target that programmable unit directly instead of dividing work across multiple hardware units. More precisely, a GPU contains many streaming multiprocessors (MPs) each containing several elements including several cores, also called streaming processors (SPs), and various types of on-chip shared memories and registers. The MPs also share some constant memory areas with very fast access and a global uncached large memory with relatively low throughput and long latency. For example, the NVIDIA GeForce 9800 GX2 used for our experiments is a dual GPU engine with 256 cores (128 per GPU) running at 1.5 GHz. These cores are regrouped into 2×16 MPs which share a global memory of 1GB with a 512-bit interface width providing a throughput of 128 GB/sec (64 GB/sec per GPU).

The MPs creates, manages, and executes concurrent threads in hardware with zero scheduling overhead. To manage hundred of threads running several different programs, the multiprocessor employs an architecture called SIMT (single-instruction multiple-thread), which resembles SIMD (single-instruction multiple-data) vector organizations, *i.e.*, single instruction controls multiple processing elements. Unlike SIMD vector machines, SIMT enables programmers to write thread-level parallel code for independent, scalar threads, as well as data-parallel code for coordinated threads.

5.5.2 GPU Computing with CUDA

Our implementation uses CUDA, a general purpose parallel computing framework developed and distributed by NVIDIA for use with their recent GPUs[1]. CUDA can be seen as an extension of C that allows developers to define C functions,

[1] CUDA was introduced in November 2006 along with the G80 series. CUDA can be downloaded for free from `http://www.nvidia.com/object/cuda_home.html`. A list of CUDA-enabled product is available at `http://www.nvidia.com/object/cuda_learn_products.html`.

called *kernels* to be executed N times in parallel by N different CUDA *threads*. CUDA threads may access data from multiple memory spaces during their execution. CUDA's programming model assumes that the CUDA threads execute on a separate *device*, whereas the rest of the program runs on a CPU. In other words, the GPU operates as a coprocessor to the *host* running the C program. Both the host and device maintain their own memory areas, allowing for concurrent programming between the CPU and the GPU(s). CUDA kernels must be compiled into binary code using `nvcc`, a C compiler for CUDA. Note that `nvcc` supports C++ programming for host functions but kernels must be written in C, possibly with templates. `nvcc` also supports device emulation.

5.5.3 CUDA implementation of QbH

The process of evaluating how well each piece of music in a database match a query (sung or hummed in the case of QbH), and rank the music pieces according to this score can be parallelized at different levels. As explained earlier, our implementation uses a variant of Smith-Waterman. Although it is possible to do so [Liu 2006], our choice was not to parallelize the implementation of Smith-Waterman itself, as this approach could only provide significant improvements for extremely large sequences. In contrast, our CUDA implementation optimizes the arithmetic intensity (the ratio of arithmetic operations to memory operations) by computing in parallel all the scores of a query with every piece of music in the database. If the database contains N pieces of music, our program virtually launches N kernels executing the Smith-Waterman algorithm in parallel. The main challenges are therefore to optimize the resource allocations and the memory operations.

After the query has been converted from its original format (typically a wave audio file), we store it in a special memory area called the *texture* memory, which allows for very fast read/write operations. The texture memory is shared among all threads (Fig. 5.11). The database usually contain too many pieces of music to be stored in any of the cached, fast memories (texture, constant, shared memory). Therefore, all the pieces of music are stored in the global memory. On a GPU, the global memory is not cached, so it is extremely important to follow the right access pattern to get maximum memory bandwidth. Throughput of memory operations is 8 operations per clock cycle, plus 400 to 600 clock cycles of memory latency. Under some size and alignment conditions, the device is capable of reading data from global memory in a single load instruction. Moreover, the memory bandwidth can be used most efficiently when the simultaneous memory access by all the active threads can be coalesced into a single memory transaction. (For more details, see [NVI 2009].)

In order to satisfy all these constraints, the pieces of music of the database are store in an array of `float2`, a CUDA structure containing two 32-bit floats, which stores the pitch and duration of each note. Although the size and alignment properties are fulfilled by this data type, storing the pieces of music sequentially, one after the other, would be very inefficient since simultaneous reading by all the threads in a single transaction would be impossible. Instead, if the database contain N pieces

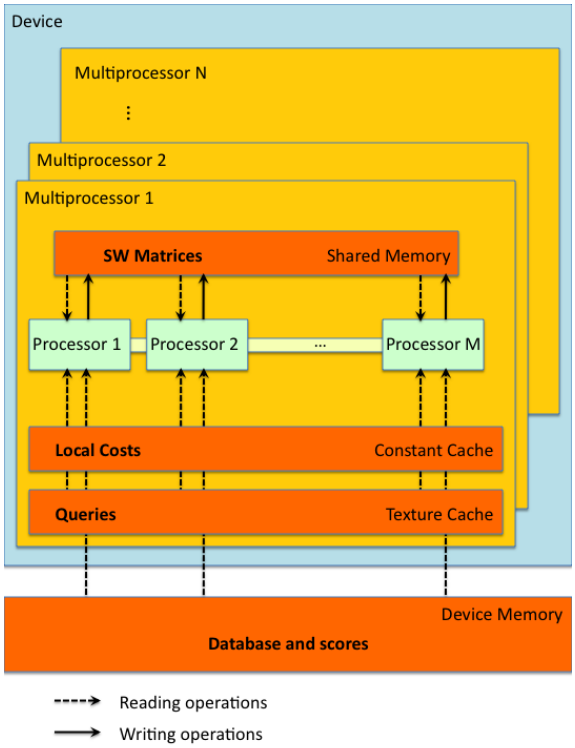

Figure 5.11: nVidia GPUs architecture. Each multi-processor executes both the conversion of queries from audio files to a vector of notes (stored in the texture memory) and the comparison between the query and each reference (stored in the device memory). Each processor store its intermediate Smith-Waterman matrices (only one row) in its own shared memory space. Constants costs and intermediate values are respectively stored in constant and shared memories.

of music, we organize the data in memory as a one dimensional array, such that its first N entries correspond to the first note (pitch, duration) of each piece; then, the next N entries correspond to the second note of each piece, etc.

Each thread performs its computations on its own matrix, more exactly on its $\Delta = 12$ transposition matrices. In order to minimize the amount of required memory, we only store the current row of each matrix. Moreover, to optimize memory alignment, allocation is based on the query's fixed size rather than the pieces of music's variable sizes. Finally, in order to allow simultaneous read/write operations by the active threads, the matrices are not stored at consecutive addresses but rather using the same strategy as the database. Fig. 5.11 describes the device architectures.

5.6 Presentation of the papers

5.6.1 Music Representation

As introduced previously, melody is an important property for the perceptual description of Western musical pieces. In the monophonic context, retrieval systems based on melodic similarity generally consider sequences of pitches and durations. We have presented and discussed in [Hanna 2007a] different symbolic representations of pitches and durations for monophonic melodies.

[Hanna 2007a] Pierre Hanna, **Pascal Ferraro** and Matthias Robine. *On Optimizing the Editing Algorithms for Evaluating Similarity Between Monophonic Musical Sequences.* Journal of New Music Research, vol. 36, no. 4, pages 267–279, 2007

The case of polyphony has been presented in [Hanna 2007b].

