Universidad del Atlántico

¡La mejor educación al alcance de todos!

Ejemplo de una tesis de Maestría en la Universidad del AtlánticoSOBRE UNA NUEVA CLASE DE POLINOMIOS TIPO APOSTOL GENERALIZADOS Y ALGUNAS PROPIEDADES

Nombres Apellidos

Tesis De Maestría

Barranquilla, Marzo 2014

Ejemplo de una tesis de Maestría en la Universidad del AtlánticoSOBRE UNA NUEVA CLASE DE POLINOMIOS TIPO APOSTOL GENERALIZADOS Y ALGUNAS PROPIEDADES

Nombre Apellido

Trabajo de Maestría

Barranquilla, Marzo 2014

Ejemplo de una tesis de Maestría en la Universidad del Atlántico

Ejemplo de Tesis

Trabajo de grado presentado para optar al grado Maestría en Matemática
para la Universidad del Atlántico (UA)

Este trabajo de investigación ha sido realizada bajo la dirección de
Dr. Director †

† Departamento de Matemática,
Universidad del Atlántico (UA)

Trabajo de investigación en Matemática,
Universidad del Atlántico (UA)

Dr. advisors:
DR. †

† Departamento de Matemática
Universidad del Atlántico (UA)

Barranquilla, Marzo 2014

UNIVERSIDAD DEL ATLÁNTICO
PROGRAMA DE MATEMÁTICAS
MAESTRÍA EN MATEMÁTICAS

SOBRE UNA NUEVA CLASE DE POLINOMIOS TIPO APOSTOL GENERALIZADOS Y ALGUNAS PROPIEDADES

Estudiante: Pedro Hernández Llanos
Carnet: 1048206189

Este Trabajo Especial de Grado ha sido aprobado en nombre de la Universidad del Atlántico por el siguiente jurado examinador:

Dr. Boris Lora Castro
Universidad del Atlántico - Colombia

Dra. Yamileth Quintana Mato
Universidad Simón Bolivar - Venezuela

Alejandro Urieles Guerrero.
Tutor

07 de Marzo de 2014

AGRADECIMIENTOS

Doy las gracias a Dios que es el principal responsable de que finalizara de manera exitosa este trabajo. Es muy difícil expresar con palabras la inmensa gratitud que le debo al profesor Alejandro Urieles Guerrero, ya que durante estos casi tres años de mi permanencia en la Universidad del Atlántico, además de mostrarme el camino de la investigación en los polinomios ortogonales y tenerme una paciencia infinita, también me ayudó a a encontrar el camino a la investigación de los polinomios tipo Apostol. El llegar a un feliz término en esta meta que me fijé hace mucho tiempo, es, en un gran porcentaje, gracias a él. Mil gracias Profesor Urieles.

También debo agradecer al departamento de Matemáticas de la Universidad del Atlántico, su apoyo no sólo económico, mediante la beca de la que he gozado durante los últimos años, sino la acogida que me brindaron cada uno de los integrantes de este agradable departamento. Particularmente, quiero resaltar el apoyo que me brindó el profesor Jorge Rodriguez Contreras, quien, como director de departamento, colaboró para que pudiera realizar muchas de las actividades académicas a las que tuve la suerte de asistir y quien además en los inicios de este proyecto, me proporcionó ayuda y consejo.

Quiero resaltar el sacrificio y dedicación de mis padres, lo que me ha permitido ser lo que soy. Finalmente, quiero decirles a Diana Patricia, Adriana y Diana Lucía que este proyecto no habría sido posible sin su apoyo y compañía, y que este logro tambin es de ustedes.

Pedro Hernández

Introducción

As a famous dirty detective once said, there must be a hundred good reasons why I shouldn't have just initiated a PhD thesis. But right now, I can't think of a single one. On the contrary, I wonder who would have rejected the appealing proposal to investigate the genomic world, which is actually the center of the life, designing programs on a high-performance computational environment.

The construction of the first modern computers was one of the major landmarks achieved by the human being in the past century. Since then, the application of computers on many intriguing problems and the constant evolution of the programs that govern them have permitted the researchers in many areas to discover new concepts that would have been otherwise unreachable for our generation without this technology.

Molecular biology is not an exception. The sequencing of the human genome would be still an impossible challenge if many automatic procedures that are now familiar to us would have not been developed before. In this context, Bioinformatics has been the relevant driving force responsible for stimulating the advance in the study of the biology of our cells. Particularly, many clues to understand the life in our planet can be found in the regulation of gene expression. Nonetheless, to be sincere I have to admit that we are still completely ignorant: a huge amount of new biological information is constantly released so that the global picture that we want to reconstruct becomes today somehow even more complex than the day before.

Understanding life is an enormous challenge. In other scale, a PhD is also an exciting challenge for a student. It is a period in which not only such a person acquires a valuable education in many aspects of his life. At the same time, this individual is supposed to be capable of applying such knowledge in the investigation of a real problem, sometimes in competition with other people that have much more experience. In my case, the task became even more complex as a computer scientist needs a solid biological background to approach this kind of problems.

This thesis not only pretends to communicate the different phases of my work during the PhD period of research. Before starting to write, it was also my commitment to elaborate a manuscript fulfilling the highest requirements of quality and accuracy in the the material that is presented. This manuscript attempts to follow a logical and continuous argument from the introductory parts to the specific chapters devoted to the presentation of the results of the thesis. In addition, a DVD with supplementary materials such as the electronic thesis, the bibliography, the software or several educational resources, is also released as an excellent complement to the thesis.

The experience and the abilities I have personally acquired during this period do not fit in just two hundred pages. From my point of view, the most relevant result of a thesis is not the compilation of scientific papers published during that time (these should be seen as a relevant consequence of a good work). On the contrary, I am totally convinced that the essential result of a PhD thesis is the improvement of the individual

that positively changes his life in many aspects, producing an amazing enrichment of his personality.

In our childhood, many of us have got an intimate and naive desire of changing the world to improve it. Surprisingly after so many years, I still have this feeling although I am quite conscious that some things are not so easy to be changed whereas others simply can not be changed. However, I am happy to see that I have acquired a solid education that will be very useful to face more complicate situations throughout my life. In fact, this PhD thesis has not represented for me a central objective but an excellent opportunity to stop and learn, driving me to more ambitious challenges.

The education of our society has been always among my priorities. To be able to teach is necessary to learn to teach before. This is reflected in the fact that I have voluntarily performed hundreds of teaching activities during my thesis, always with a high degree of motivation in my presentations. Throughout our lives we do not cease to gain new knowledge. But we investigators have the duty of communicating rigorously our achievements with honesty to our people at schools, institutes, universities, meetings and mass media. To reach this ambitious objective is necessary to be engaged and involved in such a project. If we fail now in this attempt, I suspect that the gap between those that have the power to learn and investigate and those that do not, will be dangerously large, probably too much.

Barcelona, Marzo 2014

Contents

List of Tables

List of Figures

Agradecimientos

THIS SECTION IS USUALLY THE PART OF THE THESIS in which the authors mention those people that have decisively contributed to the presented work. As I am a generous person but my gratitude is not infinite, I want to express the following acknowledgments only to those that really deserve the reward of being cited here.

I am totally convinced about how to begin and to end this section. Honestly, there is only one person that deserves the honor of appearing in the first place of this section: myself. This thesis has not been an easy work at all. In our society, most computer scientists are working on the private sector, so that the orientation of their careers to investigation is a rare fact nowadays. And I have learned to live with this pressure as well. For a computer engineer like me, it has been a rich experience to work in a research environment devoted to the biological discovery. However, it has also been very demanding, because this thesis is not only about the development of new theoretical algorithms. It was also an exercise of application of such methods in real data to obtain novel biological conclusions. To sum up, it was like doing two thesis: one about computer science, and one about molecular biology. And I am very proud to have fulfilled both aspects of my work. Therefore, I want to thank myself for not abandoning, for supporting myself, for carrying on when the main objectives of the thesis seemed to be very far, when things were going too slow, or when the adaptation to the academic world was difficult because of its competitiveness.

I want to warmly thank you my two PhD advisors, Xavier Messeguer and Roderic Guigó for the correct direction of my work. We started in 1999 with the program `geneid` to successfully obtain my degree in computer science the summer of 2000. A few years later, I am very happy to see that the majority of people in my lab have used it in their investigations with a lot of success, and some of them have even been able to modify some of its modules without difficulty. Thanks then to Xavier, for your calm, for your wisdom, for your patience with me, specially when continuous communication was sometimes difficult because I was physically working outside the department, and for believing in my work in many occasions that I will never forget. Also thanks to Roderic, for the computational facilities, for the opportunity to work with so good people in the IMIM lab, for let me learning involuntarily from your experience, for the funding, for the international meetings that increased a lot my knowledge and for being always ambitious in whatever task you are doing.

Also thanks to my colleagues at work from which I learn a lot of useful things. Many of them have also decisively contributed to improve the quality of this thesis. Specially thanks to Josep Francesc Abril that have assisted me in uncountable occasions with his priceless help. Thanks also to those that were in the lab when I arrive over there: Moisés Burset, Sergi Castellano and Genís Parra. To those that arrived later, many thanks as well: Robert Castelo, Jan-Jaap Wesselink, Mar Albà, Eduardo Eyras, Charles Chapple, Nicolás Bellora and

Miguel Pignatelli. Further thanks to our system administrators, Alfons González, Xavier Fustero and Òscar González. I want to specially acknowledge Robert Castelo for the excellent template in LaTeX from his PhD thesis. This template was later adapted by Sergi Castellano and Genís Parra, and substantially improved by Josep Francesc Abril, for their theses. This manuscript is an adaptation of such templates following my own style.

At this point I want to remember those teachers from many disciplines that have contributed positively to my education throughout my life. First of all, thanks to those in the teaching staff that positively contributed to my education at my school Hermanos Maristas de Les Corts. Second, to those good teachers I have found in my university Facultad d'Informàtica de Barcelona. Finally, many thanks to my teachers at Escola Oficial d'Idiomes de Barcelona, that help me to speak and to write correctly in English and Italian.

During this time, I have been involved in many educational activities related to teach about Bioinformatics in Masters and other programs. Specially thanks among others for your cooperation and your advice to Manuel Gómez (Centro Nacional de Astrobiología, Madrid), Silvia Atriain (Universitat de Barcelona, Barcelona) and Alfonso Valencia (Centro Nacional de Biotecnología, Madrid).

Many thanks also to Dr. Montserrat Corominas and Dr. Jorge Ferrer for two fruitful and interesting collaborations, using the expression data produced during the research performed in their labs.

For formal reasons, I have to thank the Ministerio de Educación y Ciencia of Spain and the Institut Municipal d'Investigacions Mediques (IMIM) for the funding for my thesis. Also thanks to Cold Spring Harbor Labs for several travel grants to attend their excellent meetings.

Specially during the latest stages of my thesis I have not much time for my friends so that it is now a good moment to thank you for being there. Specially thanks to David Sánchez for your friendship and for your help, and to David Valldosera for your proximity and wise advice. Also thanks to Josep Vallverdú, Roberto García and Oriol Teixidó for conserving our friendship since we first met at university.

As I said before, it was very clear to me how to begin and end this section. Now that we have arrived at the end, I would like to express my acknowledgments to those that deserve the honor of closing this section: my parents. What I have reached in my life is due to your courage and decision. You can be sure that I will never forget my roots. Thanks to both, for being always my support. Time goes by in my life but you are always here with me. This work is entirely dedicated to you.

Resumen

The sequences are very versatile data structures. In a straightforward manner, a sequence of symbols can store any type of information. Systematic analysis of sequences is a very rich area of algorithmics, with lots of successful applications. The comparison by sequence alignment is a very powerful analysis tool. Dynamic programming is one of the most popular and efficient approaches to align two sequences. However, despite their utility, alignments are not always the best option for characterizing the function of two sequences. Sequences often encode information in different levels of organization (meta-information). In these cases, direct sequence comparison is not able to unveil those higher-order structures that can actually explain the relationship between the sequences.

We have contributed with the work presented here to improve the way in which two sequences can be compared, developing a new family of algorithms that align high level information encoded in biological sequences (meta-alignment). Initially, we have redesigned an existent algorithm, based in dynamic programming, to align two sequences of meta-information, introducing later several improvements for a better performance. Next, we have developed a multiple meta-alignment algorithm, by combining the general algorithm with the progressive schema. In addition, we have studied the properties of the resulting meta-alignments, modifying the algorithm to identify non-collinear or permuted configurations.

Molecular life is a great example of the sequence versatility. Comparative genomics provide the identification of numerous biologically functional elements. The nucleotide sequence of many genes, for example, is relatively well conserved between different species. In contrast, the sequences that regulate the gene expression are shorter and weaker. Thus, the simultaneous activation of a set of genes only can be explained in terms of conservation between configurations of higher-order regulatory elements, that can not be detected at the sequence level. We, therefore, have trained our meta-alignment programs in several datasets of regulatory regions collected from the literature. Then, we have tested the accuracy of our approximation to successfully characterize the promoter regions of human genes and their orthologs in other species.

Resumen

Las secuencias son una de las estructuras de datos más versátiles que existen. De forma relativamente sencilla, en una secuencia de símbolos se puede almacenar información de cualquier tipo. El análisis sistemático de secuencias es un área muy rica de la algorítmica, con numerosas aproximaciones llevadas a cabo con éxito. En concreto, la comparación de secuencias mediante el alineamiento de éstas es una herramienta muy potente. Una de las aproximaciones más populares y eficientes para alinear dos secuencias es el uso de la programación dinámica. Sin embargo, a pesar de su evidente utilidad, un alineamiento de dos secuencias no es siempre la mejor opción para caracterizar su función. Muchas veces, las secuencias codifican la información en diferentes niveles (meta-información). Es entonces cuando la comparación directa entre dos secuencias no es capaz de revelar aquellas estructuras de orden superior que podrían explicar la relación establecida entre éstas.

Con este trabajo hemos contribuído a mejorar el modo en el que dos secuencias pueden ser comparadas, desarrollando una familia de algoritmos de alineamiento de la información de alto nivel codificada en secuencias biológicas (meta-alineamientos). Inicialmente, hemos rediseñado un antiguo algoritmo, basado en programación dinámica, capaz de alinear dos secuencias de meta-información, procediendo despues a introducir varias mejoras para acelerar su velocidad. A continuación hemos desarrollado un algoritmo de meta-aliniamento capaz de alinear un número múltiple de secuencias, combinando el algoritmo general con un esquema de clustering jerárquico. Además, hemos estudiado las propiedades de los meta-alineamentos producidos, modificando el algoritmo para identificar alineamientos con una configuración no necesariamente colineal, lo que permite entonces la detección de permutaciones en los resultados.

La vida molecular es un ejemplo paradigmático de la versatilidad de las secuencias. Las comparaciones entre genomas, ahora que su secuencia está disponible, permiten identificar numerosos elementos biológicamente funcionales. La secuencia de nucleótidos de muchos genes, por ejemplo, se encuentra aceptablemente conservada entre diferentes especies. En cambio, las secuencias que regulan la expresión de los propios genes son más cortas y variables. Así que la activación simultanea de un conjunto de genes se puede explicar sólo a partir de la conservación de configuraciones comunes de elementos reguladores de alto nivel, y no a partir de la simple conservación de sus secuencias. Por tanto, hemos entrenado nuestros programas de meta-alineamiento en una serie de conjuntos de regiones reguladoras recopiladas por nosotros mismos de la literatura y despues, hemos probado la utilidad biológica de nuestra aproximación, caracterizando automáticamente con éxito las regiones activadoras de genes humanos conservados en otras especies.

Resumen

Les seqüències són una de les estructures de dades més versàtils que existeixen. De forma relativament senzilla, en una seqüència de símbols es pot emmagatzemar informació de qualsevol tipus. L' anàlisi sistemàtic de seqüències es un àrea molt rica de l'algorísmica amb numeroses aproximacions desenvolupades amb éxit. Particularment, la comparació de seqüències mitjançant l'alineament d'aquestes és una de les eines més potents. Una de les aproximacions més populars i eficients per alinear dues seqüències es l'ús de la programació dinàmica. Malgrat la seva evident utilitat, un alineament de dues seqüències no és sempre la millor opció per a caracteritzar la seva funció. Moltes vegades, les seqüències codifiquen la informació en diferents nivells (meta-informació). És llavors quan la comparació directa entre dues seqüències no es capaç de revelar aquelles estructures d'ordre superior que podrien explicar la relació establerta entre aquestes seqüències.

Amb aquest treball hem contribuït a millorar la forma en que dues seqüències poden ser comparades, desenvolupant una família d'algorismes d'alineament de la informació d'alt nivell codificada en seqüències biològiques (meta-alineaments). Inicialment, hem redissenyat un antic algorisme, basat en programació dinàmica, que és capaç d'alinear dues seqüencies de meta-informació, procedint després a introduir-hi vàries millores per accelerar la seva velocitat. A continuació hem desenvolupat un algorisme de meta-aliniament capaç d'alinear un número múltiple de seqüències, combinant l'algorisme general amb un esquema de clustering jeràrquic. A més, hem estudiat les propietats dels meta-alineaments produïts, modificant l'algorisme per tal d'identificar alineaments amb una configuració no necessàriament col.lineal, el que permet llavors la detecció de permutacions en els resultats.

La vida mol.lecular és un exemple paradigmàtic de la versatilitat de les seqüències. Les comparacions de genomes, ara que la seva seqüència està disponible, permeten identificar numerosos elements biològicament funcionals. La seqüència de nucleòtids de molts gens, per exemple, es troba acceptablement conservada entre diferents espècies. En canvi, les seqüències que regulen l'expressió dels propis gens son més curtes i variables. Així l'activació simultànea d'un conjunt de gens es pot explicar només a partir de la conservació de configuracions comunes d'elements reguladors d'alt nivell, i no pas a partir de la simple conservació de les seves seqüències. Per tant, hem entrenat els nostres programes de meta-alineament en una sèrie de conjunts de regions reguladores recopilades per nosaltres mateixos de la literatura i després, hem provat la utilitat biològica de la nostra aproximació, caracteritzant automàticament de forma exitosa les regions activadores de gens humans conservats en altres espècies.

PART I

Preliminares

Chapter 1

Introduction

Summary

This chapter details the general questions of the document. It provides a brief explanation of the motivation for this work. Then, the list of objectives of the thesis is introduced. The completion of these tasks and the final calendar of execution of the project (year by year) is included as well. The manuscript is logically divided into three different parts: Preliminaries, State of the Art and Meta-alignments. There is also a brief description of the chapters of each part. Finally, some particular considerations about how to read the book and the layout of the document are presented.

1.1 General objectives

THE PRINCIPAL OBJECTIVE OF THE THESIS DISSERTATION is to explain in detail the topic on which the work of several years has been focused. In addition, the experience of the author at different areas has been reflected here in the numerous descriptions and solutions to several biological and computational problems. Speculation about future research and criticism have been a valuable ingredient as well.

This is a thesis about computational sequence analysis, particularly applied to characterize genomic sequences. The way in which this synergy between a biological problem and a computational solution is expressed was considered to be crucial for the success of this document. The generality of the proposed solutions, which can be applied to any type of sequence (biological or not), is also underlined in the corresponding sections.

The core of the thesis is the development of a new family of algorithms to align transcription regulatory regions. Among them, a global pairwise algorithm and a global progressive multiple algorithm have been shown to be useful in the characterization of a gene promoter region, specially when the amount of predictions by other systems is excessive. Sketches of other versions are also provided (parallel, local).

The work performed about the meta-alignment strategy has been interestingly complemented and enriched with a serious approach to the algorithms that originated the concept of sequence analysis several decades ago. Such a chapter is an interesting opportunity to review for the first time some of the classic papers in the field that are still very relevant, in spite of the deluge of new proposals and publications continuously released. The introduction of this material in the document improves without any doubt the quality of the final manuscript.

In addition, several references about the relationship between current advances in genomics and society can be found in the text. In my opinion, ethics must be part of any human achievement. Genomics and other 'Omics' disciplines promise to radically change our way of life. Medicine, biotech farming, crime investigation and personal privacy among others will be severely affected.

To sum up, this thesis aims to become an educational book reference. This is an excellent opportunity to explain in detail the topic of the meta-alignment but also to construct an exciting portrait of sequence analysis in computational biology. To satisfy all of these requirements, the use of current technologies to produce an outstanding work was also mandatory. Thus, a DVD with additional materials (electronic thesis, relevant bibliography, source code, educational material, ...) supporting the main text is a good complement to the PhD dissertation.

1.2 Objectives

The characterization of gene regulatory regions is a fundamental step toward understanding the great existing variability between different species. However, it is still an open problem due to the peculiar features of the regulatory elements. The research in this PhD thesis has been oriented to the development of new computational methods of alignment to deal with such information. However, it is important to mention that the algorithms presented here can deal with other problems that show a similar theoretical framework, lacking of a biological background.

In short, the following objectives were established in 2001 for this thesis:

① To study the biological problem of gene regulation in eukaryotes. This includes the control of gene expression, specially through the transcription of the genes: promoters, transcription factors, DNA-protein binding sites, chromatin effect, CpG islands.

② To analyze the current computational methods to search regulatory elements in a promoter region. This includes the algorithms based on pattern matching using catalogues of regulatory elements and the pattern discovery algorithms that extract useful information from a set of related sequences.

③ To investigate the more recent comparative approaches based on phylogenetic footprinting and microarrays. To understand the biological concepts behind the gene orthology. To study the biological and technological concepts of the high-throughput expression experiments.

④ To analyze the existent sequence pairwise sequence alignment algorithms. To study the concept of map, the mapping functions and the map alignment problem.

⑤ To design novel algorithms to align two regulatory sequences that produce a minor amount of false positives. To present real biological scenarios in which these approaches show to be more efficient than the conventional sequence alignment algorithms.

⑥ To compile and to maintain a public dataset of regulatory annotations suitable for training these and other algorithms that deal with data from comparative genomics and microarray experiments.

⑦ To study several alternatives to extend the basic pairwise approach developed before to align multiple sequences. Test this approach on orthologous datasets and microarray expression data.

⑧ Public distribution of the software and the databases produced during this thesis to the scientific community. To write web servers that implement most of the methods presented above.

1.3 Thesis chronology

This is a short enumeration of the main tasks implemented during the PhD thesis and their associated results, year by year:

- 2001
  - ① Planning: decide the main lines and the objectives of the thesis.
  - ② Biological problem: bibliographical research in general molecular biology books about the eukaryotic transcription and other forms of gene regulation.
  - ③ State of the art: bibliographical research in published papers about the classical algorithms and strategies to analyze gene promoter regions. Including the study of the advanced techniques such phylogenetic footprinting and microarray experiments.
  - ④ Attended conferences: Intelligent Systems in Molecular Biology (ISMB) at Copenhaguen, Denmark.
- 2002
  - ① Analysis of co-expressed genes in *Drosophila melanogaster*: gene characterization, G+C content, clustering, gene function, promoter characterization including phylogenetic analysis.

② Analysis of co-expressed genes in *Mus musculus*: the results of several microarrays were analyzed with the existing computational tools, including phylogenetic footprinting.

➢ 2003

① Developing the global and local meta-alignment first prototypes.

② Bibliographical research to find regulatory data for training the meta-alignment approach.

③ Attended conferences: Research in Computational Biology (RECOMB) at Berlin, Germany.

➢ 2004

① Tuning the meta-alignment. Improving the efficiency of the basic implementation with lists.

② Writing the web server of the pairwise meta-alignment program.

③ Training the meta-alignment on a small dataset of annotated promoters.

④ First prototypes for multiple meta-alignment.

⑤ Attended conferences: Systems Biology at Cold Spring Harbor Labs, New York, USA.

➢ 2005

① Creation of a public database of annotated promoters (ABS).

② Final tests: pairwise meta-alignment approach on the CISRED database.

③ Evaluation of the quality of weight matrices using the meta-alignment.

④ Tuning the multiple meta-alignment. Improving the computational efficiency.

⑤ Variations to allow the existence of non-colinear alignments in the results.

⑥ Starting to write the thesis dissertation.

⑦ Attended conferences: Systems Biology at Cold Spring Harbor Labs, New York, USA.

➢ 2006

① Final training of the multiple meta-alignment on a set of orthologous of multiple species.

② Finishing the thesis dissertation.

③ Public defense of the PhD thesis.

1.4 Outline of this thesis

This thesis has been written following the format of a text book. The main text is divided into three parts: introduction, state of the art and results. Every part consists of a set of chapters, each one devoted to a given topic. Chapters can be read separately to facilitate the accession to individual parts of the book, but the thesis has been written following a linear and continuous logical script.

This is a brief description of the content of each chapter:

① Introduction: general motivation of the thesis containing the objectives, the calendar and other considerations about the project and the format of the book.

② The post-genomic era: biological description of genomic concepts (genes, DNA, mRNA), the genome sequencing projects, bioinformatics, future implications of the genomic research in medicine.

③ The golden age of sequence analysis: a comprehensive historical review of the pioneering algorithms in sequence and map alignment in the seventies and eighties, including a detailed analysis of the most relevant ones.