[Hanna 2007b] Pierre Hanna and **Pascal Ferraro**, Polyphonic music retrieval by local edition of quotiented sequences, in: Proceedings of the 5th International Workshop on Content- Based Multimedia Indexing (CBMI), Bordeaux, France, 2007, pp. 61–68.

In order to take into account important musical elements as tonality, passing notes, strong and weak beats, in [Robine 2007a] we have then proposed to improve edit-distance based algorithms by considering music theory. These elements are illustrated with a few monophonic musical examples which lead to important errors in usual systems. First experiments with these examples show that the improvements induced are significant.

[Robine 2007a] Matthias Robine, Pierre Hanna and **Pascal Ferraro**. *Music Similarity: Improvements of Edit-based Algorithms by Considering Music Theory.* In Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR), pages 135–141, Augsburg, Germany, 2007.

5.6.2 Music Comparison

Existing symbolic music comparison systems generally consider monophonic music or monophonic reduction of polyphonic music. Adaptation of alignment algorithms to the music leads to accurate systems. For instance, we have proposed an extension allowing to take into account local transposition in [Allali 2007]. However extensions to polyphonic music arise new problems.

[Allali 2007] Julien Allali, **Pascal Ferraro**, Pierre Hanna and Costas Iliopoulos. *Local Transpositions in Alignment of Polyphonic Musical Sequences.* In Nivio Ziviani and Ricardo Baeza-Yates, editeurs, 14th String Processing and Information Retrieval Symposium, volume 4726 of Lecture Notes in Computer Science, pages 26–38. Springer, oct. 2007.

In [Allali 2009a], we propose a general framework for polyphonic music which permits to directly apply the substitution score scheme set for monophonic music, and which allows new operations by extending the operations proposed by Mongeau and Sankoff.

[Allali 2009a] Julien Allali, **Pascal Ferraro**, Pierre Hanna, Costas Illiopoulos and Matthias Robine. *Toward a General Framework for Polyphonic Comparison.* Fundamenta Informaticae, Accepted for Publication, 16 pages, 2009.

5.6.3 Self-similarity in Music

Motivated by the identification of the musical structure of pop songs or classical music, we present in [Hanna 2008] a method for visualising the musical structure of symbolic monophonic music.

[Hanna 2008a] Pierre Hanna, Matthias Robine and **Pascal Ferraro**. *Visualisation of Musical Structure by Applying Improved Editing Algorithms.* In Proceedings of the International Computer Music Conference (ICMC), page to appear, Belfast, Northern Ireland, 2008.

Based on a generalisation of string pattern matching techniques, in [Allali 2009] we have then introduced combinatorial problems involving overlays (non-overlapping substrings) and the covering of a text S_1 by them.

[Allali 2009] Julien Allali, Pavlos Antoniou, **Pascal Ferraro**, Costas Iliopoulos and Spiros Michalakopoulos. *Overlay Problems for Music and Combinatorics.* In Proceedings of the 15th International Conference on Auditory Display, page to appear, Copenhagen, Dn, May 2009. 15

5.6.4 Applications

In the context of Simbals project, we have developed a set of applications to analyse music. The number of copyright registrations for music documents is increasing each year. Computer-based systems may help to detect near-duplicate music documents and plagiarisms. We have developed an improved system which mainly considers

melody but takes also into account elements of music theory in order to detect musically important differences between sequences.

In [Robine 2007b], we present the improvements proposed by our system in the context of the near-duplicate music document detection. Several experiments with famous music copyright infringement cases are proposed. In both monophonic and polyphonic context, the system allows the detection of plagiarisms.

[Robine 2007b] Matthias Robine, Pierre Hanna, **Pascal Ferraro** and Julien Allali. *Adaptation of String Matching Algorithms for Identification of Near- Duplicate Music Documents.* In Proceedings of the International SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN), pages 37–43, Amsterdam, Netherlands, 2007.

In [Ferraro 2009b], we present a fast implementation of our local alignment method, which allows to retrieve a hummed query in a database of MIDI files, with good accuracy, in a time up to 160 times faster than other comparable systems.

[Ferraro 2009b] **Pascal Ferraro**, Pierre Hanna, Laurent Imbert and Thomas Izard, Accelerating Query-by-Humming on GPU, in: 10th International Society for Music Information Retrieval Conference, Kobe Japon, 2009, p. to appear, 6.

What's Next ?

6

Deciphering the complex organization of biological structures is one of the key challenges in modern biology. Complexity can be due to a variety of causes, *e.g.* multitude of processes, feedback and regulation mechanisms, superposition of processes at different scales. A lot of work has been developed in the past two decades on biological sequences. This is remarkably illustrated by the wealth of theoretical developments that has been made for the analysis and interpretation of genomic data, for which genomes are considered as sequences of 4 letters coding for genes. Biological sequences are also used at more macroscopic scales for the description of certain biological organisms. To study plant development for instance, biologists study the growth process or the branching process of axes as sequences of discrete events along the axes. At both scales, the complexity of the investigation is tightly connected with the intrinsic complexity of the studied structure, *i.e.* of biological sequences.

As we have seen, some works started to investigate the analysis and interpretation of more complex biological structures such as trees, that can be found in many domains of biology. For instance, plant structures are naturally trees of axes or trees of internodes. In the same way, RNA secondary structures are modelled by trees or arc-annotated sequences. Huge databases have been constituted for this type of biological data. Exchanging and sharing these huge tree-structured databases is more and more a necessity for the scientific community. In contrast with sequences, trees show a higher degree of organization (in particular, trees are made-up of sequences). Due to this increase of structural complexity, algorithms for analyzing trees have in general a higher complexity than their equivalent for analyzing sequences. As

a consequence, the number of methods available for biologist to investigate tree-structured data remains very limited.

Novel battery of algorithms and models for the analysis of tree-structured data must then be developed, in the spirit of the family of algorithms that were developed for sequence analysis in bioinformatics or in plant science. To achieve this goal, research might develop approaches around the notion of **data compression**. The ability to compress complex data is fundamental to its understanding. It makes it possible to remove its internal redundancy and, by emphasizing its deep incompressible nature, to provide new insights into the original data. Two types of compression strategy might be investigated which respectively make use of structural and statistical properties of trees. Those in the former class would replace long repeated subtrees by a pointer to an earlier instance of the subtrees or to an entry in a dictionary. Examples in this category are the popular Lempel-Ziv compression algorithms and their variants. Since plant architectures are known to exhibit repetitions of similar structures, we aim at generalizing structural methods to compress sequences to tree-structured data. In a statistical context, compression is intimately related to model selection, *i.e.* choosing a model that realizes the best compromise between good fitting of the observed data and parsimony of the model.

6.1 From An Algorithmic Perspective

6.1.1 Compression Algorithms

The Lempel-Ziv algorithm (LZ'77) is one of the most popular algorithms to compress strings. This algorithm is based on a data structure on texts called suffix trees. The principle of this algorithm consists in coding the text by dynamically building (simultaneously with text reading) a dictionary of words already read in the text. Then the text is coded by references to the words available in the dictionary. Decoding retransforms the code into the original text (without any loss) by maintaining in parallel the dictionary. This compression method is successful on strings but needs adaptations on tree-structured data.

6.1.2 Tree Indexing Structures

The manipulation of huge textual databases leads to the development of specific data structures and efficient algorithms to manipulate them. The suffix tree is used in this context to the construction of index. However the development of the suffix tree has been made for the research of text in strings or sequences. The generalization of the concept of suffix tree (or suffix array) to the construction of indices of databases of trees is one of the next challenge. Such generalization will allow efficient identification of patterns in tree structured data.

6.1.2.1 Self-nestedness

As presented in Section 3.3.1, first works has introduced the definition of nested trees. In this definition, trees exhibit a recursive structure. The main problem is to identify the redundancy within these trees in order to propose compressed representations as Directed Acyclic Graphs (DAGs). This means that efficient subtree isomorphisms must be proposed. Existing algorithms may be explored and adapted in order to propose efficient algorithms to compress trees into a DAG. To quantify the degree of self-nestednness of a tree T, one must evaluate how far T is from perfect self-nestedness. To formalize this idea, let us consider S the set of all the finite self-nested trees and a distance D defined on the set of all trees. Let us call $NST(T)$ the set of self-nested trees with minimal distance to T. In general for topological distances, there exists more than one self-nested tree S^* with minimal distance $D(T, S^*)$. This quantity characterizes how distant the tree T is from the nearest self-nested tree and therefore is a good candidate to quantify the degree of self-nestedness of T.

In this definition, D can be any distance between two trees. A particular choice of D corresponds to different definitions of the NST of a tree. We have proposed a first solution in [Ferraro 2009a] with a very constraint distance. To keep going further, the different possible choices of D depending on the applications must be studied. Function of the different distances, the algorithms to evaluate the nearest self-nested tree may then be developed.

6.2 From the Application Point of View

6.2.1 Plant Modelling

Internal similarities and nested structures in plants have for long been identified by botanists as key feature in the general organization of plants and the dynamics of their development (see Chapter 3). Different techniques were used until now ranging from qualitative analysis, with the help of descriptions based on botanical drawing or pictures, to various approaches of quantitative analysis.

Methods for the systematic analysis of the internal similarities of plant branching systems are important for two major reasons:

- First, they may be used to reveal structures of plants deeply hidden in complex branching systems. If all the redundancy that is expressed in the branching structure of a plant is removed, we would be likely to characterize the deep structure of the plant. Being simpler, this structure could be easier to interpret, to characterize or to compare with the corresponding structure of other plants.
- Second, the fact that plants are made up of the repetition of many similar components at different scales provides macroscopic presumptions for the existence of similarities in processes that drive meristem activity at microscopic

scales. Thus characterizing the internal similarities of plants is expected to give important clues to understand meristem functioning.

Many methods have been developed for analyzing sequences of events (*e.g.* corresponding to branching structures or to the succession of growth phases along the main axis) extracted from plant structure. Comparatively less attention has been paid to the development of methods for directly characterizing tree-like structures. However, the natural organization of plants is primarily observed at the level of branching systems that qualitatively show strong internal similarities between their own parts. Different types of remarkable patterns have been identified within plants:

- At a local scale, repeated patterns or motifs corresponding to frequent successions of entities in the form sub-trees of limited depth (e.g. corresponding to a parent and its child vertices in the most simple case). Typically, the sympodial growth, with the death or transformation of the apical meristem, induces repeated patterns.
- At a more macroscopic scale, homogeneous zones within tree structure where the morphological characteristics of the plant entities do not change substantially within each zone, but change markedly between zones.
- Nested structuring where "typical" branching systems seem to be nested one into the others. At the finest scale, elementary branching systems may correspond to specific motifs.

Although these remarkable patterns have been empirically described by botanists for decades, only few algorithmic or statistical modeling approaches were developed yet to identify and characterize similar, possibly nested, patterns in plant structures.

6.2.2 RNA Secondary Structures

From the RNA perspective, the characterization of a simple signature of a family of RNA will allow to better identify new candidates within a genome. The main issue of algorithms designed to identify genes is to capture a maximum sensitivity. However if the specificity is too accurate, a lot of false events may be captured. Thus to relax the characteristics describing a particular family researches might propose to introduce in RNA gene finder algorithms compressed version of the structure.

Several classes of computer methods have been developed to locate non-coding RNA genes (ncRNAs). The main class contains tools that exploit well-characterized primary and secondary structure elements (RnaMot, MilPat, Patscan, Rnamotif). Other classes exploit data from a number of sources including:

- comparative sequence analysis between genomes of related species,
- energy minimization in non coding regions and,
- the base composition of ncRNAs compared to coding regions (QRNA ...).

Considering the first class, we can distinguish approaches that are tailored to an RNA family (tRNAscan-SE for tRNAs) and the general-purpose programs that describe any type of ncRNAs by its primary and secondary features (rnamot *etc.*).

The main advantage of general-purpose programs is that a description can be easily modified to run a new search by the biologist, more or less sensitive depending of the knowledge of the characteristics shared by the RNA family. For programs tailored to a specific RNA family, it is not possible to modify the pattern as easily because the description of the pattern and the algorithm that searches for it are intimately associated, however, they appear as the most specific software.

Furthermore, one main challenge in Bioinformatics when working with RNA is to deal with the 3D structure (considering all interactions between 2D elements). This level of analysis, more realistic, brings lot of problems with mathematical modeling. One example is given by the distance with long-range contacts that define the sub-domains of RNA and promote interaction with ligands. In the literature, the description of anchoring motifs can be found. These motifs can be shared in different RNA family with a non similar secondary structure.

For some RNA family, the presence of secondary elements is not enough important to be used as a signature. As example, snoRNAs are a class of small RNA molecules that guide chemical modifications (methylation or pseudouridylation) of ribosomal RNAs (rRNAs) and other RNA genes (tRNAs and other small nuclear RNAs (snRNAs)). They both interact with RNA and proteins to make their biochemical role. The k-turn motif is therefore present in these molecules and is involved in the interaction of snoRNAs with proteins.

The representation of RNA through compressed trees would give a simplification of the 2D structure to help to identify interactions to a higher level. Then, the secondary elements could be simplified and the compressed view could help to only focus on their interactions. A comparative analysis of these representations from different families would be handy to study anchoring patterns involved in long range RNA-RNA or RNA-proteins interactions.

6.2.3 Symbolic Music Analysis

We have seen in this report that musical pieces can be represented by sequences or trees of symbols. They can thus be analyzed in the same way bioinformatic data are; and the the former compression methods or indexing structures might be developed in the context of symbolic music analysis.

A music similarity system can help a music consumer to find new musical pieces by finding the music that is most musically similar to specific query songs, or is nearest to songs that the consumer already likes. Therefore, the main applications of these systems that estimate the similarity between audio musical pieces are the Query-by-Humming/Singing/Whistling and Query-by-Example systems. The main idea of these applications is to help users to retrieve musical pieces from a single melody line (whistled, hummed or sung) or an audio extract of the piece searched.

Another interesting application of such systems is the retrieval of cover songs. These systems need the development of efficient methods allowing the identification of a particular pattern.

Three particular challenging applications will finally benefit of such algorithms:

- Applications on music of the researches about self similarity lead to algorithms that would be able to detect the different patterns of any musical pieces. These patterns may be very representative of the musical pieces analyzed: theme, verse, chorus, *etc.* One of the main applications induced is the automatic generation of audio summaries. Instead of being represented by the first seconds, musical pieces can be represented by a few seconds of the verse and then a few seconds of the chorus. Such summary would certainly be more representative of the original piece.