④ Computational gene and promoter characterization: a survey of the state of the art in the analysis of genomic sequences (genes and regulatory regions), and a study of the different techniques implemented such as the representation of signals, the detection of biased content regions or the homology search.

⑤ Pairwise meta-alignment of regulatory sequences: the mapping functions, the TF-map approach, basic implementations, the accurate construction of collections of examples, the training, the application on a database of co-regulated genes, the detection of promoter regions, the use of meta-alignment to evaluate the specificity of matrices. Other versions: local and parallel meta-alignment.

⑥ Multiple meta-alignment of regulatory sequences: the progressive approach, the design of the final solution, the modification to produce non-colinear alignments, the tests on orthologous promoters from multiple species.

⑦ Conclusions: the enumeration of the results of this thesis.

⑧ Appendix section: curriculum vitae, software and web servers, publications, posters, web glossary.

1.5 Particular considerations

The following are some individual considerations about the thesis:

- The electronic version of this document has hyper links for the table of contents, for the bibliographic references, but most important of all, also for the web addresses on the Internet—from now on, their Uniform Resource Locator (URL). This means that you can visit the corresponding web page by clicking your pointer on them, in case that you have your PDF viewer properly customized. Many of the URLs presented in this book have been collected in a web links reference index available on page **??**. URLs within paragraphs have been moved into that web glossary in order to avoid unbalanced line breaks and for a more pleasant reading. A reference to the corresponding page in the web reference index is provided instead.
- An attempt has been made to keep software names as provided by their authors. Those names appear in a `monospaced serif font`. Database names are typeset in a SMALLCAPS SANS-SERIF FONT. A *emphasized font* was used for gene names.
- The first time an acronym appears in the document, the full name will be provided and the acronym itself will be shown in parentheses.
- The publications and submissions of papers in which the author of this thesis was involved are included at the end of the thesis as an appendix.
- The use of colour is considered to be essential to accurately highlight some contents of the thesis such as the equations, the algorithms or the figures and the tables.

➳ The author of this thesis has carefully selected the bibliography of each chapter. Following such references, a detailed reconstruction of such a topic can be performed with great accuracy. Some of these references are also included as electronic supplementary material in the DVD companion to this thesis.

Chapter 2

The post-genomic era

Summary

This chapter is a basic survey of the molecular and cell biological concepts that will be used throughout this thesis, with special emphasis on the topics of genetics and genomics. In addition, the relatively new discipline of bioinformatics is examined, focusing on the genomic databases and the integration of data from different biological domains. The dramatic changes that medicine and drug design are going to experience after the sequencing of the human genome project are explored at the end of the chapter.

2.1 The genomic landscape

The universe of the cells

THE CELL, a small membrane-bounded compartment filled with a concentrated aqueous solution of chemicals, is the essential constituent of life. Bacteria, plants, birds or humans, all living organisms on Earth are made of at least one cell. Because of their apparent simplicity and flexibility, cells have been able to achieve an incredible success in their perpetuation efforts (**?**).

All living beings and the cells that form them are believed to have descended from a common ancestor cell through evolution by natural selection. This process involves two simple steps: (1) random variation in the genetic information passed from an individual to its descendants and (2) selection of the genetic information that permits its possessors to survive and propagate in their environment.

Evolution began billions of years ago in our planet. Simple organic molecules (molecules containing carbon) such as amino acids and nucleotides are likely to have been produced under primitive conditions on Earth. Later, these molecules associated to form polymers or more complex structures such as proteins and nucleic acids (DNA and RNA). The competition between such primitive structures for the available precursor materials in that unstable environment produced many of the biological processes present in many cells now. The interplay between DNA and RNA in the protein synthesis pathway is the best example of this. At present, DNA acts as the permanent repository of genetic information in most cells while RNA, originally the molecule from which rudimentary peptides were produced, remains as an intermediary between DNA and proteins (**?**).

The isolation from the external medium was one of the crucial events leading to the formation of the first cell. The development of an outer membrane by phospholipids around some of these primitive structures provided a brand new capability: the protection of the information that could contribute selectively in the competition against other similar systems (e.g. hereditary material such as a variant RNA that made a superior type of enzyme).

These primitive cells that have survived successfully until our days are the bacteria (also known as prokaryotes). The structure of a bacteria is a simple cell wall beneath which a plasma membrane encloses a single cytoplasmatic compartment containing the genetic material, proteins and small molecules. Basically, survival in bacterial terms means to achieve the fastest speed of replication or cell division to incorporate as many genetic changes as possible on their DNA through each generation. Genetic variability facilitates a rapid adaptation of the species to a changing environment.

The action of millions of these organisms slowly caused revolutionary changes on Earth. The atmosphere was transformed through cyanobacterial photosynthesis or respiration from a mixture with practically no oxygen to one in which oxygen constitutes 21% of the total (**?**). This dramatic change in the environment produced the extinction of many types of cells but also induced the symbiosis between ancient cells adapted to the prebiotic environment without oxygen (anaerobic) with those possessing the ability to process the oxygen (aerobic).

This transition to more structured cells named eukaryotes implied numerous additional changes in response to the new situation: bigger size, a rich array of internal membranes to facilitate the transport of the materials for biosynthetic reactions occurring inside the cell and finally, a new inner membrane to protect the increasing genetic material. The stability of the DNA double helix made the storage of higher quantities

Figure 2.1 Electron micrograph of a chicken chondrocyte. Chondrocytes are cells from the cartilage (connective tissue). Adapted from UBC BIOMEDIA IMAGE AND MOVIE DATABASE (see Web Glossary, page **??**).

of genetic information easier. Additional packaging mechanisms were required to manipulate the growing hereditary material inside this second membrane, also known as nuclear membrane (**?**).

The next step in evolution was the appearance of multicellular organisms. By collaboration and division of tasks, the efficient exploitation of resources that no single cell could utilize before was now possible. Multicellularity enables an individual to separately specialize groups of its cells to perform absolutely different tasks in a collaborative manner. An electron micrograph of an eukaryotic cell from connective tissue is shown in Figure 2.1. All of the cells of every multicellular organism have the same genetic material and are generated by repeated division from a single precursor cell. But, surprisingly, despite having an identical genetic composition when they grow, they become differentiated from others, adopting a different structure and different functions (**?**).

The mechanisms that governed this amazing ability for specialization are intimately related with the management of the basic units that form the genetic information of a cell: the genes.

Genes and inheritance

The basic component of deoxyribonucleic acid or DNA is the nucleotide, defined by its chemical base: Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). The DNA that constitutes the genetic material of

cells is a double-stranded molecule consisting of two chains of nucleotides running in opposite directions. The A-T and G-C base pairs are complementary because these bases form hydrogen bonds that keep them together. Thus, each strand of the molecule is a template to make a copy of the other sequence of bases.

The genes are the basic physical and functional units of heredity. Genes are fragments of DNA with a specific sequence of bases that encodes instructions on how to control a discrete hereditary characteristic. The set of genes belonging to an individual is the genotype.The phenotype is the set of traits expressed in an individual with a certain genotype. A polymorphic gene is a gene in which small variations in its sequence from two different individuals produce different observable physical traits. Each one of the set of alternative forms of a gene is an allele or variant[1].

In sexually reproducing organisms, such as humans, each gene in an individual is represented by two copies or alleles, one from each parent. A dominant allele is an allele that is almost always expressed, even if only one copy is present, overshadowing the other. Known examples of dominant alleles are Huntington's disease and polydactylism (extra fingers and toes). On the contrary, a recessive phenotype will only be expressed if both copies contain the recessive allele. When a recessive allele is overshadowed by a dominant allele and the recessive trait is not expressed, the individual is said to be a carrier for that trait. Recessive disorders in humans include sickle cell anemia and Tay-Sachs disease (NCBI report: genomics, see Web Glossary, page **??**).

There are exceptions to these basic laws, usually complex interactions among various allelic conditions:

- Co-dominant alleles both contribute to a phenotype, for example in the case of human blood group.
- Pleotrophy is the phenomenon in which a single gene is responsible for producing multiple and apparently distinct traits.
- A gene that masks the phenotype of another gene is an epistatic gene while the subordinated gene is the hypostatic gene such as in the case of the albinism gene.
- There are traits that are multigenic because they result from the expression of several different genes such as the three genes at least that determine eye colour.

The cell cycle is the process that a cell follows to replicate. To produce a copy of the original cell having an identical genetic composition, the hereditary material is duplicated. Errors are not unusual to happen during the copy. Moreover, dramatical changes in the environment such as exposure to ultraviolet radiation or toxic chemicals can promote changes in the DNA as well. Genetic variations are usually the result of mutations in the sequence of a functional element: substitutions, deletions or insertions of nucleotides. Mutations that occur in germ cells will be passed on to the next generation while those changes in ordinary cells will only affect the individual.

Although most defective cells die quickly, some can persist and may even become cancerous if the mutation affects cell growth control. However, not all mutations are negative. The main effect of mutations is the opportunity to adapt to a new environment by following the rules of the natural selection: most mutations do not produce any observable result in an organism, others are terribly pernicious causing severe damage, and a minority of them substantially improve the probability of success in the propagation of its genes (**?**).

[1] See **?** for a comprehensive historical review of genetics.

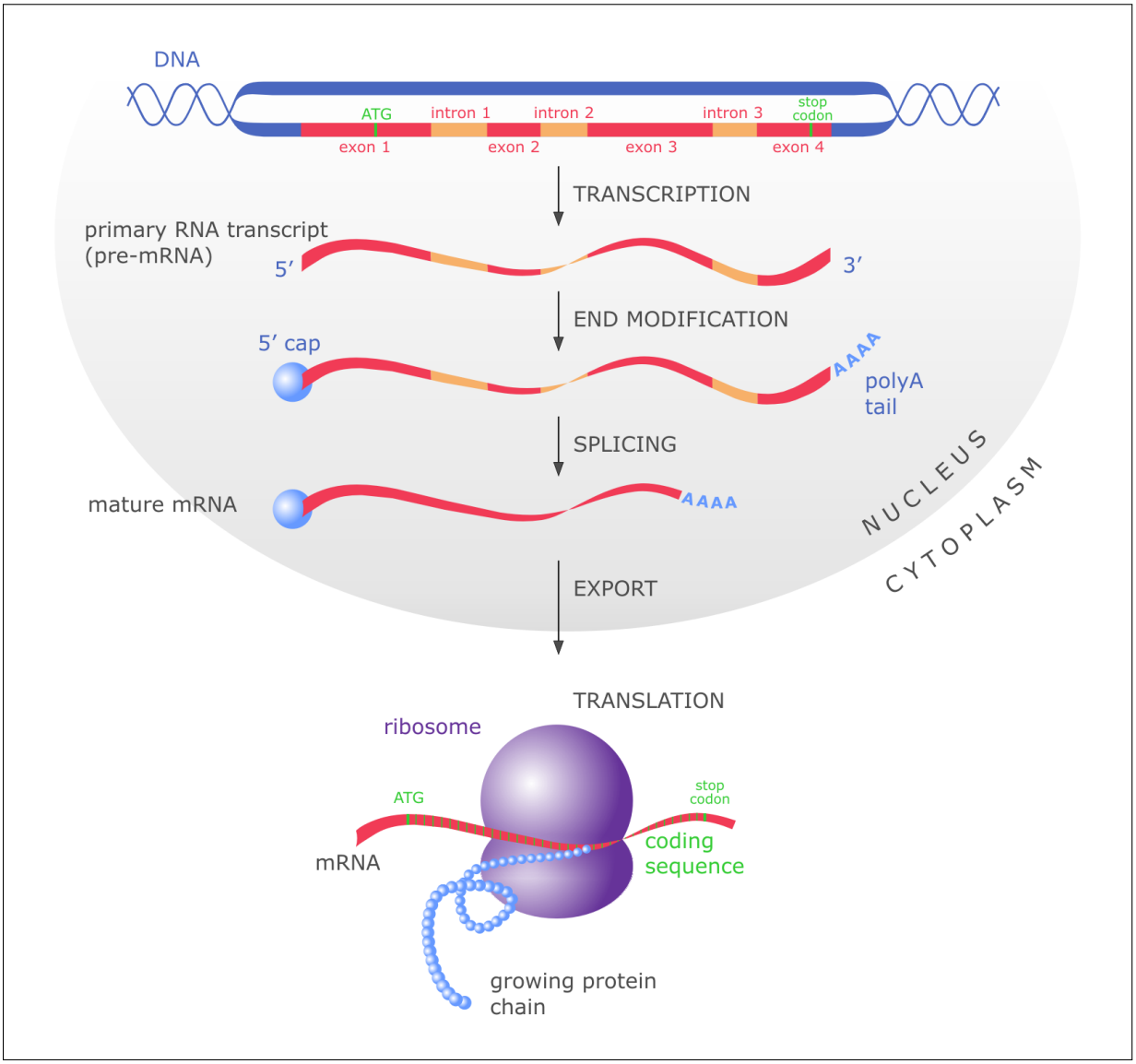


Figure 2.2 The molecular processes involved in the pathway leading from DNA to protein. See main text for further details. Adapted from **?**.

Genes and proteins

Ribonucleic acid or RNA molecules are single-stranded chains of nucleotides that are constructed using one of the two DNA strands of a given gene as a template, with the substitution of Thymine (T) for Uracil (U). Each gene produces a functional RNA molecule (**?**). Transcription from DNA to RNA is the first step in the protein synthesis pathway, schematically represented in Figure 2.2. Each RNA molecule can encode a protein

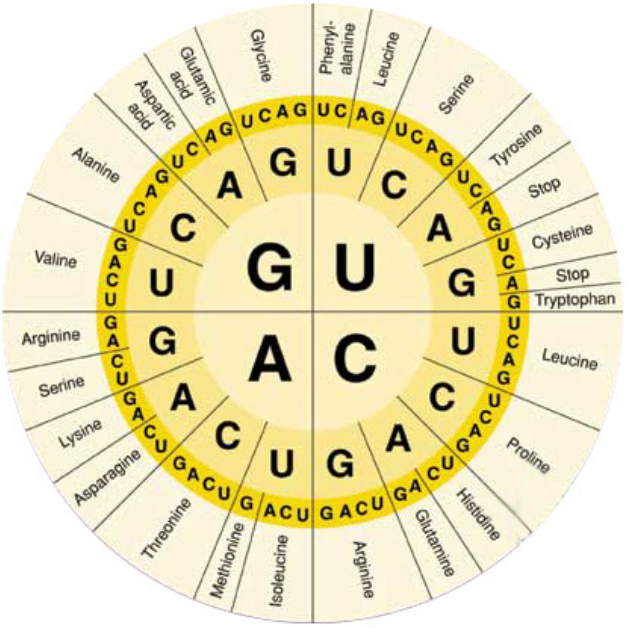


Figure 2.3 The genetic code table. Translation begins from the inner circle to the outer ones. For instance, the codon AUG is translated as *Methionine*.

or, alternatively, constitute other structures such as ribosomal RNAs, transfer RNAs or small nuclear RNAs.

RNAs that are the result of transcribing protein-coding genes undergo different modifications. First, the ends of these primary transcripts are modified to stabilize the molecule. Second, an editing process called splicing cuts and removes some fragments of the transcript (the introns) and pastes together the remaining ones that contain the information to build the protein (the exons). The processed RNA receives the name of messenger RNA or mRNA because it is then ready to leave the nucleus of the cell. For many genes, more than one splicing form is already known, increasing the volume of information contained in a given gene (**?**).

The final step is the translation of the mRNA, mediated by the rybosomes. The information contained in the sequence of nucleotides from the mRNA is used to produce a protein. Each group of three nucleotides (a codon) is translated into an amino acid that is added to the growing protein using the genetic code (see Figure 2.3). In eukaryotes, translation initiates at the start codon ATG while it is terminated when one of the stop codons TAA, TAG, or TGA is reached. Because of the length of a codon and the dual nature of the DNA molecule, there are always six different forms to translate a nucleic acid sequence: three reading frames (0,1,2) and two directions (forward and reverse).

DNA is only the carrier of genetic information in a cell. Proteins (often in combination with RNA molecules) are the biomolecules actually responsible for main cellular functions: they catalyze nearly all chemical processes in cells, give them their shape and movement capability, transmit signals through the body, recognize foreign molecules, or transport other elements.

Genes are not continuously being transcribed during each stage of the lifetime of the cell. According to every specific situation inside and outside of the cell, the need for some proteins to perform a given function launches the transcription of a subset of genes encoding those products. Contrarily, the excess of other

proteins prevents or stops the transcription of their genes. The activation of a gene is a complex procedure in which many actors play different roles in the genetic material of the cells.

Genome anatomy

In eukaryotes, DNA molecules are long linear polymers that can contain millions of base pairs arranged in an ordered sequence that encodes the genetic information of the cell. A million nucleotides measures a distance of approximately 0.03 cm, only occupying a volume of 10^{-15} cm^3. These tightly coiled packets consist of the double helical DNA structure wrapped around specific protein complexes called histones (**?**).

The genetic material of an organism is part of an apparently chaotic organization called chromatin during the entire lifetime of the cell except replication. However, the chromatin is condensed in individual units that receive the name of chromosomes when the cell is undergoing a nuclear division process. In both configurations, the complete set of DNA of an organism constitutes its genome. In Figure 2.4, a fragment of chromatin, a duplicated chromosome and the complete set of human chromosomes are shown. Only when the process of duplication of genetic material has been finished, the genome of the cell is arranged in two copies of the chromosomes to be distributed into the two new cells. In the meantime, the genome is in a semi-decondensed state in which the regions of chromatin containing genes are accessible for being transcribed.

Genomes widely vary in size because of many causes. The complexity of an organism is not directly related with the size of its genome or the number of genes encoded within. The size of several genomes in millions of base pairs is listed in Table 2.1. Interestingly, a substantial proportion of the genes are relatively conserved between different genomes due to the evolution process. The differences we observe between species are mostly because of minimal changes. For instance, the human genome sequence is 99% identical to the chimpanze sequence while the difference between two people is estimated to be less than 0.1%. One of the main types of sequence variation between individuals are the single nucleotide polymorphisms (SNPs). SNPs are sites in the genome where individuals differ in the DNA sequence by a single base. It is believed that there are at least 10 million SNPs in the human genome (DOE report, see Web Glossary, page **??**).

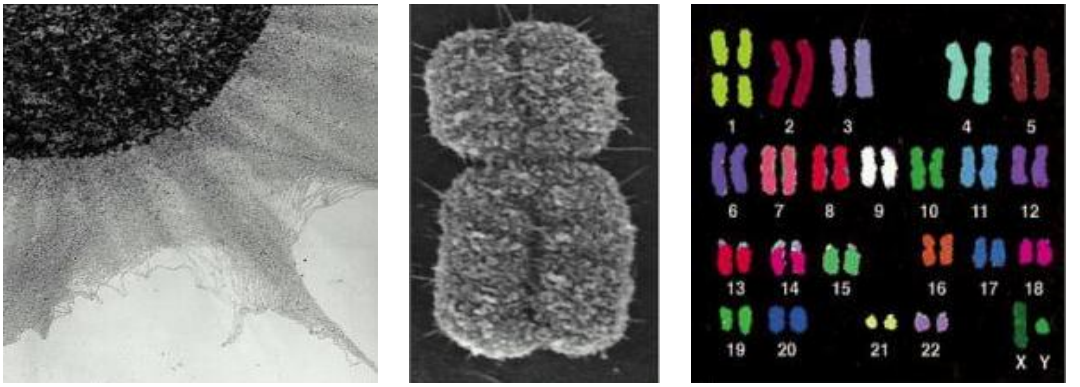
The genome is not exclusively a container of genes. On the contrary, the genomic landscape is rich and complex. Using the human genome as a reference, the protein coding fraction of the genome is only 2%. What is more, genes and related gene regulatory sequences actually occupy together a third part of the total three billion base pairs. As is represented in Figure 2.5, there is a huge part of the human genome called intergenic DNA which has been structurally characterized into different elements for which no known function has been assigned yet. They could play some role in chromosome structure and dynamics or might simply arise through an error in the process of copying the genome during cell division (**?**).

The bulk of this intergenic DNA is made up of repeated sequences. Repetitive DNA can be divided into two categories:

① Genome-wide or interspersed repeats. Repeat units distributed around the genome in an apparently random fashion. Transposable elements or transposons are mobile segments of DNA that are able to move around the genome from one place to another, leaving a copy of themselves in the original place.

② Satellite or tandemly repeated DNAs. Repeat units that are placed next to each other in an array. The commonest type of satellites are dinucleotide repeats and single nucleotide repeats.

Because of the complex nature of genomes, the annotation of the different elements that constitute the whole genomic landscape of a species is a non-trivial task and it requires many years and a lot of



A B C

Figure 2.4 A comparison of chromatin with a mitotic chromosome and the karyotype. (A) An electron micrograph showing a tangle of chromatin spilling out from a nucleus. (B) A scanning electron micrograph of a mitotic chromosome. The two copies are still linked. (C) Human chromosomes (karyotype). Staining is performed by exposing them to a collection of DNA molecules that have been coupled to a combination of fluorescence dyes. Adapted from **?**.

effort. Computers have been playing a key role in the major sequencing projects. Furthermore, they are still essential in the unveiling of the thousands of relationships between the genomic components that govern cell behavior.

SPECIES	COMMON NAME	GENOME SIZE	GENES
Saccharomyces cerevisiae	Yeast	12,156,590	6,680
Caenorhabditis elegans	Nematode worm	100,585,160	20,065
Anopheles gambiae	Mosquito	278,253,050	13,277
Apis mellifera	Honey Bee	228,567,597	13,448
Drosophila melanogaster	Fruit fly	144,138,837	13,985
Fugu rubripes	Pufferfish	393,296,343	22,008
Gallus gallus	Chicken	1,054,197,620	18,632
Mus musculus	Mouse	2,676,244,419	24,256
Rattus norvegicus	Rat	2,718,897,321	21,952
Bos taurus	Cow	1,741,208,718	23,231
Pan troglodytes	Chimpanze	2,733,948,177	22,475
Homo sapiens	Human	3,433,077,231	23,341
Triticum aestivum	Wheat*	17,000,000,000	50,000

Table 2.1 Comparison of the sizes of several eukaryotic genomes. Data extracted from ENSEMBL (May, 2006). Estimated values for wheat.

2.2 The genomic era

Bioinformatics

With major advances in the technologies that supply molecular data and the posterior explosive growth in the amount of available biological information, the application of computers to organize and understand this enormous volume of knowledge became essential. Bioinformatics is the field of science in which biology, computer science, information technology, mathematics and statistics converge to form a single discipline. The ultimate goal of bioinformatics is the combination of many sources of biological information to develop a comprehensive picture of normal cellular activities (NCBI report: bioinformatics, see Web Glossary, page **??**).

Broadly, bioinformatics tasks can be divided intro three categories:

① Implementation of databases to organize existing information from many areas of biological research such as genomics, transcriptomics and proteomics, allowing the public scientific community to efficiently access the data and to avoid redundancy and multiplicity. Doubtlessly, the advent of internet has played a central role in the achievement of this challenge (**?**).

② Development of new algorithms and statistics that aid the analysis of the data such as sequence alignment methods, motif detection techniques, phylogenetic studies or protein folding simulation. Advanced algorithmic methods and mathematical frameworks are essential to extract biological knowledge from the databases.

③ The analysis of such data and the interpretation of the results in a biologically meaningful manner to provide a more global perspective (new testable hypotheses) in future experimental designs. So far, it is far often easier to produce sequence data than to understand its function so that this is the most complicate of the three tasks (**???**).

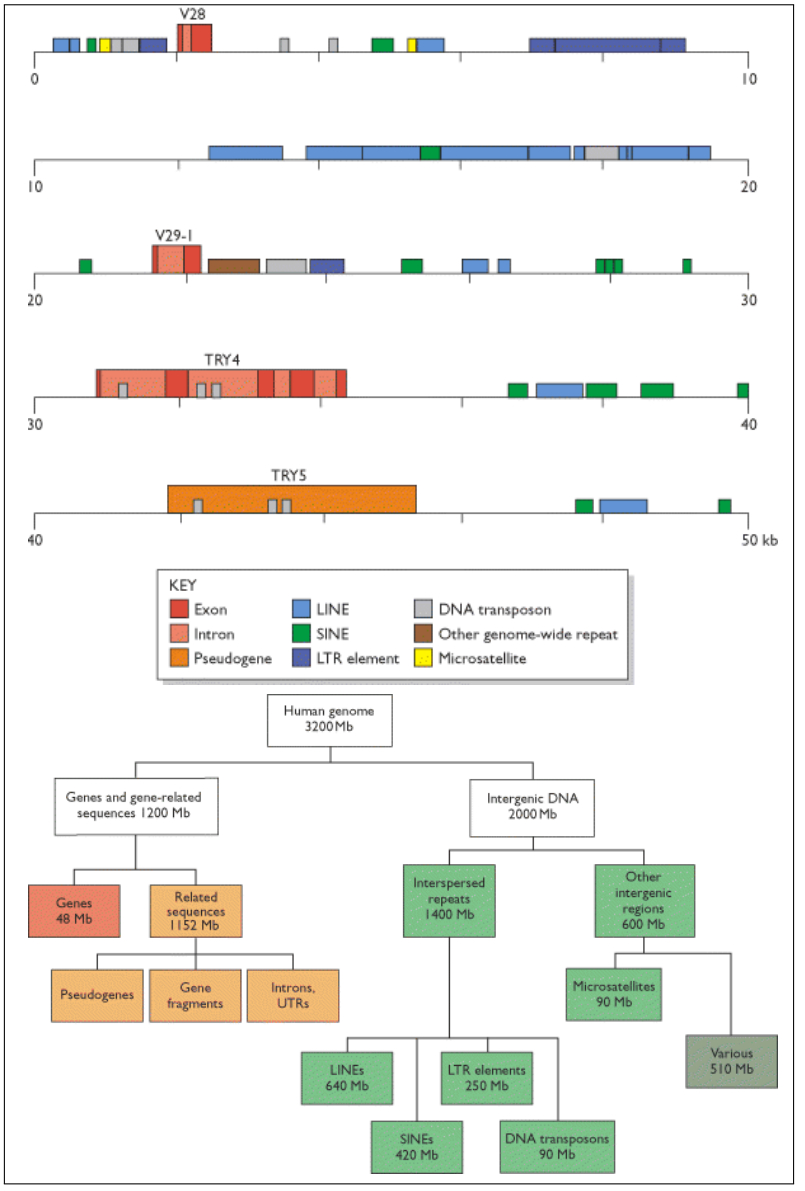


Figure 2.5 The organization of the human genome. (Top) A segment of the human genome. (Bottom) The contribution of different genomic elements to the human genome. Adapted from **?**.