- Advances on musical structure analysis, and, more generally, on musical properties analysis have to lead to systems that helps users to listen a musical piece by showing them additional information such as musical structure, rhythmic properties, *etc.* Such augmented listening could also be applied in a pedagogic context. A score and its human interpretation can be compared by measuring the different deviations with similarity methods. It can lead to a musical application, useful for self-learning an instrument or to evaluate the technical level of a musician for example.

- New browsing systems that allow users to discover musical pieces in huge databases depending on musical properties shall be developed. Since several similarities may be analyzed, users could select pieces that are similar with some properties but dissimilar considering other properties. For example, given a piece, users could be proposed new songs that are rhythmically similar, but with totally different chord progressions. Such systems may also lead to music recommendation systems.

Bibliography

[Aho 1974] A.V. Aho, I.E. Hopcroft and J.D. Ullman. The design and analysis of computer algorithms. Addison-Wesley, Reading, MA, 1974. 38, 40

[Akers 1978] S. Akers. *Binary decision diagrams.* IEEE Tansactions on computers, vol. C-27, no. 6, pages 509–516, 1978. 38, 40

[Allali 2005a] J. Allali and M.-F. Sagot. *A Multiple Graph Layers Model with Application to RNA Secondary Structures Comparison.* In String Processing and Information Retrieval 2005, volume 3772, pages 348–359, 2005. 49, 57, 48

[Allali 2005b] J. Allali and M.-F. Sagot. *A New Distance for High Level RNA Secondary Structure Comparison.* IEEE/ACM Trans. Comput. Biol. Bioinformatics, vol. 2, no. 1, pages 3–14, 2005. 47

[Allali 2007] J. Allali, P. Ferraro, P. Hanna and C. Iliopoulos. *Local Transpositions in Alignment of Polyphonic Musical Sequences.* In Nivio Ziviani and Ricardo Baeza-Yates, editeurs, 14th String Processing and Information Retrieval Symposium, volume 4726 of *Lecture Notes in Computer Science*, pages 26–38. Springer, oct. 2007. 71, 61

[Allali 2008] J. Allali, Y. d'Aubenton Carafa, C. Chauve, A. Denise, C. Drevet, P. Ferraro, D. Gautheret, C. Herrbach, F. Leclerc, A. de Monte, A. Ouangraoua, M.F. Sagot, C. Saule, M. Termier, C. Thermes and H. Touzet. *Benchmarking RNA secondary structure comparison algorithms.* In Actes des Journées Ouvertes de Biologie, Informatique et Mathématiques - JOBIM'08, pages 67–68, 2008. 57

[Arber 1950] A. Arber. Natural philosophy of plant form. University Press, Cambridge, 1950. 37, 39, 40, 41

[Barthélémy 1989] D. Barthélémy, C. Edelin and F. Halle. *Architectural concepts for tropical trees.* In L. B. Holm-Nielsen, I. C. Nielsen and E. Balslev, editeurs, Symposium on Tropical Forests, Tropical forests: Botanical dynamics, speciation and diversity, pages 89–100, Aarhus, Danemark, 1989. Academic Press, London. 37, 39

[Barthélémy 1991a] D. Barthélémy. *Levels of organization and repetition phenomena in seed plants.* Acta Biotheoretica, vol. 39, pages 309–323, 1991. 26, 37, 29, 39

[Barthélémy 1991b] D. Barthélémy. *Levels of organization and repetition phenomena in seed plants.* Acta Biotheoretica, vol. 39, pages 309–323, 1991. 39, 38

[Barthélémy 1997] D. Barthélémy, Y. Caraglio and E. Costes. *Architecture, gradients morphogénétiques et âge physiologique chez les végétaux.* In J. Bouchon, P. de Reffye and D. Barthélémy, editeurs, Modélisation et Simulation de l'Architecture des Végétaux, Science Update, pages 89–136. INRA Editions, Paris, France, 1997. 37, 40, 39

[Bartsch 2001] M. A. Bartsch and G. H. Wakefield. *To Catch a Chorus: Using Chroma-Based Representations for Audi Thumbnailing.* In Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pages 15–18, New Paltz, USA, 2001. 80

[Bello 2007] J.P. Bello. *Audio-based Cover Song Retrieval using Approximate Chord Sequences: Testing Shifts, Gaps, Swaps and Beats.* In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR), pages 239–244, Vienna, Austria, September 2007. 62, 64

[Blin 2006] G. Blin and H. Touzet. *How to Compare Arc-Annotated Sequences: The Alignment Hierarchy.* In F. Crestani, P. Ferragina and M. Sanderson, editeurs, 13th SPIRE, volume 4209, pages 291–303, Glasgow, UK, October 2006. 46, 57

[Blin 2007] G. Blin, C. Chauve, G. Fertin, R. Rizzi and S. Vialette. *Comparing genomes with duplications: a computational complexity point of view.* IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 4, no. 4, pages 523–534, October 2007. 46, 57

[Bryant 1986] R. Bryant. *Graph-based algorithms for boolean function manipulation.* IEEE Transactions on Computers, vol. C-35, no. 8, pages 677–691, 1986. 38, 40

[Caraglio 1997] Y. Caraglio and D. Barthélémy. *Revue critique des termes relatifs à la croissance et à la ramification des tiges des végétaux vasculaires.* In INRA édition, editeur, Modélisation et simulation de l'architecture des végétaux, chapitre Part I, pages 11–87. Science Update, 1997. 27, 30

[Collins 2000] L.J. Collins, V. Moulton and Penny D. *Use of RNA secondary structure for studying the evolution of RNase P and RNase MRP.* Journal of Molecule Evolution, vol. 51, no. 3, pages 194–204, 2000. 47

[Costes 1998] E. Costes, H. Sinoquet, C. Godin and J.J. Kelner. *3D digitizing based on tree topology : application to study the variability of apple quality within the canopy.* Acta Horticulturae, vol. 499, pages 271–280, 1998. in press. 31, 32

[Dannenberg 2007] R. B. Dannenberg, W. P. Birmingham, B. Pardo, N. Hu, C. Meek and G. Tzanetakis. *A Comparative Evaluation of Search Techniques for Query-by-Humming Using the MUSART Testbed.* Journal of the American Society for Information Science and Technology (JASIST), vol. 58, no. 5, pages 687–701, 2007. 62, 64

[de Reffye 1989] P. de Reffye, E. Elguero and E. Costes. *Growth units construction in trees: a stochastic approach.* Acta Biotheoretica, vol. 39, pages 325–342, 1989. 31, 39, 40, 32, 38

[Deussen 2005] Olivier Deussen and Bernd Lintermann. Digital design of nature. Springer-Verlag, 2005. 37, 39

[Doraisamy 2003] S. Doraisamy and S. Ruger. *Robust Polyphonic Music Retrieval with N-grams.* Journal of Intelligent Information Systems, vol. 21, no. 1, pages 53–70, 2003. 62, 63, 64

[Downie 2008] J. S. Downie, M. Bay, A. F. Ehmann and M. C. Jones. *Audio Cover Song Identification: MIREX 2006-2007 Results and Analyses.* In Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08), September 14-18 2008. 62, 64

[Dulucq 2003] S. Dulucq and L. Tichit. RNA *secondary structure comparison: exact analysis of the* Zhang-Shasha *tree edit algorithm.* Theoretical Computer Science, vol. 306, pages 471–484, 2003. 53

[Durand 2005] J-B. Durand, Y. Guédon, Y. Caraglio and E. Costes. *Analysis of the plant architecture via tree-structured statistical models: the hidden Markov tree models.* New Phytol, vol. 166, no. 3, pages 813–825, Jun 2005. 37, 39

[Edmonds 1972] J. Edmonds and R. M. Karp. *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems.* Journal of the Association for Computing Machinery, vol. 19, pages 248–264, 1972. 75

[Ferraro 2000] P. Ferraro and C. Godin. *A distance measure between plant architectures.* Annals of Forest Science, vol. 57, no. 5/6, pages 445–461, 2000. 28, 37, 38, 41, 31, 39, 40

[Ferraro 2003a] P. Ferraro and C. Godin. *An edit distance between quotiented graphs.* Algorithmica, vol. 36, pages 1–39, 2003. 35, 36, 50, 55, 19, 34

[Ferraro 2003b] P. Ferraro and C. Godin. *Optimal mappings with minimum number of connected componanents in tree-to-tree comparison problems.* Journal of Algorithms, vol. 48, pages 385–406, 2003. 35, 19

[Ferraro 2005] P. Ferraro, C. Godin and P. Prusinkiewicz. *Toward a quantification of self-similarity in plants.* Fractals, vol. 13, no. 2, pages 91–109, 2005. 37, 39, 41

[Ferraro 2009a] P. Ferraro, Godin C. and P. Prusinkiewicz. *Quantifying the degree of self-nestedness of trees. Application to the structural analysis of plants.* IEEE/ACM Transactions on Computation Biology and Bioinformatics, vol. in press, page 16, 2009. 39, 91, 41

[Ferraro 2009b] Pascal Ferraro, Pierre Hanna, Laurent Imbert and Thomas Izard. *Accelerating Query-by-Humming on GPU.* In 10th International Society for

Music Information Retrieval Conference, page to appear, Kobe Japon, 10 2009. 6. 82

[Foote 1999] J. Foote. *Visualizing Music and Audio Using Self-Similarity.* In Proceedings of the seventh ACM international conference on Multimedia (ACMMM), pages 77–80, Orlando, USA, 1999. 80

[Frijters 1976a] D. Frijters and A. Lindenmayer. *Developmental descriptions of branching patterns with paracladial relationships.* Automata, Langages, Development, 1976. 40, 38

[Frijters 1976b] D. Frijters and A. Lindenmayer. *Developmental descriptions of branching patterns with paraclidial relationships.* In G. Rozenberg and A. Lindenmayer, editeurs, Formal Languages, Automata and Development, pages 57–73, Noordwijkerhout, The Netherlands, 1976. North-Holland Publishing Company. 37, 39

[Gatsuk 1980] L. E. Gatsuk, O. V. Smirnova, L. I. Vorontzova, L. B. Zaugolnova and L. A. Zhukova. *Age states of plants of various growth forms : a review.* J. Ecol., vol. 68, pages 675–696, 1980. 40

[Godin 1997] C. Godin, Y. Guédon, E. Costes and Y. Caraglio. *Measuring and analyzing plants with the AMAPmod software.* In M.T. Michalewicz, editeur, Plants to ecosystems - Advances in Computational Life Sciences, 2nd International Symposium on Computer Challenges in Life Science, pages 53–84. CSIRO Australia, Melbourne, Australia, 1997. 31, 37, 32, 39

[Godin 1998] C. Godin and Y. Caraglio. *A multiscale model of plant topological structures.* Journal of theoretical biology, vol. 191, pages 1–46, 1998. 27, 28, 32, 37, 38, 39, 49, 30, 33, 40

[Godin 2005] C. Godin, E. Costes and H. Sinoquet. *Plant architecture modelling - virtual plants and complex systems.* In C. Turnbull, editeur, Plant Architecture and its Manipulation, volume 17 of *Annual plant reviews*, pages 238–287. Blackwell, 2005. 37, 39

[Greenlaw 1996] R. Greenlaw. *Subtree Isomorphism is in DLOG for Nested Trees.* International Journal of Foundations of Computer Science, vol. 7, no. 2, pages 161–168, 1996. 39, 41

[Guédon 1998] Y. Guédon and E. Costes. *A statistical approach for analyzing sequences in fruit tree architecture.* In Proceedings of Fifth international symposium on computer modelling in fruit research and orchard management, 1998. 31, 33

[Guédon 2001] Y. Guédon, D. Barthélémy, Y. Caraglio and E. Costes. *Pattern analysis in branching and axillary flowering sequences.* J Theor Biol, vol. 212, no. 4, pages 481–520, Oct 2001. 31, 37, 33, 39

[Guédon 2003] Y. Guédon, P. Heuret and E. Costes. *Comparison methods for branching and axillary flowering sequences.* J Theor Biol, vol. 225, no. 3, pages 301–325, Dec 2003. 27, 37, 30, 39

[Guignon 2005] V. Guignon, C. Chauve and S. Hamel. *An Edit Distance Between RNA Stem-Loops.* In SPIRE, pages 335–347, 2005. 57

[Guo 2006] Y. Guo, Y. Ma, Z. Zhan, B. Li, M. Dingkuhn, D. Luquet and P. De Reffye. *Parameter optimization and field validation of the functional-structural model GREENLAB for maize.* Ann Bot (Lond), vol. 97, no. 2, pages 217–230, Feb 2006. 37, 39

[Gusfield 1997] D. Gusfield. Algorithms on strings, trees and sequences - computer science and computational biology. Cambridge University Press, Cambridge, 1997. 62, 61

[Hallé 1978a] F. Hallé, R. A. A. Oldeman and P. B. Tomlinson. Tropical trees and forests. an architectural analysis. Springer-Verlag, New-York, 1978. 37, 39

[Hallé 1978b] F. Hallé, R.A.A. Oldeman and P.B. Tomlinson. Tropical trees and forests. an architectural analysis. Springer Verlag, New York, 1978. 39, 38

[Hanan 1997] J. S. Hanan and P. Room. *Practical aspects of plant research.* In M. Michalewicz, editeur, Advances in computational life sciences, Vol I : Plants to Ecosystems, volume January, pages 28–43. CSIRO, Australia, 1997. Chapitre 2. 31, 32

[Hanna 2007] P. Hanna, P. Ferraro and M. Robine. *On Optimizing the Editing Algorithms for Evaluating Similarity Between Monophonic Musical Sequences.* Journal of New Music Research, vol. 36, no. 4, pages 267–279, 2007. 63, 72, 62

[Hanna 2008] P. Hanna, M. Robine and P. Ferraro. *Visualisation of Musical Structure by Applying Improved Editing Algorithms.* In Proceedings of the International Computer Music Conference (ICMC), Belfast, Northern Ireland, 2008. 62, 80, 81, 63, 64

[Hanna 2009] P. Hanna and M. Robine. *Query by Tapping System Based on Alignment Algorithm.* In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taïwan, 2009. Institute of Electrical and Electronics Engineers (IEEE). 62, 64

[Harper 1986] J. L. Harper, B. R. Rosen and J. White. The growth and form of modular organisms. The Royal Society, London, UK, 1986. 37, 39