Sequence databases

A biological database is a large, organized body of persistent data designed to be queried and retrieved in a very efficient manner by the scientific community. Because of the nature of the first data, ancient databases were merely collections of sequences of proteins distributed as a printed work (**?**). Nonetheless, the need

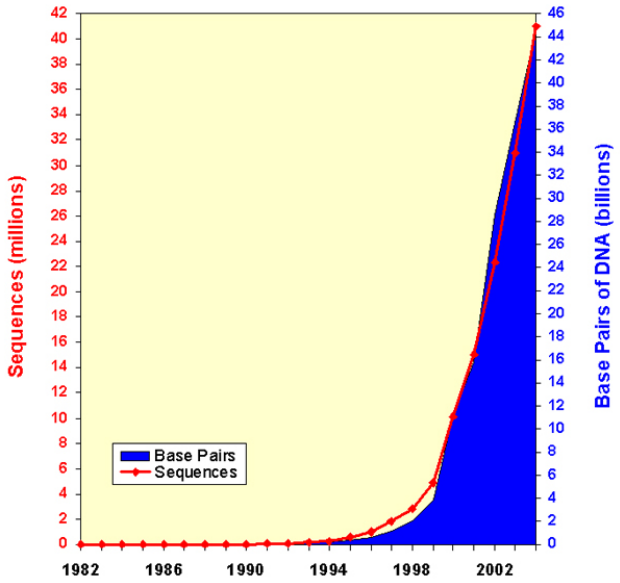


Figure 2.6 Growth of the GENBANK (1982-2004). Adapted from GENBANK (see Web Glossary, page **??**).

for an electronic format became obvious just when the amount of sequences was unmanageable (**??**). With substantial experimental sequencing improvements and the advent of DNA sequence databases initiated by the European Molecular Biology Laboratory (EMBL, Germany) and Los Alamos National Laboratory (LANL, United States), the number of available sequences experienced an exponential growth (see Figure 2.6).

Major public nucleotide and protein sequence databases such as EMBL (**?**, see Web Glossary, page **??**) or GENBANK[2] (**?**, see Web Glossary, page **??**) are repositories of sequences submitted by researchers in order to make them accessible for the rest of the biological community. An accession number and a set of annotations are provided for each sequence entry. Using flat files as a standard format, the features of each sequence are displayed in a simple format that divides each line of information into two elements: a field descriptor and a value. The popular `FASTA` format is one of the *de facto* standards that have been adopted to represent a sequence of nucleotides or amino acids (see Figure 2.7 for an example of a GenBank entry and the associated `FASTA` file).

Because of the relative lack of control over the quality and quantity of the data stored in the sequence databases during the first years, there was soon a necessity to maintain collections of data free of redundancy and errors constructed from the original repositories. Since then, numerous curated databases, also known as secondary databases, have appeared aiming to avoid any type of multiplicity and low quality data (**?**).

A successful example of these refined catalogues is the REFSEQ collection (**?**, see Web Glossary, page **??**). The major goal of this database is to provide a unique sequence for each molecule in the protein synthesis pathway (DNA, mRNA and protein). To reduce the noise produced by the representation of a single biological entity with many entries in the sequence databases, each biological entity is represented only once in REFSEQ, maintaining a non-redundant repository.

[2]GenBank is now under the auspices of the National Center for Biotechnology Information (NCBI, United States).

Genomic databases

Once the complete assembly of first eukaryotic genomes such *Saccharomyces cerevisiae* (**?**) or *Drosophila melanogaster* (**?**) was achieved, the principal focus of computational biology research shifted from individual sequences to chromosomes and whole genomes. With the release of the human genome (**???**), it became necessary to introduce an important change in the way the assemblies and the genome annotations were presented. Finally, the recent availability of the mouse genome (**?**), the chicken genome (**?**) and the sequencing of other model organisms has augmented the need for a new kind of tools to permit the annotation and comparison of many genomes in a more sophisticate form. In addition, support for genomes that have not been finished yet has also been crucial (archives of traces and preview releases).

There are three well established genome browsers that aim to fulfill this need:

- The ENSEMBL project (**?**, see Web Glossary, page **??**), a collaboration between the European Bioinformatics Institute and the Sanger Institute. The main browser currently provides a set of gene, transcript and protein predictions for each genome. Data is presented on pages called Views, each View showing a different level of detail.
- The UCSC GENOME BROWSER (**?**, see Web Glossary, page **??**), produced by the University of California, Santa Cruz Genome Bioinformatics Group. It serves annotations for many eukaryotic genomes, presenting the information in the form of tracks. Each track corresponds to a certain genomic feature.
- The NCBI MAP VIEWER (**?**, see Web Glossary, page **??**), provides maps for a lot of organisms, many of them without finished assembly. The browser is tightly linked to most services of the NCBI web. The information is displayed using maps. Maps are vertical representations of annotations along a given chromosome. There is a map associated to each genomic feature.

The core of the three browsers is the internal gene annotation pipeline that must be executed on every new sequence assembly of each genome. Genes are annotated according to experimental evidence and computational predictions. Comparisons between different genomes are also employed to improve the results. Moreover, other genome features such as regulatory regions, repeats, transcripts or sequencing markers are integrated with the sequence and the annotated genes. In Figure 2.8, a screenshot of the same gene displayed in the UCSC GENOME BROWSER and ENSEMBL is shown.

Data integration (integromics)

The biological information that can be now accessed in the databases has not been generated during a continuous process with several steps following an increasing order of complexity. On the contrary, different and discontinuous waves of genome-wide data have overlapped to form the current body of knowledge. The new high-throughput technologies that have arisen in the last decades have been the main catalyst conducing the progress. The first wave was the large-scale production of fragments of transcripts also named expressed sequence tags (ESTs). The second wave was originated by the sequencing of whole microbial organisms and was quickly followed by the achievement of the genomic sequence of many eukaryotic organisms including human. Simultaneously, microarrays and related technology have produce an overwhelming amount of expression data for which new analysis methods are still being designed (**?**).

In the near future, new waves of information are expected, such as the generation of maps of functional SNPs (see section 2.3), or the complex interaction networks produced by emerging systems biology (**?**).

Information technologies have adapted to the changing nature of the new data. With every explosion of new knowledge, previous procedures have been reused and others have been created from scratch to integrate the new type of data with the already existing information. The power of data integration arises not from the value of every separate kind of information but from the gain produced by the fusion of all of them. With the advent of more waves of knowledge, integromics will become absolutely essential to manage an amount of 'Omic' information that will exceed exabyte (10^{18} bytes) quantities (**?**).

Biological databases are essential resources used by biologists around the world. However, each one contains only a subset of biological knowledge. This specificity increases the complexity of finding the answer for the majority of questions. Thus, several databases must be explored in order to obtain the expected results. Cross-database queries require complex mechanisms of data integration that are often not implemented properly (**?**).

For instance, the name of biological objects such as genes in the genomic browsers of several species (e.g. Rad24, rad24 or RAD24) or the definition of simple entities such as the gene concept (considering only transcript or transcript and regulatory region) can be a source of disagreement. Consequently, the role of ontologies to facilitate data integration must not be neglected. The popular GENE ONTOLOGY (**?**, see Web Glossary, page **??**) establishes a taxonomy of controlled vocabulary that is used by most genome annotation projects to uniformly annotate the function of genes.

There are several ways in which databases developers have tried to integrate databases:

- Link integration. Hypertext links are used to jump from one database to another. Although it is the most popular solution, it has two severe drawbacks: links are vulnerable to name ambiguities and their updating is laborious.
- View integration. An environment around the databases is built to create the illusion of a unique resource formed by different sources of data with specific data drivers to retrieve the information. The complexity of such a design is the main disadvantage of this strategy.
- Data warehousing. Merge all of the databases into a single database. Due to the continuous updating of biological databases and the impossibility of reusing the software from one release to the next, this approach is unfeasible in practice.

2.3 The post-genomic era

New forms of investigation

The availability of many genomes and the improvement of very large-scale gene expression experiments have substantially modified the form in which current research is focused (**?**). The classical hypothesis-driven research paradigm, in which a specific proposition is addressed over a set of targets, is progressively being substituted with data-driven investigations, in which high-throughput explorations are performed typically over the whole collection of genes of an organism to detect previously unknown relationships. Data-diving excursions have several risks derived from their massive exploration. Correct normalization and replication of the results is extremely difficult. In addition, there is usually a high probability of finding pure artefactual relations due to the low signal/noise ratio observed in such experiments (**?**).

Much effort must be invested to make bioinformatics become part of the wet-dry cycles of research (**?**). Such discovery processes occur whenever a computational method is linked to a biological one, such that

predictions from the former can be tested at the bench, within a feedback strategy. Once the computational candidates have been delivered, they should be monitored during the experimental pipeline, using such results to refine the original computational model (**?**).

Genomics and health

Virtually every human illness has a hereditary component (**?**). The characterization of the genetic determinants of disease would provide remarkable opportunities for clinical medicine. Current clinical practice is still based on phenotypic criteria to define most diseases rather than studying the underlying mechanisms. Obtaining the sequence of the human genome is only the end of the beginning (**?**). Among the grand challenges to achieve after the sequence of many genomes is available is the development of strategies for identifying the genetic contributions to disease and the gene variants that promote good health and resistance to disease (**?**). Progress is slow but evidence suggests that while public health and antibiotics have played the major roles in the past 50 years, the next 50 are likely to belong to genetics and molecular medicine (**?**).

Simple changes in our genes can lead to disease. Single gene mutations, which are already commonly used in diagnostic practice (genetic and disease markers), cause approximately $6,000$ inherited diseases also known as monogenic diseases. Disorders like cystic fibrosis, anemia or hemophilia affect millions of people worldwide. For more common diseases such as heart disease, diabetes, or Alzheimer's disease, the interplay of multiple genes and multiple non-genetic factors (environment effects) that contribute to disease susceptibility is still being characterized (GSK report: Genes and diseases, see Web Glossary, page **??**, NHGRI/NIH report: Genetics, the Future of Medicine, see Web Glossary, page **??**).

For example, loss of control in the growth mechanisms of cells results in cancer. The transformation of a normal cell into a cancerous one is caused by molecular changes that underly growth-signal independence, insensitivity to anti-growth signals, evasion of immunosurveillance, apoptosis evasion, unlimited replicative potential, tissue invasion and metastasis. These molecular changes involving several genes can be produced by certain events that alter the genome such as point mutations, gene amplifications and deletions, and chromosomal translocations. The intimate relationship between cancer and genome sequencing projects has originated the recent launch of several cancer genome projects (**?**).

Pharmacogenomics

Before the end of this century, shortly after a person is born, her genotype will be saved at her physician's office to record the presence or absence of specific variations known to be relevant for assessing disease susceptibility and prediction response to drug types. Biomolecular profiling throughout her life will complement this information to provide recommendations about life-style or diet and to detect early stages of a disease. This future scenario in which personalized medicine and therapy are present in our lives to increase the quality of life and life-span is not unrealistic (**?**).

In 1998, adverse drug reactions produced over $100,000$ deaths in the United States, being one of the leading causes of hospitalization and death. The one-size-fits-all formula typically works for only 60% of the population at best. The way a person responds a drug (positively or negatively) is a complex trait influenced by many different genes. Pharmacogenomics [3] is the science that examines the gene variations that dictate

[3]The related term pharmacogenetics appeared in the 1950s describing the study of inherited genetic variation in drug metabolism and response.

drug response and explores how to use them to predict whether a patient will have a good reaction, a bad reaction or no reaction to a given drug (**?**, NCBI report: pharmacogenomics, see Web Glossary, page **??**).

First studies focused on the broadest categories of inheritance: ethnicity, geography, language and race. Several SNPs mapping projects are working to provide a catalogue of observed one-letter differences between individuals in a population. SNPs are present throughout the human genome with an average frequency of 1 per $1,000$ base pairs. Their relatively even distribution make them valuable as genetic markers. To be helpful, the polymorphism must be shared by at least 1% of the population tested, thus becoming a shared SNP. Mutations are less common differences, occurring in a smaller proportion.

With these SNP maps, genetic profile comparison of patients who may suffer from serious side effects and those that may not, might be useful to detect one or more SNPs that differ between both groups. Careful examination of the small area of the genome where the differences are found will classify them into functional and non-functional SNPs (see Figure 2.9). For instance, SNPs found in protein coding regions (cSNPs) would be good candidates to elaborate a hypothetic explanation of the observed drug response as long as they produce a change in the translated amino acid sequence (non synonymous changes).

The haplotype is the set of closely related genes (alleles) that tend to be inherited together as a single unit. The International HapMap Project is currently in charge of developing the haplotype map of the human genome (**?**). The official repository of SNPs mined by this project is the NCBI DBSNP database (see Web Glossary, page **??**) that contains information for other genomes as well. SNP annotation is also integrated in the genomic browsers explained in Section 2.2. For further information about sequence polymorphisms, see **?**.

GENBANK

```
1: U30787. Reports Human uroporphyri...[gi:1322018]

LOCUS       HSU30787                4514 bp    DNA     linear   PRI 16-MAY-1996
DEFINITION  Human uroporphyrinogen decarboxylase (URO-D) gene, complete cds.
ACCESSION   U30787 X06048
VERSION     U30787.1  GI:1322018
KEYWORDS    .
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Catarrhini;
            Hominidae; Homo.
REFERENCE   1  (bases 785 to 1143)
  AUTHORS   Romana,M., Dubart,A., Beaupain,D., Chabret,C., Goossens,M. and
            Romeo,P.H.
  TITLE     Structure of the gene for human uroporphyrinogen decarboxylase
  JOURNAL   Nucleic Acids Res. 15 (18), 7343-7356 (1987)
   PUBMED   3658695
REFERENCE   2  (bases 1 to 4514)
  AUTHORS   Moran-Jimenez,M.J., Ged,C., Romana,M., Enriquez De Salamanca,R.,
            Taieb,A., Topi,G., D'Alessandro,L. and de Verneuil,H.
  TITLE     Uroporphyrinogen decarboxylase: complete human gene sequence and
            molecular study of three families with hepatoerythropoietic
            porphyria
  JOURNAL   Am. J. Hum. Genet. 58 (4), 712-721 (1996)
   PUBMED   8644733
REFERENCE   3  (bases 1 to 4514)
  AUTHORS   Romana,M., Ged,C. and Verneuil,H.
  TITLE     Direct Submission
  JOURNAL   Submitted (30-JUN-1995) Cecile Ged, Biochimie Medicale et
            Moleculaire, University of Bordeaux II, 146 rue Leo Saignat,
            Bordeaux, France

FEATURES             Location/Qualifiers
     source          1..4514
                     /organism="Homo sapiens"
                     /mol_type="genomic DNA"
                     /db_xref="taxon:9606"
     protein_bind    1028..1037
                     /bound_moiety="Sp1"
     TATA_signal     1066..1073
     gene            1089..4514
                     /gene="URO-D"
     mRNA            join(1089..1126,1748..1860,1976..2055,2132..2194,
                     2434..2631,2749..2910,3279..3416,3576..3676,3780..3846,
                     4179..4514)
                     /gene="URO-D"
                     /product="uroporphyrinogen decarboxylase"
     CDS             join(1107..1126,1748..1860,1976..2055,2132..2194,
                     2434..2631,2749..2910,3279..3416,3576..3676,3780..3846,
                     4179..4340)
                     /gene="URO-D"
                     /codon_start=1
                     /product="uroporphyrinogen decarboxylase"
                     /protein_id="AAC50482.1"
                     /db_xref="GI:1322019"
                     /translation="MEANGLGPQGFPELKNDTFLRAAWGEETDYTPVWCMRQAGRYLP
                     EFRETRAAQDFFSTCRSPEACCELTLQPLRRFLLDAAIIFSDILVVPQALGMEVTMVP
                     GKGPSFPEPLREEQDLERLRDPEVVASELGYVFQAITLTRQRLAGRVPLIGFAGAPWT
                     LMTYMVEGGGSSTMAQAKRWLYQRPQASHQLLRILTDALVPYLVGQVVAGAQALQLFE
                     SHAGHLGPQLFNKFALPYIRDVAKQVKARLREAGLAPVPMIIFAKDGHFALEELAQAG
                     YEVVGLDWTVAPKKARECVGKTVTLQGNLDPCALYASEEEIGQLVKQMLDDFGPHRYI
                     ANLGHGLYPDMDPEHVGAFVDAVHKHSRLLRQN"
ORIGIN
        1 aagcttcgta agcacctctc gcggcacgaa agccagcgct gcctaggcgc cgcccggcgc
       61 gaggctctca cctctgccaa gaagcgcacc ggcccagcag ctgccggggg gactccagca
      121 ccgcgccggg ccatggaccc gccatgagtc agctggcgcg accgcggaca gagcttccca
      181 ccacgccctt cccgcctttt ggccagcctt tgccgtatgt tctggactaa gcgcacccca
```

FASTA

```
>gi|1322018|gb|U30787.1|HSU30787 Human uroporphyrinogen decarboxylase
AAGCTTCGTAAGCACCTCTCGCGGCACGAAAGCCAGCGCTGCCTAGGCGCCGCCCGGCGCGAGGCTCTCA
CCTCTGCCAAGAAGCGCACCGGCCCAGCAGCTGCCGGGGGGACTCCAGCACCGCGCCGGGCCATGGACCC
GCCATGAGTCAGCTGGCGCGACCGCGGACAGAGCTTCCCACCACGCCCTTCCCCGCCTTTGGCCAGCCTT
TGCCGTATGTTCTGGACTAAGCGCACCCCAGCTCTCACTGTATTGGACTGTGTACTCCCACACTCAACCA
TATTACTTATCTCTGTGCCACCCTAACCCAGCCGACCAAACCCAAGATTGGTGATTGCTACCTGATCAAT
CTCCCTCTCTCCATTTCCTTGTGACTACCATTTTATCTCTACTGCTACTACCCTCATTCAAGTCACCATT
CTAGCTAGCCTGGGTCATTGCCAACAGTCATTTTTCTGGTTCTTCGGCCTGCTGTTTTTCCTCCCACTCC
CAGCGAATCTGCTGGACTCCCTATCCTATGGGTGGTGTGATTAAAGTGTTTGAGACAATGGCCCCTTCCC
CTGCCACTGACAGGAGTCTTGAGTCATTAGGGTTGAGTTCTGTTTGACACTCCTAATCCCAAGGACACTG
GAGATCATTATTCATTTTAATGTGATTGCTGATTTCTGTTTCCCCAGTCTTGTAGCTCCTTAAAGGCTGG
GGTGTCTTGAGCAGAGCTAACCTCTGCACCTACTATAGGTCCAGGCTATAGTATGGACCTGGCTGGATAA
GACTGTTGGTATCATAGTTGGGACTTGCGCCAAGCTCCGGATACCCAGACTGTCAGATGAGAACAAATTC
CTCATGTCACCGTAAGATACATTTACAGCGGAGTTTTCTTTTGGGCCTTTGTTGTTTCGTCGCTACAGCA
AACTTTACGGTGAAAAAAGGTAGGGGTCTACGGCAGCAGCAGGGCAGCCCTGGAGCTGTCGCTGGAGTCC
GATCATGTGATCTTCAACATGGCGACGCTCTTGGTTCCCTACAGAAAGGGGCGGAGCCTGGACTGGGGGG
CAGGCTCAGATTCAGGTTAAATTGTGGATTGAGCTCGCAGTTACAGACAGCTGACCATGGAAGCGAATGG
GTTGGGGTGAGTTCTCCAGAGCACGCGGTGTGGCTAGCCGGGCTTCTAATTTGAGTCTTCCAACTCAGGA
CTCTATCCCTCTACTCCCCTTTCCCCACCCTGGAGAACCTCCCAACCTGAACTCCGTTAGCTGGATCCTG
AATCCTAAAACCATGGATTTTTGAGATGTTCATCCCAGGGCCTTAATTCAAGGGATGCCTCAGGATTTCC
AACCAGGATCTTCATTCTGGGACCATCAACTCTGATCCCTCTTTATCCCCCAGCCTGGGTATTTCTCAGC
CCCTGAACCAGCCCAGTGACATTTCCCGGTTTCTGAGGCTCACTAGTTCGAAGACCCCCAAACTATCCTT
AGTGGGCCTTCATTCCCTCCCCCCAGTCCCTCTGGTTGCTTCGAGCTTGGAAGAGTAGAGACTAAGTGGA
```

Figure 2.7 An example of GENBANK entry and a FASTA sequence.

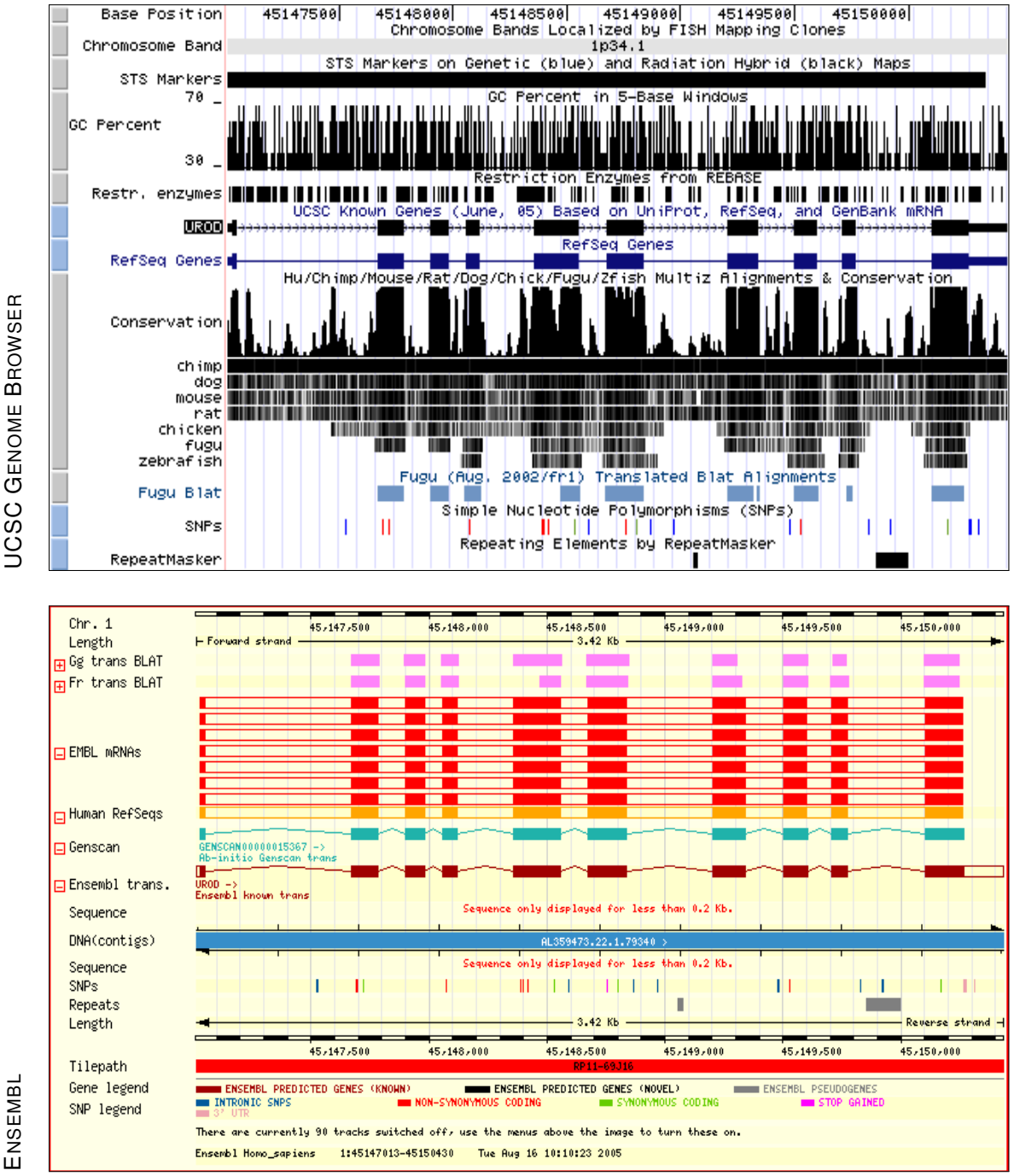


Figure 2.8 The human URO-D gene in the UCSC GENOME BROWSER and ENSEMBL.