[Hart 1991] J. Hart and T. DeFanti. *Efficient anti-aliased rendering of 3D linear fractals.* In Thomas W. Sederberg, editeur, Computer Graphics, volume 25, pages 91–100, 1991. 38, 39, 40

[Hart 1992] J. Hart. *The object instancing paradigm for linear fractal modeling.* In Graphics Interface, pages 224–231, 1992. 39, 40

[Herman 1975] G.T. Herman, A. Lindenmayer and G. Rozenberg. *Description of developmental languages using recurence system.* Mathematical systems theory, vol. 8, no. 4, pages 316–341, 1975. 40, 38

[Höchsmann 2003] M. Höchsmann, T. Töller, R. Giegerich and S. Kurtz. *Local Similarity in* RNA *Secondary Structures.* In Proceedings of Computational Systems Bioinformatics, (CSB'03), pages 159–168, 2003. 47, 50, 57

[Höchsmann 2004] M. Höchsmann, B. Voss and R. Giegerich. *Pure multiple RNA secondary structure alignments: a progressive profile approach.* IEEE/ACM transactions on computational biology and bioinformatics, vol. 1, no. 1, pages 53–62, 2004. 47

[Ituarte 2006] I. Ituarte, A. Ouangraoua and P. Ferraro. *PyTreeMatch : a python based software for comparing RNA Secondary Structure.* In JOBIM, page 127, Bordeaux, 2006. 47

[Jiang 1994] T. Jiang, L. Wang and K. Zhang. *Alignment of trees - an alternative to tree edit.* In Combinatorial Pattern Matching'94, 5th Annual Symposium, pages 75–86, 1994. 47, 50

[Jiang 2002] T. Jiang, G. Lin, B. Ma and K. Zhang. *A general edit distance between RNA structures.* Journal of Computationnal Biology, vol. 9, no. 2, pages 371–388, 2002. 46

[Kachouri 2005] R. Kachouri, V. Stribinskis, Y. Zhu, K.S. Ramos, Westhof E. and Y. Li. *A surprisingly large RNase P RNA in Candida glabrata.* RNA (New York, N.Y.), vol. 11, no. 7, pages 1064–1072, 2005. 47

[Kay 1986] T. Kay and J. Kajiya. *ray tracing complex scenes.* In Proceedings of SIGGRAPH'86, volume 20, pages 269–278, Dallas, August 1986. 39, 40

[Kececioglu 1995] J.D. Kececioglu and E. W. Myers. *Combinatorial algorithms for DNA assembly.* Algorithmica, vol. 13, pages 7–51, 1995. 31, 33

[Kilpeläinen 1995] P. Kilpeläinen and H. Mannila. *Ordered and Unordered Tree Inclusion Problem.* SIAM Journal on Computing, vol. 24, no. 2, pages 340–356, 1995. 37, 40

[Klein 2003] R. J. Klein and S. R. Eddy. *RSEARCH: Finding homologs of single structured RNA sequences.* BMC Bioinformatics, vol. 4, page 44, 2003. 47

[Kron 2004] B. Kron. *Growth of self-similar graphs.* Journal of Graph Theory, vol. 45, no. 3, pages 224–239, 2004. 39, 41

[Kuppers 1985] M. Kuppers. *Carbon relations and competition between woody species in a Central European hedgerow.* Oecologia, vol. 66, no. Growth form and partitioning, pages 343–352, 1985. 31, 32

[Kurt 1994] W. Kurt. *Growth grammar interpreter GROGRA 2.4: A software for the 3-dimentional interpretation of stochastic, sensitive growth grammar in the context of plant modelling.* Introduction and reference manual, Forschungszentrum Waldokosysteme der Universitat Gottingen, 1994. 31, 32

[LeDizès 1997] S. LeDizès, P. Cruiziat, A. Lacointe, H. Sinoquet, X. LeRoux, P. Balandier and P. Jacquet. *A Model for Simulating Structure-Function Relationships in Walnut Tree Growth Processes.* Silva Fennica, vol. 31, pages 313–328, 1997. 31, 32

[Lemström 2000] K. Lemström. *String Matching Techniques for Music Retrieval.* Computer science, University of Helsinki, 2000. 63, 58

[Lemström 2006] K. Lemström and A. Pienimäki. *Approaches for Content-Based Retrieval of Symbolically Encoded Polyphonic Music.* In Proceddings of the 9th International Conference on Music Perception and Cognition (ICMPC), Bologna, Italy, 2006. 65

[Lerdahl 1985] F. Lerdahl and R. Jackendoff. A generative theory of tonal music. Cambridge, Massachussetts: MIT Press, 1985. 66

[Levinson 1978] S.E. Levinson, A. E. Rosenberg and S. E. Levinson. *Evaluation of a word recognition system using syntax anlysis.* Bell Syst. Tech. Journal, vol. 57, pages 1619–1626, 1978. 31, 33

[Lindenmayer 1968] A. Lindenmayer. *Mathemathical models for cellular interaction in development.* Journal of theoretical biology, vol. 18, pages 280–315, 1968. 40, 38

[Liu 2006] Y. Liu, W. Huang, J. Johnson and S. Vaidya. *GPU Accelerated Smith-Waterman.* In Computational Science, ICCS 2006, volume 3994 of *Lecture Notes in Computer Science*, pages 188–195. Springer, 2006. 83

[Lu 1979] S.-Y. Lu. *A tree-to-tree distance and its application to cluster Analysis.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 1, pages 219–224, 1979. 32, 33

[Madsen 2008] Soren Madsen, Rainer Typke and Gerhard Widmer. *Automatic Reduction of MIDI Files Preserving Relevant Musical Content.* In Proceedings of the 6th International Workshop on Adaptive Multimedia Retrieval (AMR'08), Berlin, 2008. 65

[Mandelbrot 1983] B.B. Mandelbrot. The fractal geometry of nature. W.N. Freeman, New York, USA, 1983. 39, 38

[Mech 1996] R. Mech and P Prusinkiewicz. *Visual models of plants interacting with their environment.* In H. Rushmeier, editeur, SIGGRAPH 96 Conference Proceedings, pages 397–410, New Orleans (Louisiana, USA), 1996. Addison-Wesley. 31, 32

[Mendes Soares 2006] L. M. Mendes Soares and J. Valcàrcel. *The expanding transcriptome: the genome as the 'Book of Sand'.* The EMBO Journal of Algorithms, vol. 25, pages 923–931, 2006. 45

[Miclet 1984] L. Miclet. Méthodes structurelles pour la reconnaissance des formes. Eyrolles, 61, bd Saint-Germain - 75005 Paris, 1984. 31, 32

[Mongeau 1990] M. Mongeau and D. Sankoff. *Comparison of Musical Sequences.* Computers and the Humanities, vol. 24, no. 3, pages 161–175, 1990. 62, 63, 71, 72, 58, 60, 64

[Müller 2007] M. Müller and F. Kurth. *Towards structural analysis of audio recordings in the presence of musical variations.* EURASIP J. Appl. Signal Process., vol. 2007, no. 1, pages 163–181, 2007. 80

[Mündermann 2003] L. Mündermann. *Inverse modeling of plants.* PhD thesis, University of Calgary, 2003. 40, 38