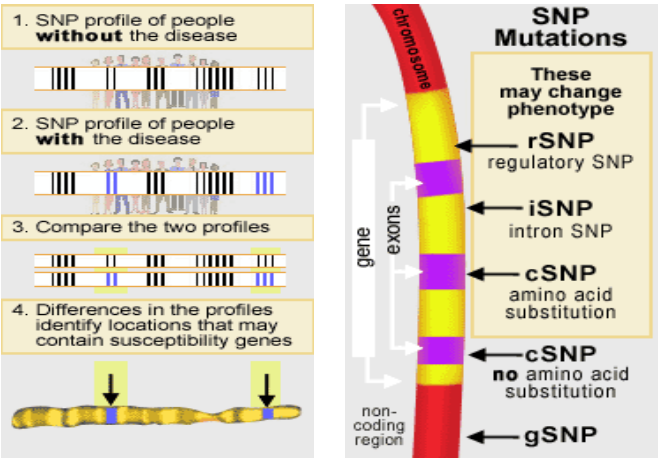


Figure 2.9 Using SNPs to locate susceptibility genes. (Left) SNP profiling of two groups of people. (Right) Categories of SNPs according to their location. Adapted from GSK report: Genes and diseases.

PART II

Estado del Arte

Chapter 3

The golden age of sequence analysis

Summary

This chapter aims to be a historical survey of the sequence comparisons algorithms analyzing the most relevant solutions. The algorithms that represented innovative changes in the field are described in detail, covering the concepts of global, local and multiple alignment of sequences. In addition, the theoretical framework of the map alignment problems necessary to understand the rest of work presented in this thesis is also formalized here.

3.1 Foundations of sequence comparison

THE TOPIC OF BIOSEQUENCE COMPARISON has a rich history dating back over 40 years. It is certainly very difficult to trace a line in some moments to establish the order in which every new development was presented because of the enormous body of publications that have contributed substantially to improve this field. Several general reviews have been used to reconstruct the history of biological sequence comparisons **??????**.

Molecular evolution began to be studied in the 1960s when a few protein sequences were available, being published into the protein sequence atlas (**?**). Soon, pioneering analysis appeared to infer the evolutionary relationships from these sequences, depicted as distances in phylogenetic trees (**?**).

Outside the molecular biology, other significant advances in mathematics and in the emerging discipline of computer science contributed decisively to the current state of the art. For instance, it is impossible to understand the history of modern sequence alignment without mentioning the birth of a new technique in the 1950s to solve multistage decision process problems called dynamic programming (**??**). A problem is solved by dynamic programming if the answer can be efficiently determined by computing a table of optimal answers to progressively larger subproblems. The principle of optimality requires that the optimal answer to a given subproblem is expressible in terms of optimal answers to smaller subproblems. During all this time, despite innumerable optimal and heuristic approaches have been proposed to obtain the best alignments between two sequences with the minimum cost, dynamic programming is still the most stable technique to solve the original problem and many of its variations.

Another key concept is the definition of several metrics of distance between sequences in the coding theory field. Since noise in a transmission channel introduces errors into the signal reception, several mechanisms were developed for detection and correction of such errors. The Hamming distance, defined as the number of positions in which two sequences differ, was oriented to detect only substitutions (**?**). Next, **?** presented the edit distance, which was the earliest known use of a distance function that is appropriate to detect insertions and deletions of symbols in the original message.

It is not clear when the basic dynamic programming algorithm for molecular sequence comparison first appeared. It was probably rediscovered many times in different contexts. The well-known paper by **?** who presented an algorithm for maximizing the number of matches minus the number of insertions and deletions is generally considered to be the first important contribution. Although no complexity analysis was provided, the original **?** algorithm measured the homology between two sequences in a $O(n^3)$ time.

A more rigorous approach with solid mathematical foundations arised from the problem of computing the distance between two sequences (**??**). **?** presented a dynamic algorithm based on the Levhenshtein metric distance. Though less flexible for future variations of the problem, this new approach fitted better with the perspective of evolutionary distance analysis developed earlier. Under the realistic assumption that both sequences have n nucleotides, the **?** algorithm have computation time proportional to $O(n^2)$. A comprehensive study of equivalence between similarity and distance was presented in **?**.

Within the field of computer science, sequence comparison appeared in simpler incarnations of the molecular biology problems, for comparing the contents of files or correcting the spelling of words. For example, the longest common subsequence problem (LCS) consists on finding an alignment that maximizes the number of identical aligned pairs between two sequences (see **?** for a review). Interestingly for long sequences, **?** applied the divide and conquer strategy to solve the LCS problem in $O(2n^2)$ time with a linear space cost instead of the established quadratic cost. **?** generalized this technique to align two sequences using $O(n)$ space.

Nonetheless, the treatment of gaps was still biologically unrealistic as a deletion of n symbols and n deletions of one symbol were punished indistinctly. **?** accommodated the same algorithm to deal with multiple deletions and insertions, introducing the concept of general gap penalty functions. **?** reduced the asymptotic cost from $O(n^3)$ to $O(n^2)$, under the application of the affine gap penalty functions in which there was an initial penalty for opening a gap and an additional minor penalty for extending an existent one. Apart from general and affine gap functions, **?** introduced the concept of concave gap function in which the cost of extending an existent gap grows with the logarithm of the length of the gap as a continuous curve. Later, **?** and **?** independently arrived at $O(n^2 \log n)$ solutions of the problem.

DNA and protein sequences are the result of an evolutionary process that tend to preserve those parts that are key to perform a function, permitting variation in the rest. Thus a global comparison can easily produce a very poor alignment of two sequences that have some parts in common while others are completely free of conservation. **?** introduced the concept of local alignment with a simple variation in the basic global similarity algorithm without increasing its cost. Under the premise of a negative gap penalty, reported alignments are regions of high similarity with a positive score within. **?** tried to export the same concept to the distance metric. Only, those paths in the matrix whose density of mismatches was below a certain threshold were reported.

Thousands of genomic and proteomic sequences, that is millions of nucleotides and amino acids, are rapidly being accumulated in the biological databases. However, searching a database with a query sequence for similarities to other sequences using the optimal algorithms enumerated above is clearly unfeasible when this simple operation involves thousands of comparisons between two sequences. To overcome this problem, a new family of heuristic procedures that produce nearly correct answers in a simple and cheaper fashion was designed. The most popular representatives of these are the program FASTA (**?**) and the program BLAST (**?**). The FASTA heuristic is based on identifying the identities between two sequences (diagonals in the matrix) and then applying some more expensive procedures only on those subalignments. BLAST processing relies on first, detecting ungapped segment pairs of high score and then, extending them from both ends until a threshold value is reached.

A collateral effect of producing hundreds of alignments was the concern about the quality of a given alignment between two sequences. The significance of a local alignment score can be tested by comparing with the distribution of scores expected by aligning two random sequences with the same length and composition (**?**). These random sequence alignment scores follow a distribution called the extreme value distribution (also known as the Gumbel distribution), which is similar to a normal distribution but with a positively skewed tail in the higher score (**?**). Less interest has traditionally been focused on global comparisons because of a global alignment is always produced by definition even between random or unrelated sequences, growing the score proportionally to the length of them.

In attempt to distinguish more distant relationships, the implementation of comparisons for more than two sequences is the logical evolution to locate elements with function that are conserved for instance in several homologous sequences. **?** naturally extended the basic dynamic programming recurrence for k sequences, with an exponential cost $O(n^k)$. As this approach is generally impractical, some heuristics appeared to solve the problem with a minor cost. The most popular of them is the hierarchical or clustering method called progressive alignment that first takes $O(k^2n^2)$ to perform all pairwise alignments and second, produce a multiple alignment following a guide tree to merge these alignments (**?**). The program CLUSTALW (**?**) combines this strategy with different weighting schemes according to the progression in the distances tree. Previously, **?** developed another method based on identifying the projections of the pairwise alignments that can form the multiple alignment. Moreover, hidden Markov models have been used to produce multiple alignments of a family of sequences to which more members can be dynamically be added (profile HMMs, see **?**).

Pattern discovery and local multiple sequence alignment have been very closely related problems (**?**). For instance, a conserved pattern or a block of ungapped common motifs in a set of sequences defines a local multiple alignment. In any case, the problem is even more difficult than pure global alignment and optimal approaches were discarded beforehand. Some heuristic approaches have been proposed to circumvent the complexity. Iterative methods do not necessarily find the best pattern, but may converge to a local maximum. Gibbs sampling (**?**) and expectation maximization (**?**) are successful examples of these stochastic techniques.

Some pattern recognition problems are too complex or too ambiguous to be expressed as a simple pattern matching operations over a sequence. In these cases, a richer environment over the basic sequences is needed to describe the comparison of such elements (**?**). For example, for most sequence comparison problems there is a corresponding map comparison algorithm. Map comparisons were introduced to model the alignment of restriction enzyme maps. These were used in the construction of physical maps prior to genome sequencing projects. The basic definition of the problem by **?** contained an $O(n^4)$ time cost algorithm although it was noticed the dynamic programming matrix was very sparse. Later, **?** improved the time efficiency by using an analytical approach that reduced the cost to $O(n^2 \log n)$. Additional refinements of the problem produced new algorithms to deal with map data errors (**?**) or to align specifically short maps to longer ones (**?**).

Not only analytical approaches have been employed for comparing sequences. Dot matrix comparisons, also known as dotplots, are visual comparisons that can be useful to conduct afterwards a deeper research with dynamic programming algorithms only on those conserved regions (**?**). Sequence logos are graphs that illustrate the amount of information in each column of an alignment or motif (**?**).

Sequence comparison algorithms that were developed to solve biological problems have been recreated and applied in other scientific fields (**?**). For instance, applications can be found in geology (stratigraphic sequences), in dendrochronology (time dating based on tree rings), or in bird song recognition (animal communication).

3.2 Alphabets, sequences and alignments

Biological significance of sequence comparison

Gene evolution is thought to occur by gene duplication, creating two tandem copies of the gene in a given ancestor species. In rare cases, new mutations in one of the copies can provide an advantageous change in function. The two copies then evolve along separate pathways. At a certain evolutionary point, a speciation event gives rise to two separate branches (two new species) of the tandem gene preserving a similar sequence due to the single gene ancestor (see Figure 3.1). The four copies of the original gene are said to be homologous: the two corresponding units of the tandem gene in each species are orthologous while the two units of each tandem gene in the same species are paralogous. Molecular evolution events include substitutions of one nucleotide or amino acid for another as well as insertions and deletions (indels) of others. More complex genetic rearrangements such inversions, transpositions, translocations or duplications can shuffle larger parts of the genes or of the proteins, producing chimeric products in which some regions are homologous and others are not (**?**).

Sequence comparison consists of finding which parts of the sequences are alike and which parts differ. This operation is extremely useful for discovering functional, structural and evolutionary information in biological sequences. If two sequences from different organisms are similar, there may have been a common ancestor sequence that would make these sequences to be homologous. Phylogenetic analyses are usually

conducted starting from multiple sequence comparisons, and then producing hierarchical trees that would explain the evolution of the species.

Alphabets and sequences

A finite alphabet is a set of symbols or characters. For instance, the four-letter DNA and RNA alphabets are defined as:

$$\Sigma_{DNA} = \{A, C, G, T\} \text{ and } \Sigma_{RNA} = \{A, C, G, U\}.$$

To support some degree of variation or ambiguity in a symbol, the IUPAC extended genetic alphabet of 15 elements allows for special symbols possessing multiple letters (see Table 3.1). The single-letter amino acid alphabet contains 20 elements [1] from which all proteins are built (see Table 3.2).

Σ^* denotes the set of all finite sequences of characters from Σ including the empty sequence λ. A generic sequence S of length $|S| = n$ symbols over a finite alphabet Σ is defined as:

$$S = s_1 s_2 \dots s_n \text{ where } \forall i : 1 \leq i \leq n : s_i \in \Sigma.$$

A subsequence of S between positions i and j of S is the contiguous series of elements between both positions[2]. If $i = 1$, the subsequence is called a prefix of S. If $j = n$, the subsequence is a suffix:

$$S_{i,j} = s_i \dots s_j \text{ where } 1 \leq i \leq j \leq n \text{ and } \forall k : i \leq k \leq j : s_k \in S.$$

Sequence alignments

Given two sequences $A = a_1 a_2 \dots a_m$ and $B = b_1 b_2 \dots b_n$ in a finite alphabet Σ, a sequence alignment of A and B is a correspondence C between the symbols from the two sequences

$$C(A, B) = \{(a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2}) \dots (a_{i_T}, b_{j_T})\} \text{ where}$$
$$1 \leq i_1 \leq i_2 \leq \dots i_T \leq m, 1 \leq j_1 \leq j_2 \leq \dots j_T \leq n$$

such that:

① Each a_k (or b_l) not appearing in the subsequence $a_{i_1} \dots a_{i_T}$ (or $b_{j_1} \dots b_{j_T}$) is considered to be an insertion in the other sequence (or a deletion in this one).

② If the pair $(a_i, b_j) \in C \Rightarrow \forall k : b_k \in B \wedge k \neq j : (a_i, b_k) \notin C$ (one symbol only matches another symbol at most).

③ If the pairs $(a_i, b_j), (a_k, b_l) \in C$ and $i < k \Rightarrow j < l$ (no inversions are allowed).

---

[1] Nowadays, new amino acids are still being unveiled such as Selenocysteine.

[2] As defined in computer science, subsequences are subsets of characters of S possibly not contiguous but arranged in their original relative order.

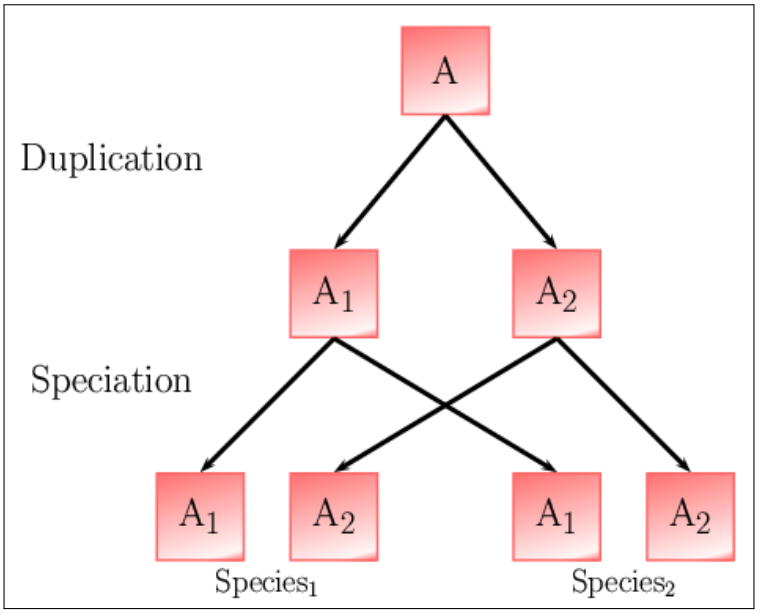


Figure 3.1 Gene evolution events.

For example, a possible alignment of the sequence A = AAGTTC and the sequence B = AGCCC is

$$\begin{array}{lcccccc} A = & A & A & G & T & T & C \\ & | & & | & & & | \\ B = & A & - & G & C & C & C. \end{array}$$

This alignment represents a certain hypothesis about the evolution of the two sequences (**?**): three of the nucleotides have not changed since the common ancestor of A and B (matches), there have been at least two substitutions (mismatches), and one nucleotide has been either inserted or deleted (a gap), which is denoted with the symbol "–".

If we adopt a scoring function that assigns a given value to a match, a mismatch and a gap, every column of the alignment will receive a score and the total score of the alignment will be the sum of the values assigned to its columns. The best alignment will be the one that optimizes the total score. In the literature, two different types of measures have been devised to construct such a scoring function : similarity and distance (see **?** for a review).

Sequence similarity

Similarity is a measure of how alike two sequences are. An alignment is scored by rewarding the identities and in less degree, the substitutions, and punishing the gaps.

Let (a_i, b_j) be a match (or a mismatch) of type k with a weight α_k and let w_l be the weight associated to a gap of length l. Then, the similarity of an alignment C of A and B with λ_x matches of type x and Δ_y gaps of length y is

SYMBOL	LETTERS	ORIGIN OF DESIGNATION
A	A	Adenine
C	C	Cytosine
G	G	Guanine
T	T	Thymine
R	A or G	puRine
Y	C or T	pYrimidine
M	A or C	aMino
K	G or T	Keto
S	C or G	Strong interaction (3 H-bonds)
W	A or T	Weak interaction (2 H-bonds)
B	C or G or T	not A, B follows A
D	A or G or T	not C, D follows C
H	A or C or T	not G, H follows G
V	A or C or G	not T (not U), V follows U
N	A or C or G or T	aNy

Table 3.1 The IUPAC extended genetic alphabet.

$$S(C) = \sum_{x} \lambda_x \alpha_x - \sum_{y} \Delta_y w_y. \tag{3.1}$$

The best alignment is the one that maximizes the similarity between A and B. The similarity can increase and decrease during the computation of an alignment score from $-\infty$ to ∞ (from dissimilarity to similarity, where 0 means absence of any type of similarity).

Sequence distance

Distance (also called edit distance) is the minimal number of changes (indels and substitutions) needed to transform one sequence into another. An alignment is scored by charging a cost to each difference in the aligned sequences (0 for exact matches).

Let (a_i, b_j) be a match (or a mismatch) of type k with a weight β_k and let w_l be the weight associated to a gap of length l. Then, the distance of an alignment C of A and B with λ_x matches of type x and Δ_y gaps of length y is

$$D(C) = \sum_{x} \lambda_x \beta_x + \sum_{y} \Delta_y w_y. \tag{3.2}$$

The best alignment is the one that minimizes the distance between A and B. Distance metric provides a more biologically natural way to compare sequences, estimating the evolutionary time that has elapsed since the sequences diverged from a common ancestor. The distance value can only increase during the computation of an alignment score, starting with a value of 0.

LETTER	ABBREVIATION	FULL NAME
A	Ala	Alanine
R	Arg	Arginine
N	Asn	Asparagine
D	Asp	Aspartic acid
C	Cys	Cysteine
Q	Gln	Glutamine
E	Glu	Glutamic acid
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
L	Leu	Leucine
K	Lys	Lysine
M	Met	Methionine
F	Phe	Phenylalanine
P	Pro	Proline
S	Ser	Serine
T	Thr	Threonine
W	Trp	Tryptophan
Y	Tyr	Tyrosine
V	Val	Valine

Table 3.2 The amino acid alphabet.

The number of alignments

The number of possible alignments between two sequences of n symbols can be computed with the following function (**??**):

$$g(n) \sim \frac{2^{2n}}{4\sqrt{n\pi}}. \tag{3.3}$$

For two sequences of $1,000$ nucleotides, $g(n) > 10^{600}$. As direct examination of all these alignments is in practice impossible, computational approaches are therefore essential to calculate the optimal alignment without exploring all of the combinations.

Classes of sequence alignments

According to the type of comparison that must be performed between sequences, sequence alignments can classified as (**?**):

- Global alignments: the entire sequence length must be aligned to include the maximum number of matches. Sequences that are quite similar and approximately have the same length are good candidates for global alignment.

```
L G P S S K Q T G K G S – S R I W D N
|       |     | | |       |       |
L N – I T K S A G K G A I M R L G D A
```

- Local alignments: only the stretches of the sequences with the highest density of matches are aligned. Sequences that differ in length or that only share certain regions are suitable candidates for local alignment.

```
– – – – – – – T G K G – – – – – – – –
          | | |
– – – – – – – A G K G – – – – – – – –
```

When the number of sequences is two, such alignments receive the name of pairwise alignments as the examples above. If the number of input sequences is higher, they are called multiple sequence alignments:

- Global multiple alignments: the whole set of sequences is aligned at their entire length. Simply known as multiple alignments, they are the starting point for evolutionary modeling. Each column of the alignment is examined and significant changes observed in this position collaborate in the construction of a phylogenetic tree.

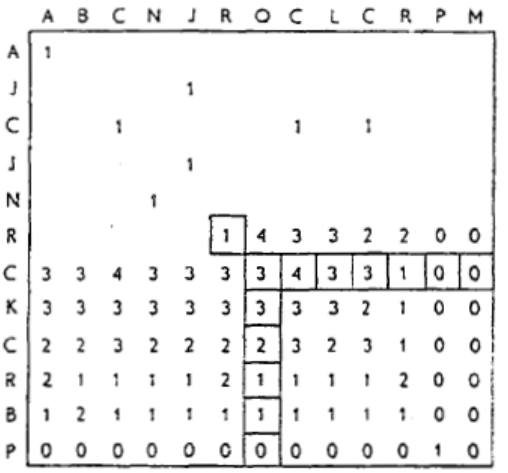
```
L G P S S K Q T G K G S – S R I W D N
|         | |                     |
L N – I T K S A G K G A I M R L G D A
|         | |                     |
L N – K Q Q S A G K C A I M – L G D A
```

- Local multiple alignments: they are equivalent to searching a pattern conserved in a set of sequences. Rather than be defined as a form of alignment, it is conceptually considered a pattern discovery problem.

```
– – – – – – – T G K G – – – – – – – –
– – – – – – – A G K G – – – – – – – –
– – – – – – – A G K C – – – – – – – –
```

3.3 An anthology of algorithms for global alignments

This section aims to be a catalogue of different approaches to solve the global pairwise alignment which was the first problem introduced in the field of sequence comparisons. Naturally, the extension to the multiple alignment of sequences has been also treated although optimal solutions were discarded because of their expensive time and space costs. Different heuristics to cope with multiple alignment are explained in detail in Section 3.5.



	A	B	C	N	J	R	O	C	L	C	R	P	M
A	1												
J					1								
C			1					1		1			
J					1								
N				1									
R						1	4	3	3	2	2	0	0
C	3	3	4	3	3	3	3	4	3	3	1	0	0
K	3	3	3	3	3	3	3	3	3	2	1	0	0
C	2	2	3	2	2	2	2	3	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
B	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 3.2 The maximum-match operation for necessary pathways. The cell $(R, R, 1)$ corresponds to the current $M(i, j)$. Adapted from **?**.

The Needleman and Wunsch algorithm (1970)

For the authors, the similarity or maximum match value between two proteins depends on the largest number of amino acids from the first protein that can be matched with those of the second one allowing possible interruptions in either sequence.

Each pair of amino acids from each sequence is the smallest unit of significance. All possible pair combinations are represented in a two-dimensional matrix M. The pathways through the cells of the matrix are representations of every possible comparison of the two sequences. If a given value is assigned to each identity and mismatch, the maximum match between two sequences A and B is then the largest number that would result from the sum of the cell values of every pathway.

The original **?** algorithm is actually a description of a method to systematically count the number of identities (denoted as 1's in the simplest formulation) between both sequences. No complexity analysis was provided although a careful analysis determines the cost of the process is cubic (see next section). In addition, the authors implicitly suggested the extension of the method to allow multiple comparison of several proteins or the inclusion of a gap penalty factor as a function depending on the length of the gap.

The assessment of the significance of a given match value was also proposed: first, two sets of random sequences with the same composition of the original proteins are constructed; second, the maximum-match between pairs of these sequences is determined several times and is compared to the value obtained between real proteins; third, the match between one of the real proteins and several of the random sequences is also computed and evaluated. In all of the cases, the difference between the real match and the artificial ones should be statistically significant. Otherwise, the match between both proteins would be explained in part only by a similar composition.

Formulation and cost

The objective of the algorithm is to compute the pathway in the matrix M that according to a certain scoring schema is assigned the maximum value. The procedure to efficiently compute this value consists of two stages (see Figure 3.2):

① Each cell of the matrix $M(i,j)$ is assigned the corresponding value whether there is a match or a mismatch in this position (e.g. 1 for identities, void or 0 for mismatches).

② Beginning at the terminals of the sequences and proceeding toward the origins in the matrix, the value of the maximum-match starting at each cell $M(i,j)$ can be obtained by adding to its value, the maximum value from among all the cells which lie on a pathway to it. The pathways are negatively weighted with the value g according to the number of gaps they contain.

$$M(i,j) = M(i,j) + max \begin{cases} M(i+1,j+1) \\ M(i',j+1) + g \times (i'-i+1), & i+2 \leq i' \leq |A| \\ M(i+1,j') + g \times (j'-j+1), & j+2 \leq j' \leq |B|. \end{cases} \tag{3.4}$$

If $|A| = |B| = n$, then the cost of visiting each cell of the matrix is $O(n^2)$. Additionally, for each cell the best pathway among all of the possible ones in the previous row, in the previous column and in the diagonal is searched. The cost of accessing the values of the pathways in a given column or row is $O(n)$, while accessing the diagonal is constant $O(1)$. Therefore, the final cost of the **?** algorithm is $O(n^3)$.

Implementation

The implementation of the algorithm is shown in Figure 3.3. The matrix is processed following a systematic order. Both processing steps described above are integrated in a single one. For each pair of amino acids from both sequences represented by a cell $M(i,j)$ in the matrix , the optimal pathway starting there is constructed selecting the best pathway in the diagonal, and in the $i+1$ row and the $j+1$ column (here weighting according to the number of gaps) that have been previously computed.

The matrix P is used to record the cell from which the maximum pathway was selected. The retrievement of the solution, not shown here, consists on (1) searching the maximum value (cell x,y) both in the first row and in the first column and (2) using recursively the coordinates in $P(x,y)$, to construct the arrangement of both sequences until a cell at the last column or row is reached.

The Sellers algorithm (1974)

In the 1970s, most techniques used in taxonomic tree construction depended on the introduction of a measure of distance between sequences (**?**). The work on distances or metrics on protein sequences was essentially based on discovering what genetic mutations were required to change one sequence into another.

A metric space is a function $\rho : S \times S \to \mathcal{Z}^+$ on a generic set S, with the following properties:

Non-negative	$\forall a, b \in S : \rho(a, b) \geq 0$
Identity	$\forall a, b \in S : \rho(a, b) = 0 \Leftrightarrow a = b$
Reflexivity	$\forall a, b \in S : \rho(a, b) = \rho(b, a)$
Transitivity	$\forall a, b, c \in S : \rho(a, b) \leq \rho(a, c) + \rho(c, b)$.

? described the construction of an evolutionary tree, which assumes that evolutionary distance is a metric. The minimum distance $D(A, B)$ between two sequences A and B is defined as the smallest possible weighted sum of insertions, deletions, and substitutions which transforms one sequence into the other.

? showed that if a scoring function $d(a, b)$[3] forms a metric space over the underlying alphabet of symbols then the minimum distance function $D(A, B)$ forms a metric space over the set of finite sequences constructed with such an alphabet. In addition, he proportioned the dynamic programming recurrence to efficiently compute the minimum distance D between two sequences using several scoring functions. In fact, many comparison algorithms that use distance functions with a given weighting scheme provide an optimal alignment only if such a scheme is a metric (**?**).

Formulation and cost

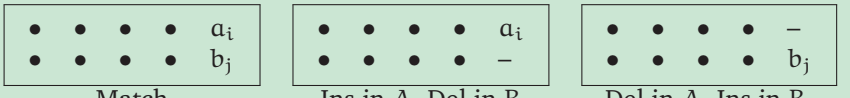
? generalized the algorithm to allow for various weighting schemes. Let a and b be two symbols. The simplest scheme d to score this match is defined as:

$$d(a, b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{if } a \neq b. \end{cases} \tag{3.5}$$

Using this scoring function d, the following recurrence calculates the optimal distance between two sequences $A = (a_1, a_2, \ldots a_m)$ and $B = (b_1, b_2, \ldots b_n)$, and provides the initial values as well:

$$\begin{aligned} D(i, j) = \quad & \min \begin{cases} D(i-1, j-1) + d(a_i, b_j) & \textit{Match} \\ D(i-1, j) + d(a_i, -) & \textit{Gap in B} \\ D(i, j-1) + d(-, b_j) & \textit{Gap in A} \end{cases}, \\ D(i, 0) = \quad & \textstyle\sum_{k=0}^{i} d(a_k, -), \\ D(0, j) = \quad & \textstyle\sum_{k=0}^{j} d(-, b_k). \end{aligned} \tag{3.6}$$

To avoid the exponential number of combinations to construct an alignment between two sequences, this dynamic programming recurrence decompose the problem in smaller alignments of prefixes of the original sequences. Thus, starting from the one-letter prefixes , the minimum distance of the alignment ending at the prefixes $A_{1,i}$ and $B_{1,j}$ can be calculated from the three different forms of finishing such an alignment:



Match Ins in A, Del in B Del in A, Ins in B.

```
Pre ≡ A, B: sequences; id,mis,gap ∈ ℤ

    (* Begin the series of sums from last row and column *)
    for i = |A| to 1 do
      for j = |B| to 1 do
 5:     (* Setting the identity or mismatch value for the cell *)
        if a_i = b_j then
          M(i,j) ← id;
        else
          M(i,j) ← mis;
10:     end if
        if i ≠ |A| and j ≠ |B| then
          (* Search the maximum-match pathway beginning here *)
          (* A. The maximum from diagonal *)
          max ← M(i+1,j+1);
15:       P(i,j) ← (i+1,j+1);
          (* B. The maximum value from previous column *)
          ngaps ← 1;
          for i' = i+2 to |A| do
            value ← M(i',j+1) + gap * ngaps;
20:         if value > max then
              max ← value;
              P(i,j) ← (i',j+1);
            end if
            ngaps ← ngaps + 1;
25:       end for
          (* C. The maximum value from previous row *)
          ngaps ← 1;
          for j' = j+2 to |B| do
            value ← M(i+1,j') + gap * ngaps;
30:         if value > max then
              max ← value;
              P(i,j) ← (i+1,j');
            end if
            ngaps ← ngaps + 1;
35:       end for
          (* The maximum-match pathway is formed *)
          M(i,j) ← M(i,j) + max;
        end if
      end for
40: end for
```

Figure 3.3 The ? algorithm.

If both sequences have the same length n, the cost of the **?** algorithm is $O(n^2)$ which is the time to visit

[3] Also known as a weighting scheme.

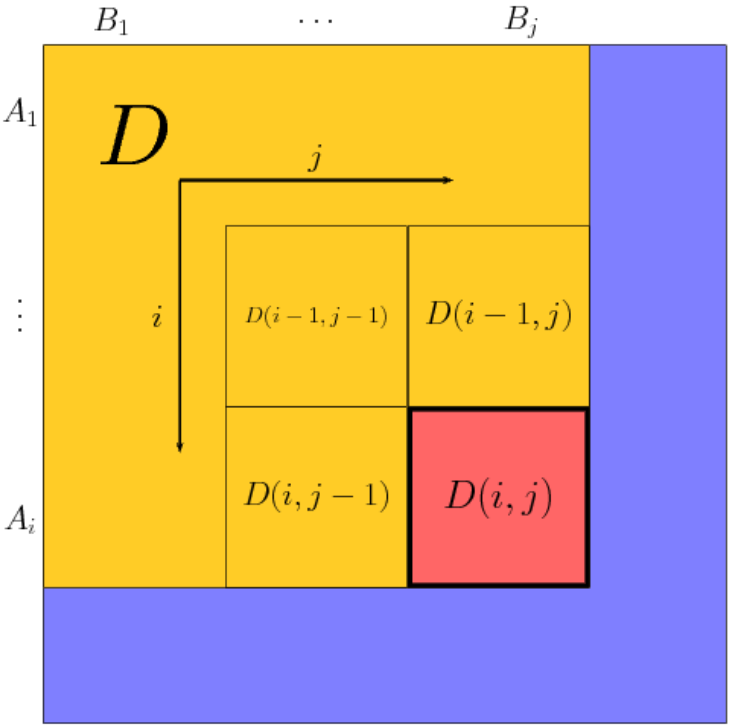


Figure 3.4 The dynamic programming matrix. In yellow, the part of the alignment matrix that has been computed. In blue, the part that must be still calculated. The cell $D(i,j)$ is the match currently in process.

all of the cells of the dynamic programming matrix (see Figure 3.4). For each cell, only three neighbours are consulted: in the diagonal, in the horizontal and in the vertical.

The procedure to trace-back the distance matrix, reconstructing the alignment was adapted from **?** by **?**. A second matrix of pointers is needed for recording from which direction was taken the value to update a given cell matrix.

Implementation

The **?** algorithm requires to fit the **?** $m \times n$ matrix in an artificial 0-column and 0-row to increase the initial distance when starting the alignment with gaps[4].

Then, the algorithm starts at $D(1,1)$ and the matrix is filled by rows (from top to bottom) and within a row by columns (from left to right). Thus, when a cell $D(i,j)$ is reached, its neigbours $D(i-1,j-1)$, $D(i-1,j)$ and $D(i,j-1)$ have been already calculated.

Contrarily to the **?** algorithm (in which the maximum match was searched in the last column and the last row), the minimum distance between both sequences will be saved at the end into the cell $D(m,n)$ because of the different initialization.

[4]There is an easy modification of the algorithm to permit not to punish this kind of gaps.

```
Pre ≡ A, B: sequences; d: metric on Σ

    (* Initialize the 0-column and the 0-row *)
    for i = 0 to |A| do
      D(i, 0) ← i × d(a_i, −);
 5: end for
    for j = 1 to |B| do
      D(0, j) ← j × d(b_j, −);
    end for
    (* Filling the matrix *)
10: for i = 1 to |A| do
      for j = 1 to |B| do
        (* A. Match *)
        min ← D(i − 1, j − 1) + d(a_i, b_j);
        P(i, j) ← (i − 1, j − 1);
15:     (* B. Gap in sequence B *)
        value ← D(i − 1, j) + d(a_i, −);
        if value < min then
          min ← value;
          P(i, j) ← (i − 1, j);
20:     end if
        (* C. Gap in sequence A *)
        value ← D(i, j − 1) + d(−, b_j);
        if value < min then
          min ← value;
25:       P(i, j) ← (i, j − 1);
        end if
        D(i, j) ← min;
      end for
    end for
```

Figure 3.5 The ? algorithm.

As in the case of the **?**, there is an auxiliary matrix P that saves the source of each calculation in a given cell to recursively reconstruct the alignment with such a distance.

A linear space algorithm: Hirschberg (1975)

In some occasions when aligning two sequences, the limiting factor is not the time but the space (memory). Any algorithm that solves the alignment of two sequences can not decrease the quadratic time cost unless any assumption is made over the length of the inputs. However, the quadratic cost in terms of space can be reduced to a linear cost.

? designed a divide and conquer algorithm to solve the LCS problem in linear space without increasing the asymptotic time cost. Later, **?** demonstrated how this technique could optimally deal with general sequence alignment problems.

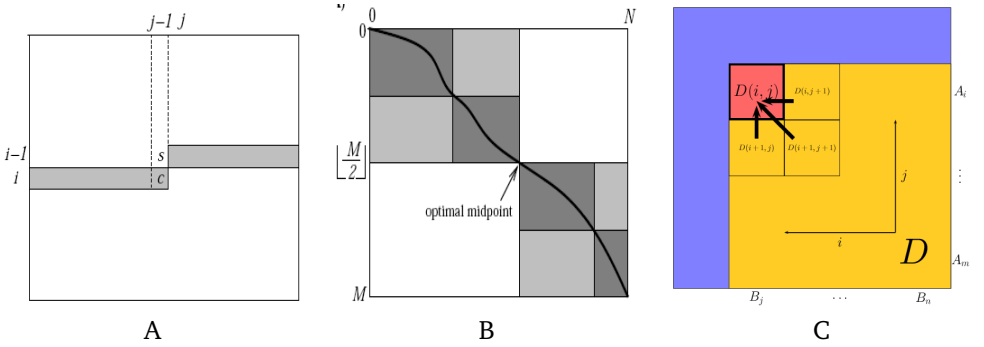


Figure 3.6 The ? linear space approach. (A) Using a single array to compute $D(i,j)$. (B) The divide and conquer strategy applied over the dynamic programming approach. (C) The backward propagation of values.

The key point of the algorithm is based on the fact that in the alignment between the sequences A and B, any element of A will be aligned either to a gap or another element in B. Thus, the problem of aligning both sequences can be expressed in terms of making this decision for a current element a_i, assuming the optimal alignments between the subsequences from A and B around this element are already computed.

Another important fact is the ability to compute the distance between two sequences in linear space. If the dynamic programming matrix is filled in from top to bottom (row by row), and fixing a row, from left to right (column by column), then the values in a row i depend only on the values stored at the previous row $i-1$ and on the values in the same row i. The other previous rows are therefore not necessary to obtain the final value $D(m,n)$ (**??**).

Furthermore, instead of using two arrays to represent the rows i and $i+1$, the computation can be performed in a single array D (see Figure 3.6 (A)), overwriting the old values on the left of the current column j. The equivalence between each cell $D(i,j)$ in the original dynamic programming matrix and the content of this unidimensional array D when the row i is being processed is:

$$\begin{aligned} D(k) &\approx D(i,k) \;\text{ when }\; k < j \;\text{ (current row, i)} \\ D(k) &\approx D(i-1,k) \;\text{ when }\; k \geq j \;\text{ (previous row, } i-1\text{).} \end{aligned} \tag{3.7}$$

Formulation and cost

In the optimal alignment between two sequences A and B, a given element a_i from A will be either matched to another element b_j from B or aligned to a gap between a certain b_j and b_{j+1}. Then, this optimal alignment can be decomposed in three parts:

① The optimal alignment between the elements from both sequences on the left (prefixes).

② The match between a_i with a certain b_j or a gap.

③ The optimal alignment between the elements from both sequences on the right (suffixes).

For a given i, the optimal point j can be unveiled with the application of the algorithm to compute only the distance between two sequences in linear space time. Such a solution provides the point in which the optimal alignment path will cross the i-row in the the dynamic programming matrix. As it is shown in Figure 3.6 (B), once the points i and j are established, the general problem is divided into two subproblems and recursively the same procedure is applied until reaching the base case (empty sequences).

The algorithm that computes only the distance between two sequences in linear space is in fact a method to provide the minimum distance between the first sequence and any of the prefixes of the second sequence. As the right parts are also aligned in the main procedure, a modification of such an algorithm is necessary to obtain the minimum distance between the first sequence and any of the suffixes of the second one.

In fact, the dynamic programming scheme is not restricted to construct the final alignment from alignments between prefixes of the input sequences. The same recurrence is appropriate for building it from alignments between suffixes of them. The procedure now begins in the position $D(|A|,|B|)$, and propagates the values from bottom to top, and from right to left (see Figure 3.6 (C)). The Equation 3.6 must be slightly modified to accommodate this backward propagation:

$$\begin{aligned} D(i,j) = & \quad \min \begin{cases} D(i+1,j+1) + d(a_i, b_j) & \textit{Match} \\ D(i+1,j) + d(a_i, -) & \textit{Gap in B} \\ D(i,j+1) + d(-, b_j) & \textit{Gap in A} \end{cases}, \\ D(i,|B|+1) = & \quad \textstyle\sum_{k=0}^{i} d(a_k, -), \\ D(|A|+1,j) = & \quad \textstyle\sum_{k=0}^{j} d(-, b_k). \end{aligned} \tag{3.8}$$

The cost of obtaining just the value $D(i,j)$ following the forward or the backward manner is again quadratic in terms of time. However, the cost in terms of space of this function is linear as only a single array is used in both cases.

To compute the cost of a recursive divide and conquer function, a different cost scheme must be applied. Let us consider n the length of both input sequences that are aligned. At the beginning, the routine performs some computations and then, there is an approximate $\frac{n}{2}$ reduction in the size of the input data for the subsequent two recursive calls. The cost $T(n)$ of computing such a recursive function can be expressed in terms of its children as:

$$T(n) = \begin{cases} g(n) & 0 \leq n < c \quad \textit{Base case} \\ aT(\frac{n}{c}) + bn^k & n \geq c \quad \textit{Recursive case} \end{cases}, \tag{3.9}$$

where a is the number of recursive calls, c is the size of the fragmentation and bn^k is the cost of the non-recurrent operations performed on each call. From the relationship between a and c^k, the corresponding cost function is inferred following the Master Theorem of recurrent equations (see **?**, Sections 4.3 and 4.4). In the **?** recurrence is easy to notice $g(n) \in O(n)$, $a = 2$, $b = 2$, $c = 2$ and $k = 2$. Therefore as $a < c^k$ or $2 < 2^2$, according to the Master Theorem $T(n) \in \Theta(n^k)$, that is $T(n) \in \Theta(n^2)$.

Nevertheless, the spatial cost of the **?** algorithm is linear. All of the computations on the theoretical dynamic programming matrix are performed over single rows implemented with unidimensional arrays.

```
Procedure ComputeOnlyDistanceForward

Pre ≡ A, B: sequences; d: metric on Σ
Post ≡ D: array (|B|+1);

    (* Simulating the initialization of the 0-row *)
    for j = 0 to |B| do
      D(j) ← j × d(−, b_j);
 5: end for
    for i = 1 to |A| do
      diag ← D(0);
      (* Simulating the initialization of the i-row *)
      D(0) ← i × d(a_i, −);
10:   for j = 1 to |B| do
        (* This cell will be the next diagonal *)
        temp ← D(j);
        (* A. Match (↖) *)
        min ← diag +d(a_i, b_j);
15:     (* B. Gap in sequence A (↑) *)
        value ← D(j) + d(a_i, −);
        if value < min then
          min ← value;
        end if
20:     (* C. Gap in sequence B (←) *)
        value ← D(j − 1) + d(−, b_j);
        if value < min then
          min ← value;
        end if
25:     D(j) ← min;
        (* Update diagonal *)
        diag ← temp
      end for
    end for
```

```
Procedure ComputeOnlyDistanceBackward

Pre ≡ A, B: sequences; d: metric on Σ
Post ≡ D: array (|B|+1);

    (* Simulating the initialization of the last row *)
    for j = |B| to 1 do
      D(j) ← j × d(−, b_j);
 5: end for
    for i = |A| − 1 to 1 do
      diag ← D(|B| + 1);
      (* Simulating the initialization of the i-row *)
      D(|B| + 1) ← i × d(a_i, −);
10:   for j = |B| to 1 do
        (* This cell will be the next diagonal *)
        temp ← D(j);
        (* A. Match (↖) *)
        min ← diag +d(a_i, b_j);
15:     (* B. Gap in sequence A (↓) *)
        value ← D(j) + d(a_i, −);
        if value < min then
          min ← value;
        end if
20:     (* C. Gap in sequence B (→) *)
        value ← D(j + 1) + d(−, b_j);
        if value < min then
          min ← value;
        end if
25:     D(j) ← min;
        (* Update diagonal *)
        diag ← temp
      end for
    end for
```

Figure 3.7 An algorithm to compute $D(i,j)$ in $O(n)$ space cost. (Left) The computation is done from $D(0,0)$ to $D(|A|,|B|)$. (Right) The computation is done from $D(|A|+1,|B|+1)$ to $D(1,1)$.

Implementation

Computing the value $D(i,j)$ in linear space

To implement the fusion of the previous row and the current one in a single array in a forward manner, the temporary variables *diag* and *temp* are necessary to save the values $D(i-1,j-1)$ –diagonal– and $D(i-1,j)$ –gap in A–, respectively. At the end of the forward computation, the array D will contain the same values as the last row of the bidimensional classic dynamic programming matrix, that is, the distance between the sequence A and any of the prefixes of the sequence B. In particular, the array position $D(|B|)$ will contain the distance between the sequences $|A|$ and $|B|$

The backward computation is symmetrical to the forward processing. The propagation of values starts now in the position $D(|B|+1)$, moving the values from right to left, and from bottom to top. At the end of the backward computation, the array D will contain the same values as the last row of the bidimensional reverse dynamic programming matrix, that is the distance between the sequence A and any of the suffixes of the sequence B. In particular, the array position $D(1)$ will contain the distance between the sequences $|A|$ and $|B|$.

The divide and conquer algorithm

The **?** linear space algorithm is a function *Alignment* that computes the position of a given symbol a_i from A in the optimal alignment (aligned to a gap or to a certain b_j from B) and then splits the general problem into two smaller subproblems (left and right halves of the corresponding sequences).

The initial call is $Alignment(A, B, d, 1, |A|, 1, |B|, 0, 0)$ where $(0, 0)$ are the boundaries (al_1,al_2) of the optimal alignment that is in construction. Additionally, a pair of arrays *alA, alB* will save the correspondence between the symbols from both sequences.

As we divide the problem into two minor parts, the base cases are the empty sequences (alignment of the rest of the symbols in one sequence with gaps in the other one). The general case selects a middle point or symbol a_i from A. Then, the distance between the prefix $A_{1,i-1}$ to all of the prefixes of the sequence B is computed by the routine *ComputeOnlyDistanceForward*. The routine *ComputeOnlyDistanceBackward* likewise calculates the same value between the suffix $A_{i+1,|A|}$ and all of the suffixes of the sequence B.

Now, a sweep shifting j along the whole row is perform to detect the point j in which the alignment constituted by the prefix of A and a given prefix of B, the symbol a_i and a gap or a symbol b_j, and the suffix of A and a given suffix of B is optimal. This operation is easily implemented by accessing with the proper indexes the arrays *prefDist* and *suffDist* that were filled in by the corresponding *ComputeOnlyDistance* functions.

Once the optimal j for the current a_i has been found, a recursive call to discover the part of the optimal alignment on the left of this symbol is launched. Then, the conquer step assigns the correct position to a_i aligned to a gap or a certain b_j in the alignment (arrays *alA,alB*). Finally, a second recursive call is performed to place correctly the right part of the optimal alignment.

The variables *posmin,vmin, typemin* save at each moment the minimum distance value in the loop along j and the position and the symbol to be aligned to a_i. The type of symbol is important to correctly split the sequence B at the divide step.

The Needleman and Wunsch algorithm revisited by Smith *et al.* (1981)

Although the denomination of **?** algorithm and **?** algorithm have survived throughout the years, the standard formulation in terms of distance and similarity methods that is widely known today was provided by **?**. In their work, they adapted the **?** method to a dynamic programming recurrence complementary to that introduced by **?** and presented an analysis of equivalence between both measures (see next section).

The similarity measure did not conserve the mathematical properties of the distance metrics. Nonetheless, this revisited version of the algorithm became very popular because it was easily extended to cope with the local alignment problem (see Section 3.4).

Formulation and cost

If matches are positively rewarded and gaps are punished negatively, the recurrence for computing the maximum similarity between two sequences A and B is:

$$
\begin{aligned}
S(i,j) = \quad & \max \begin{cases} S(i-1,j-1) + s(a_i, b_j) & \textit{Match} \\ S(i-1,j) + s(a_i, -) & \textit{Gap in B} \\ S(i,j-1) + s(-, b_j) & \textit{Gap in A} \end{cases}, \\
S(i,0) = \quad & \textstyle\sum_{k=0}^{i} s(a_k, -), \\
S(0,j) = \quad & \textstyle\sum_{k=0}^{j} s(-, b_k).
\end{aligned}
\tag{3.10}
$$

where the function $s(a_i, b_j)$ provides a positive or a negative value for a given match (mismatch) according to the aligned elements. If this is an alignment of proteins, the function s can be a popular amino acid substitution matrix with an additional penalty for aligning a symbol to a gap.

The cost of this revisited **?** algorithm is $O(n^2)$, being correct the same analysis explained in the **?** approach.

Implementation

This implementation is a symmetric translation from the implementation of the **?** algorithm. The same procedures to fill the matrix in and to retrieve the optimal alignment are performed.

Equivalence between distance and similarity: Smith *et al.* (1981)

From the formulation of the **?** similarity algorithm and the **?** distance algorithm a couple of relevant questions quickly arised: (1) When are both algorithms equivalent? (2) When do they provide the same set of optimal alignments? **?** stated that the two algorithms are defined to be equivalent if given the scoring scheme for one algorithm, there is a choice of a scoring scheme for the second algorithm such that the set of alignments achieving the maximum similarity is equal to the set of alignments achieving the minimum distance.