[Mündermann 2005] L. Mündermann, Y. Erasmus, B. Lane, E. Coen and P. Prusinkiewicz. *Quantitative modeling of Arabidopsis development.* Plant Physiol, vol. 139, no. 2, pages 960–968, Oct 2005. 37, 39

[Needleman 1970] S. Needleman and C. Wunsch. *A General Method Applicable to the Search for Similarities in the Amino Acid Sequences of Two Proteins.* Journal of Molecular Biology, vol. 48, pages 443–453, 1970. 67, 68, 70

[Noetzel 1983] A S. Noetzel and S. M. Selkow. *An Analysis of the General Tree-Editing Problem.* In David Sankoff and Joseph B. Kruskal, editeurs, Time Wraps, Strings Edits, and Macromolecules: the theory and practice of sequence comparison, chapitre 8, pages 237–252 ,. Addison-Wesley Publishing Company Inc, University of Montreal, Quebec, Canada, 1983. 32, 33

[Nozeran 1984] R. Nozeran. *Integration of organismal development.* In P.W. Barlow and D.J. Carr, editeurs, Positional controls in plant development, pages 375–401. Cambridge University Press, 1984. 40

[NVI 2009] NVIDIA CUDA. *Programming Guide*, April 2009. Version 2.2. Available at `http://www.nvidia.com/object/cuda_home.html`. 83

[Orio 2006] N. Orio. *Music Retrieval: A Tutorial and Review.* Foundations and Trends in Information Retrieval, vol. 1, no. 1, pages 1–90, 2006. 62, 64

[Ouangraoua 2007] A. Ouangraoua, P. Ferraro, S. Dulucq and L. Tichit. *Local similarity between quotiented ordered trees.* Journal of Discrete Algorithms, vol. 5, no. 1, pages 23–35, 2007. 38, 47, 49, 50, 52, 54, 55, 57, 19, 40, 53

[Ouangraoua 2009a] A. Ouangraoua and P. Ferraro. *A constrained edit distance algorithm between semi-ordered trees.* Theoretical Computer Science, vol. 410, no. 8-10, pages 837–846, 2009. 28, 29, 32, 33, 35, 19, 31

[Ouangraoua 2009b] A. Ouangraoua and P. Ferraro. *A new constrained edit distance between quotiented ordered trees.* Journal of Discrete Algorithms, vol. 7, no. 1, pages 78–89, 2009. doi:10.1016/j.jda.2008.05.001. 47, 49, 55, 56, 19

[Owens 2008] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips. *GPU Computing.* Proceedings of the IEEE, vol. 96, no. 5, pages 879–899, May 2008. 82, 65

[Peckam 1995] S. Peckam. *New results for self-similar trees with applications to river networks.* Water Resource Res., vol. 31, pages 1023–1029, 1995. 39, 41

[Pollack 2001] A.W. Pollack. *'Notes on...' series. The Official rec.music Beatles Home Page.* 1989-2001. 79

[Preparata 1973] F. Preparata and R. Yeh. Introduction to discrete structures for computer science and engineering. Addison-Wesley, Reading Menlo Park London, 1973. 29

[Prusinkiewicz 1990] P. Prusinkiewicz and A. Lindenmayer. The algorithmic beauty of plants. Springer Verlag, New York, 1990. 31, 37, 39, 40, 32, 38

[Prusinkiewicz 1998] P. Prusinkiewicz. *Modeling of spatial structure and development of plants: a review.* Scientia Horticulturae, vol. 74, pages 113–149, 1998. 37, 39

[Prusinkiewicz 2001] P. Prusinkiewicz, L. Mündermann, R. Karwowski and B. Lane. *The use of positional information in the modeling of plants.* In ACM SIGGRAPH, editeur, Computer Graphics Proceedings, Annual Conference Series, 2001, pages 289 – 300, Los Angeles, CA, August 2001. 40, 38

[Prusinkiewicz 2004] P. Prusinkiewicz. *Selfsimilarity in plants : integrating mathematical and biological perspectives.* In M. M. Novak, editeur, Thinking in Patterns: Fractals and Related Phenomena in Nature, pages 103–118. World Scientific, Singapore, 2004. 37, 39, 40, 41

[Renton 2006] M. Renton, Y. Guédon, C. Godin and E. Costes. *Similarities and gradients in growth unit branching patterns during ontogeny in 'Fuji' apple trees: a stochastic approach.* J. Exp. Bot., vol. 57, no. 12, pages 3131–3143, 2006. 37, 39

[Rizo 2002] D. Rizo and J.M. Inesta-Quereda. *Tree-Structured Representation of Melodies for Comparison and Retrieval.* In Proceedings of International Workshop on Pattern Recognition in the Information Society, PRIS 2002, page 155, 2002. 63, 58

[Robine 2009] M. Robine, P. Hanna, T. Rocher and P. Ferraro. *Structured Representation of Harmony for Music Retrieval.* In Proceedings of the International Computer Music Conference – ICMC'09, page to appear, Montreal, Canada, 2009. 66

[Room 1994] P.M. Room, L. Maillette and J.S. Hanan. *Module and metamer dynamics and virtual plants.* Advances in Ecological Research, vol. 25, pages 105–157, 1994. 37, 39

[Schenker 1935] H. Schenker. Der frei satz. published in English as Free Composition, translated and edited by E. Oster, Longman, 1979, 1935. 66

[sci 2005] *Science*, Sept. 2005. 45

[Segura 2006] V. Segura, A. Ouangraoua, P. Ferraro and E. Costes. *Comparison of tree architecture using a tree edit distance: application to 2-year-old apple hybrids.* In XIII EUCARPIA Biometrics in Plant Breeding Section Meeting, page 21, 2006. 28, 31

[Segura 2008] V. Segura, A. Ouangraoua, P. Ferraro and E. Costes. *Comparison of tree architecture using tree edit distances: application to two-year-old apple hybrids.* Euphytica, vol. 161, pages 155–164, 2008. 28, 31

[Selkow 1977] S. M. Selkow. *The tree-to-tree editing problem.* Information processing letters, pages 184–186, 1977. 32, 49, 66, 33, 48

[Serrá 2008] J. Serrá, E. Gómez, P. Herrera and X. Serra. *Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification.* IEEE Transactions on Audio, Speech and Language Processing, vol. 16, pages 1138–1151, 2008. 62, 64

[Shapiro 1990] B. A. Shapiro and K. Zhang. *Comparing multiple* RNA *secondary structures using tree comparisons.* Cabios, vol. 6, pages 309–318, 1990. 48, 49, 50

[Siebert 2005] S. Siebert and R. Backofen. *MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons.* Bioinformatics, vol. 21, no. 16, pages 3352–9, 2005. 47

[Sinoquet 1997a] H. Sinoquet and P. Rivet. *Measurement and visualisation of the architecture of an adult tree based on a three-dimensional digitising device.* Trees-Structure and Function, vol. 11, pages 265–270, 1997. 37, 39

[Sinoquet 1997b] H. Sinoquet, P. Rivet and C. Godin. *Assessment of the three-dimensional architecture of walnut trees using digitizing.* Silva Fennica, vol. 3, pages 265–273, 1997. 31, 32

[Smith 1981] T.F. Smith and M.S. Waterman. *Identification of Common Molecular Subsequences.* Journal of Molecular Biology, vol. 147, pages 195–197, 1981. 50, 53, 62, 63, 69, 70