Given two sequences A and B, the optimal alignment $\mathcal{A}$ that maximizes $S(A,B)$ or minimizes $D(A,B)$ can be decomposed into two sections: the matched elements (λ_i) and the elements in one sequence that are aligned with gaps in the other one (Δ_k):

$$
\begin{array}{c|c}
s(a_i, b_j) = \alpha_k & d(a_i, b_j) = \beta_k \\
g(k) \geq 0 & g'(k) \geq 0
\end{array}
$$

$$
\begin{aligned}
\lambda_i \quad & \rightarrow \#\text{ of aligned symbols of type } i \\
\Delta_k \quad & \rightarrow \#\text{ of gaps of length } k
\end{aligned}
\tag{3.11}
$$

$$
S(A,B) = \max_{\mathcal{A}}\{\textstyle\sum_i \alpha_i\lambda_i - \sum_k g(k)\Delta_k\} \;\Big|\; D(A,B) = \min_{\mathcal{A}}\{\textstyle\sum_i \beta_i\lambda_i + \sum_k g'(k)\Delta_k\}.
$$

The following consideration that relates the length of the input sequences to the number of aligned symbols and gaps is essential for the next equations:

$$
|A| + |B| = 2\textstyle\sum_i \lambda_i + \sum_k \Delta_k. \tag{3.12}
$$

For instance, this equation applied on the alignment

$$\begin{array}{lcccccc} A: & A & A & T & T & C & A \\ & | & & & | & | & \\ B: & T & - & - & T & C & - \end{array}$$

with $|A| = 6$ and $B = |3|$ with three matches, one gap of two positions and one gap of one position, produces:

$$6 + 3 = 2 \times 3 + 1 \times 2 + 1 \times 1.$$

? showed that with a certain scoring model for both algorithms, the optimal alignments are equivalent. Let α_M be $\alpha_M = max_i \alpha_i$ (the maximum value of similarity). Then, the other scoring model must be defined as $\beta_i = \alpha_M - \alpha_i$. Intuitively, the higher the similarity, the lower the distance. Thus, maximum similarity (α_M) equals to minimum distance (0). The development of the Equation 3.11 produces:

$$\begin{aligned} S(A,B) &= max_{\mathcal{A}}\{\textstyle\sum_i \alpha_i\lambda_i - \sum_k g(k)\Delta_k\} \\ &\star \beta_i = \alpha_M - \alpha_i \star \\ &= max_{\mathcal{A}}\{\textstyle\sum_i (\alpha_M - \beta_i)\lambda_i - \sum_k g(k)\Delta_k\} \\ &= max_{\mathcal{A}}\{\alpha_M \textstyle\sum_i \lambda_i - \sum_i \beta_i\lambda_i - \sum_k g(k)\Delta_k\} \\ &\star |A| + |B| = 2\textstyle\sum_i \lambda_i + \sum_k \Delta_k \star \\ &= max_{\mathcal{A}}\{\alpha_M(\tfrac{|A|+|B|}{2} - \textstyle\sum_k \tfrac{k}{2}\Delta_k) - \sum_i \beta_i\lambda_i - \sum_k g(k)\Delta_k\} \\ &= max_{\mathcal{A}}\{\alpha_M(\tfrac{|A|+|B|}{2}) - \textstyle\sum_k \tfrac{\alpha_M k}{2}\Delta_k - \sum_i \beta_i\lambda_i - \sum_k g(k)\Delta_k\} \\ &= max_{\mathcal{A}}\{\alpha_M(\tfrac{|A|+|B|}{2}) - \textstyle\sum_i \beta_i\lambda_i - \sum_k (\tfrac{\alpha_M k}{2} + g(k))\Delta_k\} \\ &= \alpha_M(\tfrac{|A|+|B|}{2}) + max_{\mathcal{A}}\{-\textstyle\sum_i \beta_i\lambda_i - \sum_k (\tfrac{\alpha_M k}{2} + g(k))\Delta_k\} \\ &= \alpha_M(\tfrac{|A|+|B|}{2}) - min_{\mathcal{A}}\{\textstyle\sum_i \beta_i\lambda_i + \sum_k (\tfrac{\alpha_M k}{2} + g(k))\Delta_k\} \\ &= \alpha_M(\tfrac{|A|+|B|}{2}) - D(A,B). \end{aligned} \tag{3.13}$$

To sum up, the minimum distance $D(A,B)$ is equivalent to the maximum similarity $S(A,B)$ when the following scoring model for the distance scheme is employed:

$$\begin{aligned} \beta_i &= \alpha_M - \alpha_i \\ g'(k) &= \tfrac{\alpha_M k}{2} + g(k). \end{aligned} \tag{3.14}$$

Given the similarity scoring model $(s, g(k))$, the following distance scheme is therefore compatible:

$$s(a,b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{if } a = b \end{cases} \tag{3.15}$$

$$d(a,b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{if } a \neq b \end{cases} \tag{3.16}$$

where $\alpha_M = 1$, $\beta_i = 1 - \alpha_i$, $d(a, b) = 1 - s(a, b)$ and $g'(k) = \frac{k}{2} + g(k)$.

Obviously, not all the possible s functions will have a compatible d counterpart (see local alignment, Section 3.4).

The Sellers algorithm generalized by Waterman *et al.* (1976)

From an evolutionary point of view, a single mutation event involving a gap with k positions is more probable than the same number of distinct mutations of k isolated spaces. In the previous algorithms, gaps have been treated as another symbol producing simple mismatches. However, longer indels should not be weighted as the sum of single indels.

Let $g(k)$ an arbitrary function that determines the penalty for a gap of length k, in which the existence of any relationship between the penalty of a gap having k characters and a gap of $k+1$ is not assumed (general gap scoring model):

In **?** / **?**	A more realistic weighting scheme
$g(k) = kg(1)$	$g(k) \leq kg(1)$

? introduced a new metric. Let $\tau = \{T|T : S \rightarrow S\}$ be a set of transformations (including identity) applied over an input sequence. Every transformation has an associated weight w. Given two sequences A and B, a sum of weights $\sum_{i=1}^{k} w(T_i)$ can be computed for each sequence of transformations $T_1, T_2, \ldots T_k$ from τ such that $T_1 \circ T_2 \circ \ldots \circ T_k(A) = (B)$. The minimum sum of weights of such sequences of transformations can be viewed as the distance from A to B and a metric space is obtained [5].

τ can be employed with different sets of transformations and weights (**?**). Specifically, the authors defined a τ-metric which included longer deletions and insertions, and generalize the **?** algorithm for computing the new distance.

Formulation and cost

In the **?** algorithm, the optimal alignment between the prefixes $A_{1,i}$ and $B_{1,j}$ could contain a match between a_i and b_j or an alignment of one of them to a gap in the other sequence. In this new generalized gap model, an alignment of one of them to a gap of length k in the other sequence is also possible.

To deal with gaps that have different scores according to their lengths, given a cell $D(i, j)$ in the dynamic programming matrix, all of the possible gaps of $1..(i-1)$ symbols (scanning a column, fixing j) and all of the possible gaps of $1..(j-1)$ symbols (scanning a row, fixing i) must be evaluated (see Figure 3.10).

This modification also receives the name of block indel variation because there are now three classes of implicit blocks to establish the optimal alignment between two symbols: either a match between both or an alignment between a substring of symbols in one of the sequences to a block of gaps in the other.

[5] The weights associated to every class of transformation must be non-negative.

```
Procedure Alignment

Pre ≡ A, B: sequences; d: metric on Σ; i1, i2, j1, j2, al1, al2 in Z
Post ≡ alA: array (al1..al2), alB: array (al1..al2);

    if A_{i1,i2} = ∅ then
      (* Base case 1 *)
      for k = j1 to j2 do
 5:     alA(al1 + k) ← −;
        alB(al1 + k) ← B(j1 + k);
      end for
      al2 ← al1 + k;
    else if B_{j1,j2} = ∅ then
10:   (* Base case 2 *)
      for k = i1 to i2 do
        alA(al1 + k) ← A(i1 + k);
        alA(al1 + k) ← −;
      end for
15:   al2 ← al1 + k;
    else
      (* General case *)
      (* Select the point i *)
      i ← ⌊(i1+i2)/2⌋
20:   (* Compute the distance to the prefixes/suffixes of B *)
      prefDist ← ComputeOnlyDistanceForward(A_{i1,i}, B_{j1,j2}, d);
      suffDist ← ComputeOnlyDistanceBackward(A_{i+1,i2}, B_{j1,j2}, d);
      (* The column 0 *)
      posmin ← j1 − 1;
25:   typemin ← SPACE;
      vmin ← prefDist(j1 − 1) + d(ai, −)+ suffDist(j1 − 1);
      (* A sweep along the row i *)
      for j = j1 to j2 do
        (* Match *)
30:     value ← prefDist(j − 1) + d(ai, bj)+ suffDist(j + 1);
        if value < vmin then
          vmin ← value;
          posmin ← j;
          typemin ← SYMBOL;
35:     end if
        (* Gap *)
        value ← prefDist(j) + d(ai, −)+ suffDist(j + 1);
        if value < vmin then
          vmin ← value;
40:       posmin ← j;
          typemin ← SPACE;
        end if
      end for
      (* Divide and conquer with these values of i and j *)
45:   if typemin ← SPACE then
        Align(A,B,d,i1,i − 1,j1,posmin,al1,al_tmp);
        alA(al_tmp) ← A(i);
        alB(al_tmp) ← −;
        Align(A,B,d,i + 1,i2,posmin+1,j2,al_tmp,al2);
50:   else
        Align(A,B,d,i1,i − 1,j1,posmin−1,al1,al_tmp);
        alA(al_tmp) ← −;
        alB(al_tmp) ← B(posmin);
        Align(A,B,d,i + 1,i2,posmin+1,j2,al_tmp + 1,al2);
55:   end if
    end if
```

Figure 3.8 The ? linear space algorithm.

```
Pre ≡ A, B: sequences; s: substitution matrix

    (* Initialize the 0-column and the 0-row *)
    for i = 0 to |A| do
      S(i, 0) ← i × s(a_i, −);
 5: end for
    for j = 1 to |B| do
      S(0, j) ← j × s(b_j, −);
    end for
    (* Filling the matrix *)
10: for i = 1 to |A| do
      for j = 1 to |B| do
        (* A. Match *)
        max ← S(i − 1, j − 1) + s(a_i, b_j);
        P(i, j) ← (i − 1, j − 1);
15:     (* B. Gap in sequence B *)
        value ← S(i − 1, j) + s(a_i, −);
        if value > max then
          max ← value;
          P(i, j) ← (i − 1, j);
20:     end if
        (* C. Gap in sequence A *)
        value ← S(i, j − 1) + s(−, b_j);
        if value > max then
          max ← value;
25:       P(i, j) ← (i, j − 1);
        end if
        S(i, j) ← max;
      end for
    end for
```

Figure 3.9 The ? algorithm revisited.

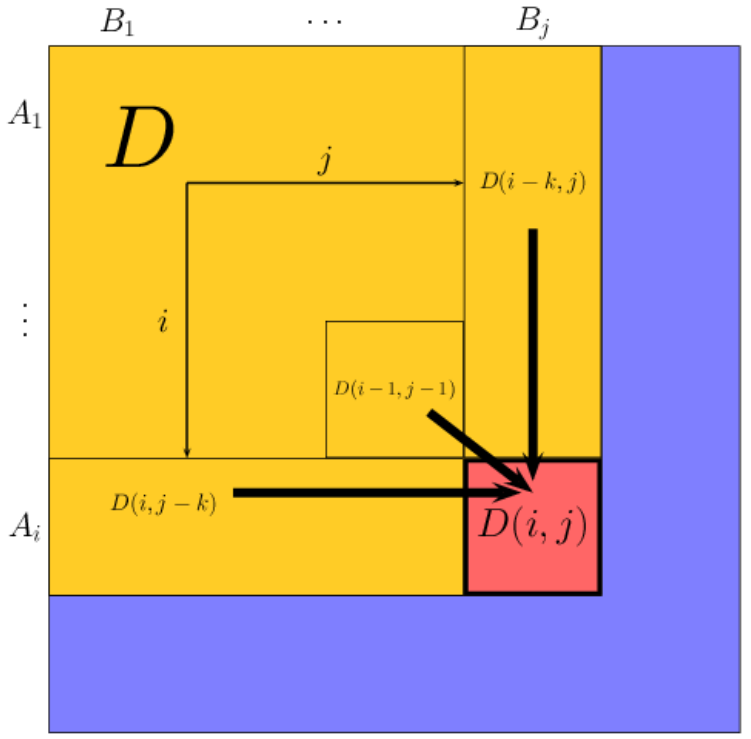


Figure 3.10 The generalized dynamic programming matrix. In yellow, the part of the alignment matrix that has been computed. In blue, the part that must be still calculated. The cell $D(i,j)$ is the match currently in process.

The following recurrence represents the generalization of the **?** algorithm by **?**:

$$
\begin{aligned}
D(i,j) = \quad & \min \begin{cases} D(i-1,j-1) + d(a_i, b_j) & \textit{Match} \\ \min_{1 \le k \le i}\{D(i-k,j) + g(k)\} & \textit{Gap of length } k \textit{ in } B \\ \min_{1 \le k \le j}\{D(i,j-k) + g(k)\} & \textit{Gap of length } k \textit{ in } A \end{cases}, \\
D(k,0) = \quad & g(k), \\
D(0,k) = \quad & g(k).
\end{aligned}
\tag{3.17}
$$

The algorithm must evaluate for each cell $D(i,j)$, all of the previously computed cells in that row and column. If the length of the sequences is m and n respectively, the cost of performing such an alignment with a general gap scoring model is therefore $O(mn(m+n))$, that is, $O(m^2n + mn^2)$ or $O(n^3)$ if both sequences have the same length.

Implementation

This algorithm requires the existence of an artificial 0-column and 0-row to compute the distance when starting the alignment with gaps. Then, the algorithm starts at $D(1,1)$ and the matrix is filled by rows (from

```
Pre ≡ A, B: sequences; d: metric on Σ; g(k): gap scoring function

    (* Initialize the 0-column and the 0-row *)
    for i = 0 to |A| do
      D(i, 0) ← g(i);
 5: end for
    for j = 1 to |B| do
      D(0, j) ← g(j);
    end for
    (* Filling the matrix *)
10: for i = 1 to |A| do
      for j = 1 to |B| do
        (* A. Match *)
        min ← D(i − 1, j − 1) + d(a_i, b_j);
        P(i, j) ← (i − 1, j − 1);
15:     (* B. Gap of length k in sequence B *)
        for k = 1 to i − 1 do
          value ← D(i − k, j) + g(k);
          if value < min then
            min ← value;
20:         P(i, j) ← (i − k, j);
          end if
        end for
        (* C. Gap of length k in sequence A *)
        for k = 1 to j − 1 do
25:       value ← D(i, j − k) + g(k);
          if value < min then
            min ← value;
            P(i, j) ← (i, j − k);
          end if
30:     end for
        D(i, j) ← min;
      end for
    end for
```

Figure 3.11 The ? algorithm generalized.

top to bottom) and within a row by columns (from left to right). For a given a cell $D(i,j)$ in the matrix, its neighbour in the diagonal $D(i-1,j-1)$ is evaluated (match) and additionally, all of the previous cells at that row and column must be separately visited to measure the contribution of $g(k)$ to their final value.

The minimum distance between both sequences will be saved at the end into the cell $D(m,n)$. The optimal alignment with such a distance value can be recursively retrieved from the auxiliary matrix P that saved the direction of the alignment for each cell.

The Waterman *et al.* algorithm revisited by Gotoh (1982)

Despite its cubic cost, the **?** algorithm provided a more realistic gap treatment model from the biological standpoint. Several posterior proposals were presented to reduce that cost by simplifying the gap model. **?** proposed a less general model called the affine gap scoring model in which the gap scoring function presents a linear schema based on a different penalty for opening a gap and for extending an existing one.

Let $g(k)$, the affine gap model to score a gap of k positions, then

$$\begin{aligned} g(k) &= \begin{cases} a, & \text{if } k = 1 \\ a + bk, & \text{if } k > 1 \end{cases} \quad \text{where } a, b \geq 0 \\ g(k+1) &= a + b(k+1) = a + bk + b = g(k) + b. \end{aligned} \tag{3.18}$$

With such a function, if $a > b$ then the first space in a gap of length k is more expensive than the rest of $k-1$ spaces that extend the gap. As the value $g(k+1)$ can be computed only using the previous value $g(k)$, there is no need to perform an exhaustive scanning of a given row and column for each pair i, j in the dynamic programming matrix.

Formulation and cost

? rewrote the general recurrence by **?**, introducing two additional functions E and F that substituted the two loops along the column and the row of a given cell $D(i,j)$ to evaluate the gaps of length k:

$$\begin{aligned} D(i,j) &= \min\{D(i-1,j-1) + d(a_i, b_j), E(i,j), F(i,j)\} \\ E(i,j) &= \min_{1 \leq k \leq i}\{D(i-k,j) + g(k)\} \\ F(i,j) &= \min_{1 \leq k \leq j}\{D(i,j-k) + g(k)\} \\ D(k,0) &= g(k), \\ D(0,k) &= g(k). \end{aligned} \tag{3.19}$$

Unfolding the value of $E(i,j)$ in the k and $k+1$ iterations, the combination between Equation 3.18 and Equation 3.19 produced the following result (**?**):

$$\begin{aligned} E(i,j) &= \min_{1 \leq k \leq i}\{D(i-k,j) + g(k)\} \\ &= \min\{D(i-1,j) + g(1), \quad \min_{2 \leq k \leq i}\{D(i-k,j) + g(k)\}\} \\ &= \min\{D(i-1,j) + a, \quad \min_{1 \leq k \leq i-1}\{D(i-(k+1),j) + g(k+1)\}\} \\ &= \min\{D(i-1,j) + a, \quad \min_{1 \leq k \leq i-1}\{D(i-1-k),j) + g(k)\} + b\} \\ &= \min\{D(i-1,j) + a, \quad E(i-1,j) + b\}. \end{aligned} \tag{3.20}$$

The same recursion is applied to the function F producing

$$F(i,j) = \min\{D(i,j-1) + a, F(i,j-1) + b\}. \tag{3.21}$$

Now, there are only three operations that must be performed to compute each $D(i,j)$: the match between both symbols, and the alignment of one of them to a gap (of any length) in the other sequence. The cost of using the affine gap scoring model is therefore $O(n^2)$, notably smaller than the cubic cost of the original general solution.

Implementation

To implement the functions E and F, two additional matrices are necessary. Then, the value $D(i,j)$ is selected (minimum) among the value of $D(i-1,j-1)$ (a match) and the values of $E(i,j)$ (a gap in the second sequence) and $F(i,j)$ (a gap in the first sequence).

The matrix P is used again to maintain the pathway associated to the optimal distance between any prefix of the input sequences.

Concave gap penalty functions: Waterman (1984)

In the affine gap penalty model, the same penalty is associated to the second space and to all of the next spaces in a gap. **?** studied the behaviour of the multiple indels scoring models in the coding region of the chicken α and β hemoglobin genes. They determined that a specific range of gap penalties was necessary to obtain correct alignments. Later, **?** formally introduced the concave gap functions.

In this scoring scheme, the gaps after the first one are not punished proportionally as in the case of the affine model. Once there is a gap, it must be biologically easier to incorporate more gaps. Let $g(k)$ the function that provides the penalty for a gap of length k, then:

$$g(k+1) - g(k) \leq g(k) - g(k-1). \tag{3.22}$$

The affine model arises directly when the equality is required. Strict inequality corresponds to those increasing functions with decreasing differences between consecutive gaps, also referred to as concave downward or simply concave. For instance, the function

$$g(k) = a + b\log(k) \text{ where } a, b \geq 0. \tag{3.23}$$

Let $f(k) = a + bk$ the affine gap penalty function, for a given length k the difference with the behavior of g is clear. For instance, if $k = 16$ then $f(16) = a + 16b$ whereas $g(16) = a + b\log_2(16) = a + 4b$, being less penalized this large gap in comparison to smaller gaps.

Formulation and cost

Lying between the general gap model, with a $O(n^3)$ cost, and the affine gap model, with a $O(n^2)$ cost, the concave gap problem has been proved to have an algorithm with a cost $O(n^2 \log n)$. **?** introduced the

concept and conjectured such a cost. Posteriorly, two independent groups arrived at different solutions with such a cost (**??**).

3.4 A short overview on local sequence alignment

Local alignments are usually more meaningful than global alignments because they only detect the patterns that are conserved in the sequences. The statistical significance of these patterns is usually evaluated. Uncommon degree of conservation of these segments in long sequences could be explained in terms of conservation of biological function.

Two alternative lines using dynamic programming approaches were proposed to rigorously detect such fragments: algorithms based on similarity and algorithms based on distance metrics. Traditionally, similarity schemes have shown to be easier to be implemented whereas distance measures are more complex to be adapted to this problem. Additional works about pattern discovery and multiple local alignments are provided in Chapter 4.

The Smith and Waterman algorithm (1981)

In a short communication, **?** published a slight modification of the **?** algorithm revisited by **?** to deal with local alignments. The main objective was to find the pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology).

The key point is to stop the traceback that starts from the cell having the maximum similarity whenever a negative similarity zone is detected. The score function s must therefore include negative values for mismatches to provide optimal alignments with this strategy.

Posterior refinements by **?** allow to report the second best path disjoint from the first one, the third best and so on. Essentially, the positions of the visited previous maximum paths are marked up and a new recomputation of some parts of the matrix is done to repeat the traceback.

Formulation and cost

In this formulation, a cell $S(i,j)$ of the dynamic programming matrix whose value after evaluating its neighbours is negative must be automatically set to 0 (the value for representing the lack of similarity of any local alignment ending at this cell). In fact, all of the positions in the matrix with a 0 are candidates to become the left boundary of the optimal local alignment between two sequences A and B.

The Equation 3.10 is just slightly modified to accommodate this concept:

$$
\begin{aligned}
S(i,j) = \quad & \max \begin{cases} S(i-1,j-1) + s(a_i, b_j) & \textit{Match} \\ S(i-1,j) + s(a_i, -) & \textit{Gap in } B \\ S(i,j-1) + s(-, b_j) & \textit{Gap in } A \\ 0 & \textit{Segment termination} \end{cases}, \\
S(i,0) = \quad & 0, \\
S(0,j) = \quad & 0.
\end{aligned}
\tag{3.24}
$$

As long as the scoring function $s(a,b)$ with $a \neq b$ (mismatch) returns negative values, the similarity of every path in the matrix will increase and decrease according to the associated alignment. Once the matrix has been completed, the cell having the highest value will be the right boundary of the optimal local alignment. From this point, the rest of the maximum similarity segment must be retrieved going back until a 0 is reached.

The natural generalization to support multiple insertions/deletions ($g(k)$) is naturally derived:

$$\begin{aligned} S(i,j) = \quad & \max \begin{cases} S(i-1,j-1) + s(a_i,b_j) & \textit{Match} \\ \max_{1 \leq k \leq i}\{S(i-k,j) + g(k)\} & \textit{Gap of length } k \textit{ in } B \\ \max_{1 \leq k \leq j}\{S(i,j-k) + g(k)\} & \textit{Gap of length } k \textit{ in } A \\ 0 & \textit{Segment termination} \end{cases}, \\ S(i,0) = \quad & 0, \\ S(0,j) = \quad & 0. \end{aligned} \tag{3.25}$$

Reduction to $O(n^2)$ can be achieved applying the **?** results as in the global alignment case. The time cost function of the versions above is the same as in their global counterparts as no additional operations are needed.

Implementation

In contrast to the global alignment algorithm, the initialization procedure reset to 0 the 0-row and the 0-column. In this implementation 0 means termination of current segment at the traceback process. The matrix P is used again to save the optimal pathway of the segment maximizing similarity ending at each position of the matrix S.

To retrieve such a segment, the position $S(i,j)$ which contains the maximum value in the matrix is found. Then, an ordinary traceback in P must be performed, reconstructing this local alignment until a cell whose value is 0 is reached, terminating then.

Distance-based scoring schemes

As it has been shown in Section 3.3, **?** determined the following relationship between a metric distance $D(A,B)$ and a homology function $S(A,B)$:

$$S(A,B) + D(A,B) = \alpha_M \tfrac{(m+n)}{2}, \tag{3.26}$$

where α_M is the maximum score for a match, and m and n are the lengths of the respective sequences A and B. From this, it might seem that the problem of finding segments of maximum similarity can be simply reformulated into a problem of finding segments of minimum distance. However, several differences between both measures prevent the establishment of such an equivalence:

- The maximum similarity is a positive number that depends on the aligned segments. On the contrary, the minimum distance is always 0.

- The similarity scoring scheme typically has a negative reward for mismatches and gaps and a positive reward for matches. However, the distance metric has no positive reward for matches: the extension of an alignment with a minimum distance d can only receive a score equal or worse than the original one (continuously growing function).

- In the similarity model, during the traceback a local alignment starting at the cell having the maximum value $S(i,j)$ is extended. Then, 0 is employed as a limit of such an extension. In the distance model, there is not a simple minimum value $D(i,j)$ in the matrix to start the traceback because smaller segments would be better by definition. Furthermore, there is not here an equivalent of the 0 in the similarity model during the traceback procedure.

To overcome some of these limitations, **?** considered to include the length of the segments in the scoring scheme as a way to favor longer alignments against shorter alignments with distance 0. The mismatch density of an alignment $\mathcal{A}$ between two segments is defined as the ratio of the minimum distance D between both sequences and the length L of the alignment. In addition, only those alignments with a mismatch density below a certain positive threshold R must be reported:

$$\frac{D(\mathcal{A})}{L(\mathcal{A})} \leq R. \tag{3.27}$$

Essentially, the segment maximizing the similarity should be equivalent to a segment starting at $D(i_0, j_0)$ and ending at $D(i,j)$ with $i_0 < i$ and $j_0 < j$ such that the difference $\Delta D = D(i,j) - D(i_0, j_0)$ is the minimum taking into account the length of such an alignment.

? also transformed this distance scheme into a similarity scheme that must be maximized, with the following manipulations:

$$\frac{D(\mathcal{A})}{L(\mathcal{A})} \leq R \equiv D(\mathcal{A}) \leq RL(\mathcal{A}) \equiv RL(\mathcal{A}) - D(\mathcal{A}) \geq 0. \tag{3.28}$$

Formulation and cost

First approaches

Previously to **?**, **?** approached the problem with an algorithm to determine the segments S and T such that for any aligned pair (S', T') in a small neigbourhood, $D(S,T) \leq D(S', T')$. Obviously, $D(S,T)$ was guaranteed to be only a relative minimum in such a set of alignments. Therefore, the procedure provided many alignments like this that needed further screening.

Later, **?** used the mismatch density concept to propose an algorithm in two steps. The solution is better understood if alignments are represented by paths in a lattice of points:

① Use the **?** global alignment algorithm to fill in the matrix D_F (minimize distance). This formulation computes the values from left to right and from top to bottom. This form corresponds to obtain the optimal alignment using the increasing prefixes of the sequences (forward graph).

② The same algorithm can be formulated in terms of suffixes of the input sequences (see the explanation about the **?** algorithm). Then, use such an algorithm over the same sequences to fill in the matrix D_B (backward graph).

③ Report those paths that were common in D_F and D_B.

This solution limited the number of paths but there is not a clear procedure to show that these are optimal. The cost of the algorithm is clearly $O(n^2)$.

Multi-sweep algorithm by Sellers (1984)

? described a more rigorous extension of the **?** algorithm in which several iterations over a single matrix are necessary to remove the edges of the paths that are not supported in the forward o backward computations.

Given a positive constant R, the algorithm produces all paths P such that:

① All prefixes of P have mismatch density less than R.

② All suffixes of P have mismatch density less than R.

③ The path P is locally maximal. The paths meeting the two previous conditions that intersect with P have a lowest score.

The algorithm starts with a matrix G_0 in which every possible alignment of the two given sequences is represented. First, the forward procedure removes all edges of the paths not being part of any alignment[6], creating the matrix G_1. Second, in a backward computation, all edges from G_1 not meeting the alignments of the suffixes are also erased to form the matrix G_2. Then, alternating forward and backward computations are performed over G_i, removing edges of the paths at each stage until no variation is observed. At the end, all of the disjoints paths present in the matrix are reported as local alignments or segments minimizing the mismatch density criterion.

No more than $O(n)$ sweeps are ever required to converge (**?**). As every forward or backward operation takes $O(n^2)$ time, the final cost of the approach by **?** is $O(n^3)$, notably higher than the $O(n^2)$ cost of the simple **?** design.

Databases searches

The information available at the sequence databases is useful to infer the function of similar sequences. Anonymous sequences can be aligned to other sequences whose function, structure or biochemical activity is known. As explained in Chapter 2, the size of such databases grows exponentially since the very first days of computational sequence analysis.

It is important to mention that from now on the term database simply refers to a large collection of sequences. It does not imply any extra capabilities of fast access, data sharing, and so on, commonly found in standard database management systems.

[6] In the dynamic programming recurrence, each edge corresponds to a decision in the optimization step.

Ordinary alignment algorithms based on dynamic programming are very inefficient to search large collections of sequence because of their quadratic time cost. Novel methods based on heuristics have been employed to reduce in several orders of magnitude the time to align two sequences, providing near optimal results. The search on a database for sequences that are similar to a query sequence usually performs hundreds of thousands of such alignments.

This search typically provides a list of sequences with which the query sequence can be aligned better, using certain quality score function. These results can be expanded using each sequence found before to find more distant relatives of the initial sequences.

To speed the search, the sequences of the database are usually preprocessed to store computations about their content (usually word distribution) that will be used during the future searching operations.

FASTA

The FAST family of algorithms is a group of heuristic methods for string comparison, specially to compare a query sequence with each sequence on a database. (**??**). The FASTA program that is included on such a package is entirely based on the following assumption: good local alignments are likely to contain exact matching subsequences. The FASTA strategy is therefore to locate firstly the segments of both sequences richer in exact matches and secondly, try to reconstruct the final alignment using these specific regions.

The FASTA processing is divided into four main steps (see Figure 3.14) that are repeated to compare the query sequence to each sequence in the database:

① Detection of regions of identity. Determine the words of length k (*k-tuples*) that are common to both sequences. The offset of an exact word match between a substring s starting at position x and a substring t starting at position y is defined as the difference $x - y$. Matches that are located in the same diagonal of the dotplot comparison have the same offset value (see (A) in Figure 3.14). An array addressed by the offsets is used to locate those diagonals with more exact matches.

During the preprocessing of each sequence in the database, a hash table is used to store where each word of length k is appearing along such a sequence (**?**). Then, the query sequence is scanned and each *k-tuple* in it is looked up in the hash table. For all common occurrences, the entry of the corresponding offset is incremented.

Next, each offset is analyzed to merge those exact matches in the same diagonal that are in close proximity (without introducing gaps, including intervening sequence). These merged regions do not contain any insertions or deletions because they are derived from a single diagonal. The score of these diagonal regions is the sum of the exact matches scores combined with a penalty that increases with the distance among them. According to this scoring scheme, the 10 best diagonal regions are selected to constitute the future local alignment (see (B) in Figure 3.14).

② Re-scoring. The 10 best diagonals are evaluated again using an amino acid (or nucleotide) substitution matrix to allow conservative replacements and exact matches shorter than k to contribute to the similarity score. The diagonal region with maximal score is identified (highest scoring initial region). Those regions whose score is below a given threshold are discarded (see (C) in Figure 3.14).

③ Optimal alignment of diagonal regions. The regions from compatible diagonals are combined following certain rules. The segments that are close to each other (not in the same diagonal) can be part of an alignment whose score is a function of a joining penalty (moving from one diagonal to other involves

gap introduction), their scores and their location. The optimal alignment initial region is a combination of compatible regions with maximal score. This score is a reference to rank the library of sequences according to their similarity to the query (see (D) in Figure 3.14).

④ The highest scoring library sequences are finally aligned with a modification of the **?** and **?** algorithms. Using dynamic programming, all possible alignments of the query and each sequence in the database that fall within a band centered around the highest scoring initial region are considered.

BLAST

As FASTA, the BLAST family programs (**??**) are able to achieve a substantial gain in terms of speed by searching first for common words or k-tuples in the query and in each database sequence. However, FASTA searches for all possible words of the same length whereas BLAST limits the search to those that are the most significant by integrating a substitution matrix in this step.

The central concept of the BLAST strategy is the neighbourhood of a sequence. The T-neighbourhood of a word w is the set of all sequences of the same length that align to w with score better than T. Such an alignment is gapless and the similarity score is the sum of the similarity values for each pair of aligned residues. Thus, searching a match between a given word in the query and other word in a sequence of the database is equivalent to searching a match between a neighbour of the original word in the query with a score greater than T and the same word in the other sequence. BLAST will only seek in the database for those significant words that would form with w a pair with a score of at least T, if any.

As a substitution matrix is used to score this alignment, the words where conservative substitutions have been introduced can also obtain a high score because the matches with them also may be biologically informative. In addition, different amino acid identities are not scored in the same manner: for instance, the alignment between a query word composed by very common amino acids and itself might not achieve a score better than T, and therefore it would not be included in the search process [7].

Whenever one of these significant words is found in one entry of the database, the respective word w in the query and the detected neighbour are aligned and form the seed of a segment pair that will be enlarged later. If the extended segment pair is assigned a score better than S, such a sequence is reported to be similar to the query.

The BLAST pipeline is constituted by these steps (see Figure 3.15) that are repeated to compare the query sequence to each sequence in the database:

① The query sequence is filtered to remove low-complexity regions (repeats) that can distort the word search (optional) to produce significant alignments.

② Generate the T-neighbourhood of every word of length k in the query sequence. Given a word w, the matches between any other combination of k amino acids and w are evaluated with a substitution matrix. For instance, if $k = 3$, there are 8000 possible words to align with w. The neighbours are ranked according to the score of this alignment. A deterministic finite automaton is constructed to recognize the language of the high-scoring neighbours (the most significant ones).

③ See if any sequence in the database contains one of these strings with the automaton constructed before (a match).

[7] BLAST allows the user to force the inclusion of the original words in the following steps.

④ Every match is used as a seed to find a locally maximal segment pair containing that hit, also called a maximal segment pair (MSP). The alignment between both words in their respective sequences is extended in each direction along the respective sequences, continuing the extension as long as the score does not fall more than a drop-off threshold. Such a score is a cumulative value resulting from evaluating with a substitution matrix the matches, mismatches and gaps of the alignment.

⑤ BLAST reports the database sequences with MSPs above a certain threshold S (the high-scoring segment pairs or HSPs). Such significant value is computed for each database according to the size of the query and the database, being unlikely to find a random sequence that achieves a score better than S when compared with the query (**?**).

The procedure is heuristic: only word pairs with a score above the threshold T can be the core of local similarity regions. Therefore, a segment pair of score better than S that does not contain any subsequence of length k with a score greater than T will not be detected. In addition, the selection of the parameters is not trivial: this method is feasible in practice only when the values of k, T and S are carefully chosen (**??**).

3.5 A short overview on multiple sequence alignment

From a simplistic point of view, a multiple sequence alignment (MSA) is a rectangular array of sequences optimally arranged to obtain the greatest number of similar characters on each column of the alignment. From a evolutionary perspective, however, the alignment of multiple sequences is intimately related to the study of molecular evolution. For example, the number and the class of changes in the residues of a MSA may be used to develop a preliminar phylogenetic analysis. Each column in the alignment of a set of sequences may predict the mutations that occurred at one site during the evolution of such a sequence family, revealing which positions in the sequences were conserved and which diverged from a common ancestor sequence.

The natural extension of the pairwise dynamic programming recurrence produces a multidimensional representation of the similarity matrix, being not possible to be implemented in practice (see the example for three sequences in Figure 3.16 (A)). Because of its $O(n^k)$ cost, where k is the number of sequences and n is the length of them (**?**)), several approaches have tried to circumvent such a problem by introducing some heuristic functions.

? developed a method assuming that the optimal MSA can be constructed from the best pairwise alignments between each pair of sequences (the projections). Thus, each optimal pairwise alignment defines a set of spatial positions within which the optimal MSA is supposed to be when projected on such a plane (see Figure 3.16 (B) and (C)).

The generalization to multiple alignment also induced a problem in the dimension of the substitution matrices and in the form of scoring an alignment in general. Let k be the number of aligned characters in a column of a MSA. In principle, $2^k - 1$ combinations with such elements are possible, but a substitution matrix of such dimensions would be absolutely unfeasible. The ordinary approach when scoring a MSA is usually the sum of pairs (the SP-score) that weights the n^2 combinations between two elements in the same column with a normal substitution matrix to provide a final score.

The hierarchical or clustering method called progressive alignment rapidly became popular because of its simplicity and biological feasibility (**?**). This strategy initially selects the best pairwise alignment and progressively incorporates the rest of sequences to this alignment. However, this dependence on the first alignment produces somehow a loss of flexibility in the rest of the subsequent alignments as most of the conserved positions in such an alignment are preserved throughout the process. The order of sequence

selection relies on the creation of a phylogenetic tree that guides the process to create the MSA. There are several well-known techniques to infer the best tree for a set of sequences. Distance based methods are based on minimizing the number of global changes between each pair of input sequences. The neighbour-joining algorithm (**?**) is a distance based method that first joins the clusters of sequences that are close to each other and apart from the rest, minimizing the sum of the branch lengths in the final tree.

The program CLUSTALW (**?**) incorporates a number of improvements to the progressive alignment implementation. In an initial round, all of the pairwise alignments are performed to calculate a distance matrix in $O(k^2n^2)$. A guide tree is constructed from this matrix using the neighbour-joining method. An initial alignment starting with the two most related sequences is then constructed. Finally, the sequences are gradually aligned according to the branching order in the guide tree (see the complete process in Figure 3.17).

During the construction of the tree, CLUSTALW assigns weights to the sequences to correct unfair sampling across all evolutionary distances in the data set. Highly divergent sequences without close relatives receive high weights. For instance, the weight 0.221 for the Hbb_Human gene in Figure 3.17 is calculated in this form:

$$0.221 = 0.081 + \frac{0.226}{2} + \frac{0.061}{4} + \frac{0.015}{5} + \frac{0.062}{6}$$

In addition, different substitution matrices are used on every stage of the alignment. Position-specific gap penalties that depend on several factors such as the existence of other gaps, the type of residues or the length of the sequences are also used during the alignment.

The dynamic programming recurrence must be now adapted to allow the alignment between two profiles or clusters of sequences that have been previously aligned. The score of the alignment between a column in a first alignment and a column in a second alignment is the average of all of the pairwise substitution matrix scores from the residues in the two sets of sequences multiplied by the weight of the sequences. Let C_i and C_j be two multiple alignments:

$$C_i = \begin{cases} x_1^1 \dots x_1^{l_i} \\ \vdots \quad \vdots \\ x_{|i|}^1 \dots x_{|i|}^{l_i} \end{cases} \qquad C_j = \begin{cases} y_1^1 \dots y_1^{l_j} \\ \vdots \quad \vdots \\ y_{|j|}^1 \dots y_{|j|}^{l_j} \end{cases} \tag{3.29}$$

Let $p \in C_i, q \in C_j$, two columns of the previous alignments. The score $S(C_i^p, C_j^q)$ of the alignment between both columns is computed as:

$$S(C_i^p, C_j^q) = \frac{\sum_{r=1}^{|i|} \sum_{s=1}^{|j|} w_r \cdot w_s \cdot M(x_r^p, y_s^q)}{|i||j|} \tag{3.30}$$

All of the methods above produce a global multiple sequence alignment. Local alignment of several sequences is intimately related to motif finding techniques, all of them heuristics. In Chapter 4, there is a brief overview about several pattern discovery methods.

3.6 Map alignments

Restriction enzymes and genomic maps

The DNA molecules in a cell can be randomly broken into small pieces by mechanical forces. However, the probability of randomly breaking a molecule to produce a fragment that contains a gene is null. Restriction nucleases, which can be purified from bacteria, are enzymes that cut the DNA double helix at specific sites defined by the local nucleotide sequence, producing DNA fragments of defined sizes (**?**). In fact, every nuclease recognizes a specific sequence of four to eight nucleotides (see Figure 3.18 for examples).

Different species of bacteria make restriction nucleases with different sequence specificities. More than 100 nucleases are now available commercially. It is relatively simple to find a restriction nuclease that create a DNA fragment including a particular gene (**?**).

After treatment with a combination of several restriction nucleases, a restriction map of a particular genetic region can be constructed showing the location of each restriction site in relation to the neighbour sites (see Figure 3.19 for an example of map comparison). The sites thus act as genetic markers and the map reflects their arrangement in the region. This arrangement allow the comparison of the same region of DNA in different species without having to determine the nucleotide sequence in detail (**?**). Indeed, mutations at a single letter of a sequence of DNA can cause the appearance or disappearance of a restriction site.

```
Pre ≡ A, B: sequences; d: metric on Σ; g(k) = a + bk;

    (* Initialize the 0-column and the 0-row *)
    for i = 0 to |A| do
      D(i, 0) ← g(i);
 5:   E(i, 0) ← g(i);
      F(i, 0) ← g(i);
    end for
    for j = 1 to |B| do
      D(0, j) ← g(j);
10:   E(0, j) ← g(j);
      F(0, j) ← g(j);
    end for
    (* Filling the matrix *)
    for i = 1 to |A| do
15:   for j = 1 to |B| do
        (* A. Update the E matrix *)
        min ← D(i − 1, j) + a;
        value ← E(i − 1, j) + b;
        if value < min then
20:       min ← value;
        end if
        E(i, j) ← min;
        (* B. Update the F matrix *)
        min ← D(i, j − 1) + a;
25:     value ← F(i, j − 1) + b;
        if value < min then
          min ← value;
        end if
        F(i, j) ← min;
30:     (* C. Minimum between Match, E and F *)
        min ← D(i − 1, j − 1) + d(a_i, b_j);
        P(i, j) ← D(i − 1, j − 1);
        if E(i, j) < min then
          min ← E(i, j);
35:       P(i, j) ← E(i, j);
        end if
        if F(i, j) < min then
          min ← F(i, j);
          P(i, j) ← F(i, j);
40:     end if
        D(i, j) ← min;
      end for
    end for
```

Figure 3.12 The ? algorithm.

```
Pre ≡ A, B: sequences; s: substitution matrix

    (* Initialize the 0-column and the 0-row *)
    for i = 0 to |A| do
      S(i, 0) ← 0;
 5: end for
    for j = 1 to |B| do
      S(0, j) ← 0;
    end for
    (* Filling the matrix *)
10: for i = 1 to |A| do
      for j = 1 to |B| do
        (* A. Segment termination *)
        max ← 0;
        P(i, j) ← (0, 0);
15:     (* B. Match *)
        value ← S(i − 1, j − 1) + s(a_i, b_j);
        if value > max then
          max ← value;
          P(i, j) ← (i − 1, j − 1);
20:     end if
        (* C. Gap in sequence B *)
        value ← S(i − 1, j) + s(a_i, −);
        if value > max then
          max ← value;
25:       P(i, j) ← (i − 1, j);
        end if
        (* D. Gap in sequence A *)
        value ← S(i, j − 1) + s(−, b_j);
        if value > max then
30:       max ← value;
          P(i, j) ← (i, j − 1);
        end if
        S(i, j) ← max;
      end for
35: end for
```

Figure 3.13 The ? algorithm.

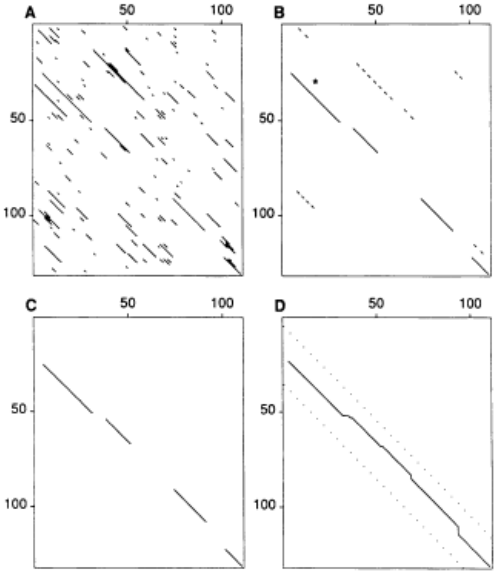


Figure 3.14 Identification of sequence similarities by FASTA. Adapted from **?**.

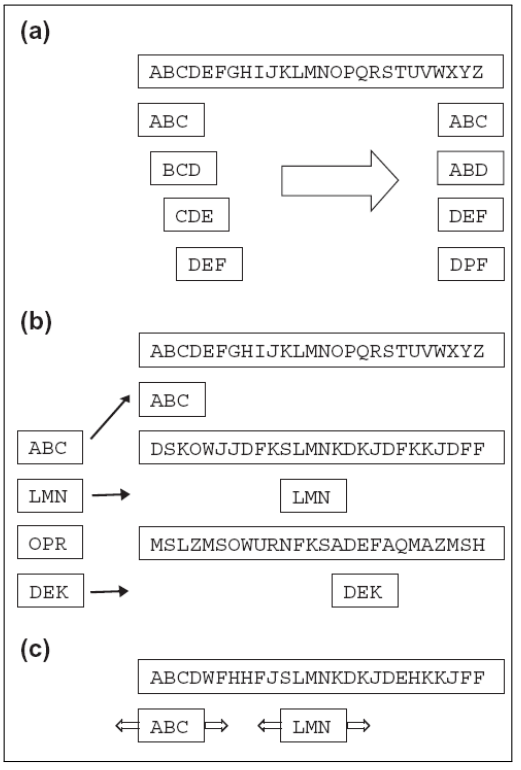


Figure 3.15 BLAST processing. Adapted from **?**.

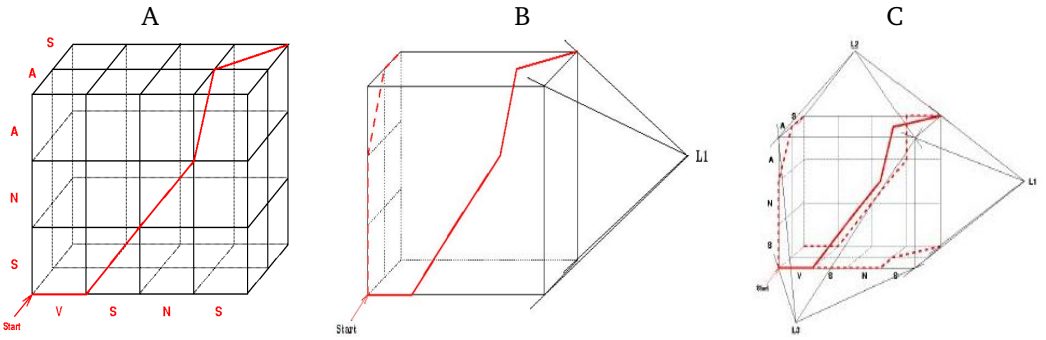


Figure 3.16 Generalized MSA dynamic programming matrix. (A) The $O(n^2)$ matrix is generalized into a $O(n^k)$ matrix in a multiple sequence alignment. (B) The projection of the optimal MSA into one of the pairwise alignments (**?**). (C) The optimal MSA alignment projected into all of the pairwise alignments (**?**).

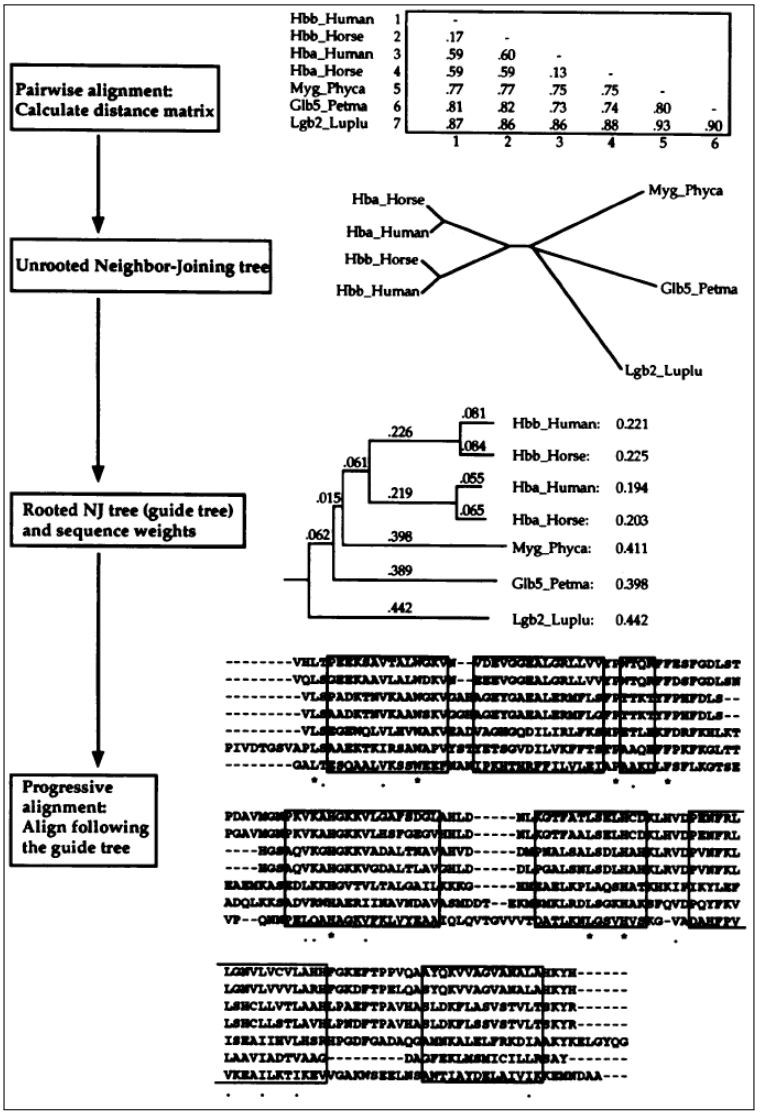


Figure 3.17 The basic CLUSTALW progressive alignment procedure. Adapted from ?.

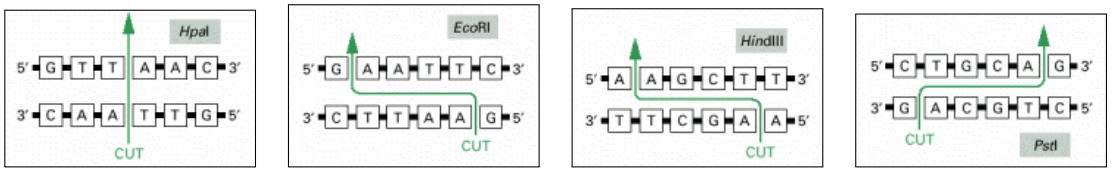


Figure 3.18 The DNA nucleotide sequences recognized by four widely used restriction nucleases. Adapted from **?**.

Common problems involving restriction maps are:

- Prior to genomic sequencing projects, to organize genomic DNA, one approach was to make restriction maps of relatively small pieces to utilize these maps later to determine overlap of pieces and thus construct a map that includes larger parts of the genome.
- The fragment lengths from a digestion of a DNA sequence can be measured after using two enzymes separately, or by both applied together. The problem of determining the positions of the cuts from fragment length data is known as the Double Digest Problem (**?**).