[Sobel 1986] E. Sobel and H. Martinez. *A multiple sequence alignement program.* Nucleic Acid Res., vol. 12, pages 75–88, 1986. 31, 33

[Soler 2003] C. Soler, F. X. Sillion, F. Blaise and P. Reffye. *An efficient instantiation algorithm for simulating radiant energy transfer in plant models.* ACM Transactions on Graphics, vol. 22, no. 2, pages 204–233, 2003. 39, 40

[Stanley 1986] R.P. Stanley. Enumerative combinatorics, volume 1. Cambrigde University Press, 1986. 31, 33

[Sutherland 1963] I. Sutherland. *Sketchpad - a man-machine graphical communication system.* In Proceedings of the Spring Joint Computer Conference, 1963. 38, 40

[Tai 1979] K.-C. Tai. *The tree-to-tree correction problem.* Journal of the Association for Computing Machinery, pages 422–433, 1979. 31, 49, 53, 33, 48, 52

[Tanaka 1986] E. Tanaka, T. Toyama and S. Kawai. *High speed error correction of phoneme sequences.* Pattern Recognition, vol. 19, no. 5, pages 407–412, 1986. 31, 33

[Tarjan 1983] Robert Endre Tarjan. Data structures and network algorithms. CBMS-NFS - Regional Conference Series In Applied Mathematics, 1983. 35, 75, 37

[Tokunaga 1978] E. Tokunaga. *Consideration on the composition of drainage networks and their evolution.* Geogr. Rep. Tokyo Metro. Univ., vol. 13, pages 1–27, 1978. 39, 41

[Toroslu 2007] I. H. Toroslu and G. Üçoluk. *Incremental assignment problem.* Inf. Sci., vol. 177, no. 6, pages 1523–1529, 2007. 78

[Troll 1937] W. Troll. *Die Wuchsform der Bäum und Sträucher.* In W. Troll, editeur, Vergleichunde Morphologie der höheren Pflanzen, volume 1, pages 636–643. Borntraeger, Berlin, 1937. 37, 39

[Troll 1964] W. Troll. Die infloreszenzen, volume 1. Gustav Fisher Verlag, 1964. 40, 38

[Typke 2004] R. Typke, R. C. Veltkamp and F. Wiering. *Searching Notated Polyphonic Music Using Transportation Distances.* In Proceedings of the 12th ACM Multimedia Conference (MM), pages 128–135, New-York, USA, 2004. 62, 75, 64

[Typke 2008] R. Typke and A. Walczak-Typke. *A Tunneling-Vantage Indexing Method for Non-Metrics.* In Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR), pages 351–352, Philadelphia, USA, Sept. 2008. 62, 64

[Uitdenbogerd 1998] Alexandra L. Uitdenbogerd and Justin Zobel. *Manipulation of Music for Melody Matching.* In Proceedings of the Sixth ACM International Conference on Multimedia, pages 235–240. ACM Press, 1998. ACM MM '98 Electronic Proceeding Article. 63

[Uitdenbogerd 2002] A. L. Uitdenbogerd. *Music Information Retrieval Technology.* PhD thesis, RMIT University, Melbourne, Victoria, Australia, July 2002. 62, 63, 65, 58, 60, 64

[Ukkonen 2003] E. Ukkonen, K. Lemstrom and V. Makinen. *Geometric Algorithms for Transposition Invariant Content-Based Music Retrieval.* In Proc. of the 4th International Conference on Music Information Retrieval, pages 193–199, Baltimore, USA, 2003. 62, 64

[Vauchaussade de Chaumont 1985] M. Vauchaussade de Chaumont and X.G. Viennot. *Enumeration of RNAs secondary structures by complexity.* Enumeration of RNAs secondary structures by complexity, vol. 57, pages 360–365, 1985. 47

[Viennot 1989] X. Viennot, G. Eyrolles, N. Janey and D. Arqués. *Combinatorial analysis of ramified patterns and computer imagery of trees.* In SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques, pages 31–40, New York, NY, USA, 1989. ACM. 39, 41

[Wagner 1974] R. A. Wagner and M. J. Fisher. *The string-to-string correction problem.* Journal of the association for computing machinery, vol. 21, pages 168–173, 1974. 31, 49, 67, 68, 70, 33, 48

[Wang 1997] L. Wang and D. Gusfield. *Improved Approximation Algorithms for Tree Alignment.* Journal of Algorithms, vol. 25, no. 2, pages 225–273, 1997. 37, 40

[Wang 1998] J. T.L. Wang, B. A. Shapiro, D. Shasha, K. Zhang and K. M. Currey. *An algorithm for finding the largest approximately common substructures of two trees.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, pages 889–895, 1998. 51, 50

[Wang 1999] J. T.L. Wang, K. Zhang and C. Y. Chang. *Identifying approximately common substructures in trees based on a restricted edit distance.* Information Sciences, vol. 126, pages 367–386, 1999. 38, 51, 40, 50

[Wang 2004] Z. Wang and K. Zhang. *Multiple RNA Structure Alignment.* In IEEE Computational Systems Bioinformatics Conference (CSB'04), pages 246–254, 2004. 47

[White 1979] J. White. *The plant as a metapopulation.* Annual Review of Ecology and Systematics, vol. 10, pages 109–145, 1979. 26, 39

[Zamir 2002] M. Zamir. *On Fractal Properties of Arterial Trees.* Journal of Theoretical Biology, vol. 1997, no. 4, pages 517–526, 2002. 39, 41

[Zhang 1989] K. Zhang and D. Shasha. *Simple fast algorithms for the editing distance between trees and related problems.* SIAM Journal on Computing, vol. 18, no. 6, pages 1245–1262, 1989. 32, 41, 47, 48, 49, 50, 55, 56, 33, 42

[Zhang 1993] K. Zhang. *A new editing based distance between unordered labeled trees.* In Combinatorial Pattern Matching, 4th Annual Symposium CPM 93, Padova (IT), 1993. 31, 32, 33, 38, 34, 40

[Zhang 1994] K. Zhang and T. Jiang. *Some MAX SNP-hard results concerning unordered labeled trees.* Information Processing Letters, vol. 49, pages 249–254, 1994. 34, 37

[Zhang 1996] K. Zhang. *A constrained edit distance between unordered labeled trees.* Algorithmica, vol. 15, no. 3, pages 205–222, 1996. 31, 32, 34, 35, 38, 41, 33, 37, 40

Analysis of Tree Structures : From Computational Biology to Music Analysis

Abstract: This document present a synthetic point of view of my research for the last ten years.

Sequence and tree structure data analysis is a standard operation for which there are well-known utilities. In particular, trees are widely used structures for coding data in biology. A huge number of projects have been proposed to develop method for analysing these biological tree structured data. During the last ten years, I have proposed to adapt these different tree-structured data analysis methods to different contexts from plant architecture modeling to music analysis and I have then developped new approaches to analyse tree data structures.

The increasingly number of databases describing structured biological data (RNA secondary structures, gene sequences, plant architecture, *etc.*) or musical pieces generates a need for new investigational tools.

Recent development in computer science and applied mathematics has allowed me to define new methods integrating simultaneously structure and dynamic of development.

Keywords: Plant Architecture Analysis, RNA Secondary Structures Comparison, Symbolic Music Analysis, Tree Data Structures, DAGs, Multi-Scale Tree-Graphs