A more general definition can be considered. Genomic mapping is the process of determining where an object of biological interest (e.g. a marker, a gene, a genomic variation, or a disease predisposition locus) lies within a defined genomic sequence. Such a map therefore describes biological attributes of each genomic position (see **?** for a comprehensive introduction about mapping concepts).

Map alignments

Given a sequence S of m symbols, a site $a_i = (r_i, p_i)$ is an element of a certain type r_i mapped on a certain position p_i relative to the origin of S. A map A is then defined as an ordered set of n sites such that:

$$A = a_1 a_2 \dots a_n \text{ where } \forall i : 1 \leq i \leq n : a_i = (r_i, p_i), r_i \in \Sigma_{sites}, 1 \leq p_i \leq p_{i+1} \leq m$$

? first defined the notion of map comparison using alignments and developed an algorithm that handles the distances between the sites as well as the linear sequence of sites. If intersite distances were ignored, then the **?** algorithm could be immediately applied to align two maps (**?**).

Let $A = a_1 a_2 \dots a_m$ and $B = b_1 b_2 \dots b_n$ be to maps of m and n sites respectively with $a_i = (r_i, p_i)$ and $b_j = (s_j, q_j)$. An alignment of A and B is a sequence of ordered matching pairs of sites $(a_{i_1}, b_{j_1})(a_{i_2}, b_{j_2}) \dots (a_{i_T}, b_{j_T})$ such that:

① $(a_i, b_j) \in C$ if and only if $r_i = s_j$ (that is, two elements are aligned if and only if they correspond to the same site).

② if $(a_i, b_j) \in C$ then there are no other elements $b_l (l \neq j)$ in B such that $(a_i, b_l) \in T$, nor elements $a_k (k \neq i)$ in A such that $(a_k, b_j) \in T$ (that is, each element in A is aligned at most to one element in B, and vice versa)

③ if $(a_i, b_j) \in C$ and $(a_k, b_l) \in C$ and $i < k$ then $j < l$ (that is, the alignment maintains the colinearity between the sequence A and B).

For instance, this is an example of a map alignment between the maps $A = \{(B,1)(D,15)(A,20)(E,32)(D,50)(F,95)\}$ and $B = \{(B,5)(D,17)(D,47)(C,78)(A,87)(F,92)\}$:

$$\begin{array}{lccccccc} A = & (B,1) & (D,15) & (A,20) & (E,32) & (D,50) & - & - & (F,95) \\ & | & | & & & | & & & | \\ B = & (B,5) & (D,17) & - & - & (D,47) & (C,78) & (A,87) & (F,92). \end{array}$$

Let α be the reward given to each matching pair (optional), let λ be the penalty associated to each unaligned site from both maps and let μ be the penalty associated to the discrepancy in distance between adjacent aligned pairs $(a_{i_{t-1}}, b_{i_{t-1}})$ and (a_{i_t}, b_{i_t}). Then, the score of the map alignment C between maps A and B that contains T matched pairs is defined to be:

$$\begin{aligned} S(C) = \; & \alpha T \\ & -\lambda(m + n - 2T) \\ & -\mu(|q_{i_1} - p_{i_1}|) \\ & -\mu \textstyle\sum_{t=2}^{T} (|(p_{i_t} - p_{i_{t-1}}) - (q_{i_t} - q_{i_{t-1}})|) \\ & -\mu(|(p_{i_m} - p_{i_T}) - (q_{i_n} - q_{i_T})|). \end{aligned} \tag{3.31}$$

That is, the score of the alignment increases with the score of the matches of the aligned elements (α, optional), and decreases with the number of elements not in the alignment (λ), and with the difference in the distance between matches of consecutive aligned elements (μ).

The Waterman *et al.* map alignment algorithm (1984)

? firstly formalized the problem of map alignment and introduced an algorithm distinct from usual sequence comparison algorithms, to investigate the relationships among restriction maps of homologous regions.

This algorithm yields a measure of distance between two maps and provides an alignment of them. Such a distance is the minimum weighted sum of genetic events required to convert one map into the other, where the genetic events are the appearance/disappearance of restriction sites and changes in the number of bases between them. Mutations from one site to other are ignored because this event was considered to be unlikely (**?**).

Formulation and cost

Let $A = a_1 a_2 \ldots a_m$ be a map of sites where each pair $a_i = (r_i, p_i)$ represents the restriction site r_i occurring at position p_i of a sequence of nucleotides, let $B = b_1 b_2 \ldots b_n$ be a second map of sites denoted as $b_j = (s_j, q_j)$: a map alignment between A and B is a correspondence $(a_{i_1}, b_{j_1})(a_{i_2}, b_{j_2}) \ldots (a_{i_T}, b_{j_T})$ in which two sites a_{i_t} and b_{j_t} constitute a match if they correspond to the same type of restriction site (see Figure 3.19 for an example).

To measure the distance between two maps, two events must be taken into account:

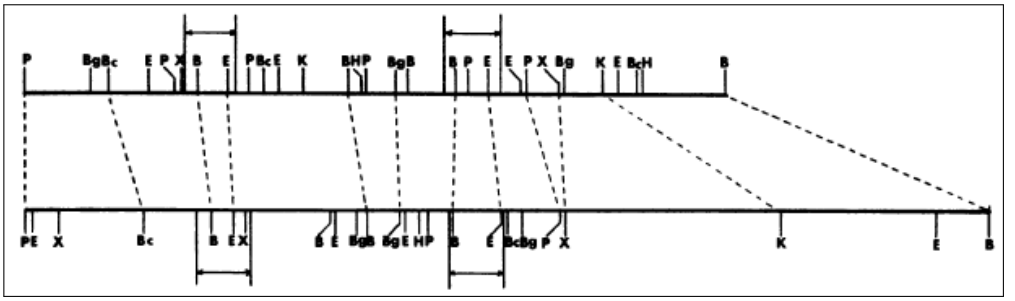


Figure 3.19 A restriction map alignment including the β and δ globin genes from the lowland Gorilla and the Owl Monkey. Adapted from **?**.

① Each site from A and from B that is not aligned receives a weight λ.

② The number of bases between every pair of aligned sites in A that changes by x bases in B receives the weight $\mu(x)$.

Let $D(i,j)$ the minimum sum of weights of events required to convert the map A into the map B where the site a_i is equal to the site b_j (otherwise $D(i,j) = \infty$). Then, $D(i,j)$ is calculated as

$$D(i,j) = \min_{\substack{0<i'<i \\ 0<j'<j}} \{D(i',j') + \lambda(i-i'-1+j-j'-1) + \mu(p_i - p_{i'} - q_j + q_{j'})\}. \tag{3.32}$$

Thus, the optimal alignment ending at a given pair (a_i, b_j), where r_i is equal to s_j, is optimally computed by:

① Searching among the alignments ending at previous matches $(a_{i'}, b_{j'})$.

② Evaluating the value $D(i,j)$ if the pair $(a_{i'}, b_{j'})$ was placed immediately before the current pair (a_i, b_j) in the optimal alignment in construction.

Note that to compute the optimal score at $D(i,j)$ with this algorithm, all the cells $D(k,l)$ with $k < i$ and $l < j$ need to be explored. Therefore, if the length of the two maps A and B is m and n respectively, the cost of computing $D(A,B) = D(a_m, b_n)$ is $O(mn \cdot mn) = O(m^2n^2)$. Under the assumption that m and n are similar, the final cost function is $O(n^4)$. However, as there are hundreds of distinct types of sites, the dynamic programming matrix is actually very sparse (there is a smaller number of matches), being less prohibitive such a cost.

Implementation

A direct implementation of the recursion above involves the recursive filling of the cells $D(i,j)$ in the matrix D (**?**). In the pseudocode below, the elements of the maps A and B are represented as structures a_i and

```
Pre ≡ A, B: maps; λ, μ ∈ Z+
    (* Calculating the element i, j in D *)
    for i = 0 to |A| − 1 do
      for j = 0 to |B| − 1 do
        if site(a_i) = site(b_j) then
 5:       D(i, j) ← ComputeInitialDistance();
          (* Searching the best previous element in D *)
          for i′ = 0 to i − 1 do
            for j′ = 0 to j − 1 do
              y ← λ((i − i′ − 1) + (j − j′ − 1));
10:           z ← μ(|(pos(a_i) - pos(a_i′)) - (pos(b_j) - pos(b_j′))|);
              currentDist ← D(i′, j′) + y + z;
              if currentDist < D(i, j) then
                D(i, j) ← currentDist;
              end if
15:         end for
          end for
        end if
      end for
    end for
```

Figure 3.20 The ? map alignment algorithm.

b_j, with the functions *site* and *pos* returning the values of the corresponding fields. The variable *currentDist* stores the minimum distance so far computed.

The resulting map alignment can be easily retrieved using a supplementary structure *path(i,j)* which points to the previous cell in the optimal path leading to cell $D(i, j)$ In addition, for each cell $D(i, j)$, the function *ComputeInitialDistance* calculates the initial score of a hypothetical alignment that includes only a_i and b_j.

The Myers and Huang map alignment algorithm (1992)

The formulation of the problem by **?** for aligning two maps A and B of m and n sites respectively, leads directly to a $O(m^2n^2)$ algorithm. **?** presented an algorithm for comparing restriction maps based on some works related to sequence comparison algorithms in the cases where gap costs are concave (see Section 3.3). Because of the distance between two maps relies not only on the number of gaps in the lists of sites but also on the physical distances between sites, multiple indels or gaps can be treated as a unit.

Basically, the $O(n^4)$ cost of the original algorithm can be decomposed into two $O(n^2)$ components:

① The worst-case number of possible matches between A and B.

② The cost of retrieving the best previous match that minimizes the distance of the alignment ending at the current match.

While the cost of the first component is unavoidable, the second contribution can be reduced in many

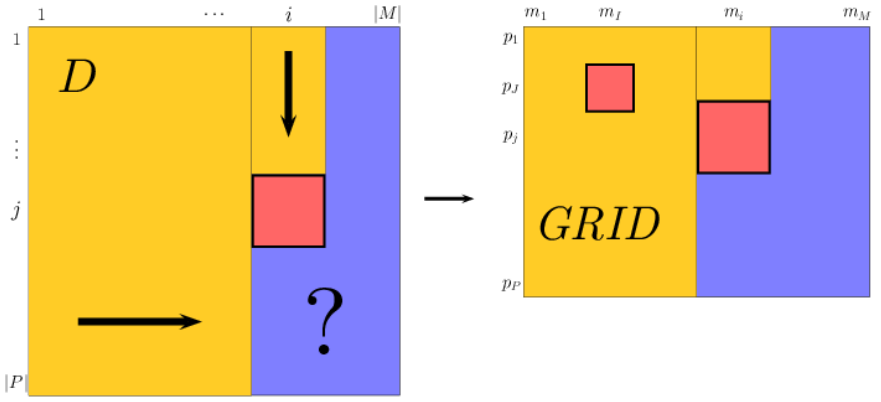


Figure 3.21 Mapping the D matrix over the rectangle $[0, m_M] \times [0, p_P]$.

ways, specially in the cases in which the dynamic programming matrix is very sparse. In **?**, such a cost is dramatically reduced to a logarithmic function through the application of several analytical methods. First of all, the formulation of the score (distance) of a map alignment is rewritten again: the elements that do not depend of the current match (a_i, b_j) are now isolated to be computed only once. Second, the dynamic programming matrix that is addressed with the sites in A and B is substituted by a grid of points that correspond to the physical positions of the elements from both maps. Finally, a list of candidates (previous matches) that induces a partition in the set of sites from the second map is updated when the matrix is filled in, at the same time the sites in the first map are being processed.

Formulation and cost

Let $M = \{M_1, M_2, \ldots M_M\} = \{(a_1, m_1), (a_2, m_2) \ldots (a_M, m_M)\}$ be a map of sites where each pair (a_i, m_i) represents the restriction site a_i occurring at position m_i of a sequence of nucleotides, and let $P = \{P_1, P_2, \ldots P_P\} = \{(b_1, p_1), (b_2, p_2) \ldots (b_P, p_P)\}$ be a shorter map of sites (a probe) where each pair (b_j, p_j) represents the restriction site b_j occurring at position p_j of a sequence of nucleotides. Then, the score of an alignment $C = (M_{i_1}, P_{j_1})(M_{i_2}, P_{j_2}) \ldots (M_{i_L}, P_{j_L})$ between the map M and the probe P is defined to be:

$$Score(C) = \lambda(P - L) + \mu \sum_{k=2}^{L} (|(m_{i_k} - m_{i_{k-1}}) - (p_{j_k} - p_{j_{k-1}})|). \tag{3.33}$$

That is, the distance between the map and the probe according to such an alignment increases with the number of elements of P not in the alignment (λ), and with the difference in the distance between matches of consecutive aligned elements (μ).

Let Matchpoints be the matches between the map and the probe $\{(i, j) | a_i = b_j\}$. Then, to compute the minimum distance between a map and a probe, the Equation 3.32 is rewritten by **?** in terms of the contribution of a previous match to the current one:

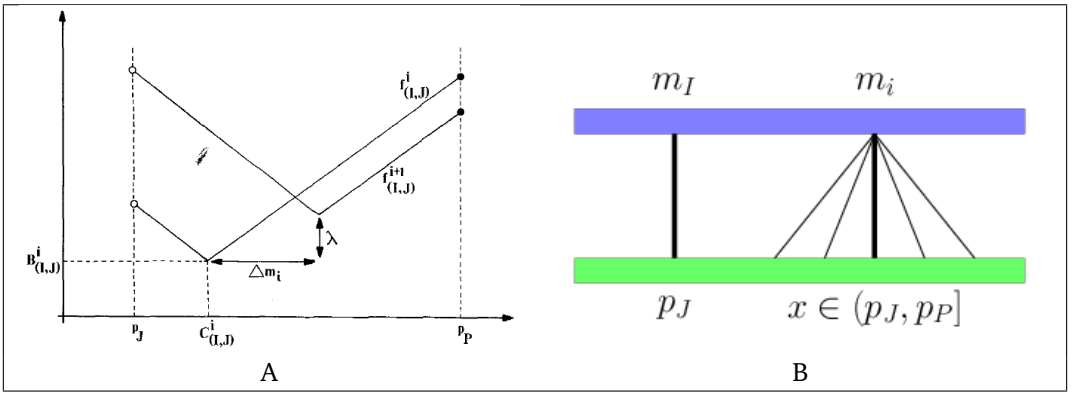


Figure 3.22 Analytical methods in ?. (A) An illustration of $f^i_{I,J}(x)$ and $f^{i+1}_{I,J}(x)$. (B) The contribution of a (I, J) to each match point (m_i, x) in the interval $(p_J, p_P]$ of P.

$$D(i,j) = \min(\lambda(P-1), \min_{\substack{(I,J) \in \text{Matchpoints} \\ I<i, J<j}} \text{contrib}_{I,J}(i,j)). \tag{3.34}$$

Such a contribution of a previous match (I, J) to the current one (i, j) is defined as:

$$\text{contrib}_{I,J}(i,j) = D(I,J) + \lambda(i-I-2) + \mu(|(m_i - m_I) - (p_j - p_J)|). \tag{3.35}$$

Instead of dealing with the classical dynamic programming matrix that is usually accessed using the sites in the maps, **?** proposed to map the original problem into a matrix representing the domain of physical positions. Thus, the procedure that completes the original matrix column by column is exported to this new grid whose dimensions are the position of the last site in both maps respectively (see Figure 3.21).

This algorithm computes each column of the matrix D in increasing order of i (M), simultaneously updating a list of match points called candidates. Each one of these previous matches (I, J) are actually associated to a given partition of the probe P, constituting the best previous match for the current point $D(m_i, p_j)$ in this column m_i and row p_j. The step of scanning back the matrix to retrieve the best previous match is then substituted with a list that returns the best element in a logarithmic time. Several additional definitions must be provided to manage the candidate list. These concepts are all of them based on an analytical description of the computation of the matrix D.

The contribution f of a match point (m_I, p_J) to future points in a given column m_i $(I < i)$ on the interval $x \in (p_J, p_P]$ can be divided into the components associated to λ and μ. At the same time, each one can be split into the values that depend on the current m_i and those that were already computed when the match point (m_I, p_J) was reached:

$$\begin{array}{ll} f^i_{I,J}(x) = \mu|C^i_{I,J}(x)| + B^i_{I,J} & \text{where} \\ C^i_{I,J} = m_i + \Delta_{I,J}, & \Delta_{I,J} = p_J - m_i \\ B^i_{I,J} = \lambda i + E_{I,J}, & E_{I,J} = D(I,J) - \lambda(I+2). \end{array} \tag{3.36}$$

It is direct to see that $\text{contrib}_{I,J}(i,j) = f^i_{I,J}(p_j)$, as the terms in Equation 3.33 have been simply rearranged. For the μ factor, $(m_i - m_I) - (p_j - p_J) = m_i + (p_J - m_I) - x = m_i + \Delta_{I,J} - x$. For the λ factor, $D(I,J) + \lambda(i - I - 2) = D(I,J) - \lambda(I+2) + \lambda i = \lambda i + E_{I,J}$. In this case, the values $\Delta_{I,J}$ and $E_{I,J}$ do not depend on i, being already precomputed.

The new contribution of a match point (m_I, p_J) to the next position (column m_{i+1}) is easily computed from its contribution to the previous one:

$$f^{i+1}_{I,J}(x) = f^i_{I,J}(x - \Delta m_i) + \lambda \tag{3.37}$$

The updating consists of two changes: (1) a unit of λ is increased because a new site has not been included in the alignment (m_i); (2) the physical position of the match point (m_{i+1}, x) must be updated in the computation of the μ factor by decreasing x with Δm_i to recover the new value of the μ penalty.

As shown in Figure 3.22 (A), a given function $f^i_{I,J}$ can be represented graphically. The minimum value that can be reached is $B^i_{I,J}$ corresponding to the point $x = C^i_{I,J}$. For the rest of x values, the λ penalty and $D(I,J)$ are the same so that changes depend directly from the μ penalty. This value will decrease as long as x approaches the $m_I - m_i$ vertical until $C^i_{I,J}$. From that point, it will increase again due to the progressive movement away such point (see Figure 3.22 (B)). The contribution in the next column can also be represented as a similar function with the corresponding new values of x and $f(x)$.

Let m_i be the current column: each previous match point (m_I, p_J) has a different contribution to each one of the new match points (m_i, x) found in this column. For a given x, the best contribution of the previous match points in the alignment ending at such point is the minimum value among the $f^i_{I,J}(x)$ functions. In addition, the optimal value $D(i,j)$ is either the distance of an alignment containing only this match point or the alignment with the previous match point that showed the highest contribution for (m_i, x):

$$\begin{aligned} P^i(x) = &\quad \min_{\substack{I,J \\ (I,J) \in \text{Matchpoints} \\ I < i, J < j}} f^i_{I,J}(x) \\ D(i,j) = &\quad \min(\lambda(P-1), P^i(p_j)). \end{aligned} \tag{3.38}$$

An i-profile is then defined to be the intersection between the contributions of all of the available match points computed before. For each interval between two consecutive points in P, the f function with the lowest value over there is claimed to be the owner of such interval. The calculation of the value $P^i(x)$ consists on locating the representative of an interval to know its contribution. In Figure 3.23, the i-profile represented as the minimum envelope of the f-curves of all match points left to the column m_i is graphically shown.

For simplicity, the list of candidates that form an i-profile is decomposed into two different lists L and R which correspond to the parts of the f-curves that are before and after the point $C^i_{I,J}$. For each one, insertions and updates to the list of candidates must be performed in a slightly different form (see Figure 3.24).

In the case of the R-list, their members are always increasing straight lines. For that reason, whenever one of the candidates has another candidate below, this first element is said to be dominated by the second one. When this candidate is dominated in all of the intervals over P, it becomes dead and it is removed from the list. In the case of the L-list, the processing must take into account the stationarity of the left ends of the curves when shifting horizontally.

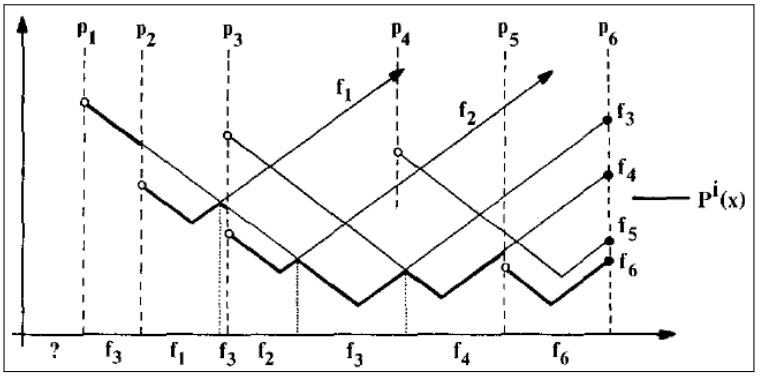


Figure 3.23 An illustration of an i-profile. Adapted from **?**.

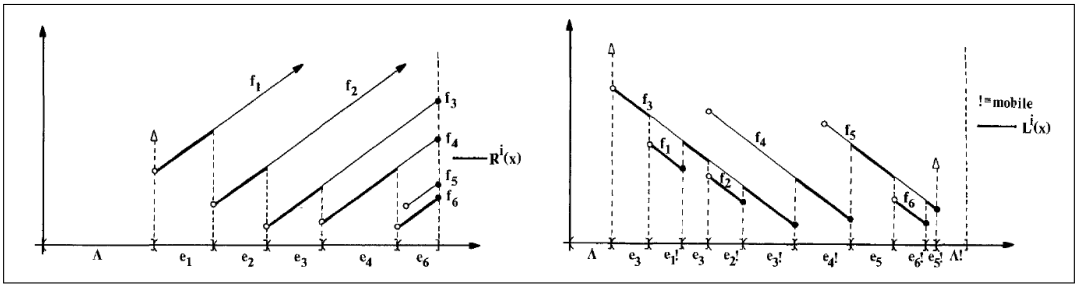


Figure 3.24 An illustration of a R-profile and a L-profile. Adapted from **?**.

For both lists the management is similar: each match point that has just been computed must be inserted into the L-list and the R-list. This insertion can cause the removal of the match points that become dominated by this new element. Similarly, once a column m_i has been processed, the elements of the lists must be processed to be ready for the next column m_{i+1}, involving the recomputation of their values using Δm_i. Again, this operation can force some match points to be removed from the list because they can not contribute positively to any of the future ones.

These type of sorted lists that must provide a direct access to a given element (e.g. the owner of an interval) can be implemented using balanced trees. These trees support logarithmic insertion, deletion and search primitives. Let M and P the length of the map and the probe respectively, there are $R = MP$ potential match points. For each one, the owner of its interval can be retrieved in a logarithmic time $O(\log(P))$. The insertion in the list and the shifting operation are also performed with a logarithmic cost taking into account some particular considerations. Thus, the final cost of this algorithm is $O(R \log(P))$ (for further details see **?**).

Such an algorithm was designed primarily for comparisons between a map and a probe. However, **?** also presented some changes to convert the problem in a comparison between two maps of length M and N respectively in $O(MN(\log M + \log N))$ time.

```
Pre ≡ M, P: maps; λ, μ ∈ Z+
    Initialize_candidate_list();
    (* Current column m_i *)
    for i = 0 to |M_m| − 1 do
      for j = 0 ∈ Match(site(m_i)) do
 5:     D(i, j) ← min(λ(P − 1), Find_min(i, j));
      end for
      if i < M then
        for j ∈ Match(site(a_i)) and j < P do
          Insert(i, j);
10:     end for
        Update(i);
      end if
    end for
```

Figure 3.25 The ? map alignment algorithm.

Implementation

The main algorithm of the **?** strategy consists of a loop that visits column by column the $m_M \times p_P$ matrix. For each site in m, there is a function Match(site(a_i)), precomputed only once at the beginning, which returns the sites x in P that share the same restriction enzyme.

Then, the optimal alignment ending at every new match point (m_i, x) is constructed between either an alignment only constituted by this match or the contribution of the owner of its interval, which is directly identified with the function Find_min by accessing the list of candidates implemented as two balanced trees (the L-list and R-list).

The new match points that have been processed in the current column are then inserted in the corresponding lists with the function Insert, removing from the lists those candidates that are categorized as dead.

Once the current column has been completely processed, both lists of candidates must be updated with the function Update to be prepared for the next element m_{i+1}, taking into account the value of Δm_i.

Chapter 4

Computational Gene and Promoter Characterization

Summary

The computational identification of genes in an eukaryotic genome and the description of their promoter regions are reviewed here. An important fraction of the information used by the cell to activate the genes and to recognize their protein-coding regions is contained in the genomic sequences. The methods to represent such cellular signals and to detect functional regions presenting unusual statistical content are similar in both cases. This chapter introduces the different alternatives proposed throughout the past years, providing a glimpse of the future.

4.1 Genes and promoters

Towards a catalogue of the genome

ONE OF THE MAJOR PROBLEMS THAT BIOLOGISTS HAVE EVER FACED is how to extract relevant information from millions of nucleotides produced by large-scale genome sequencing projects. The first task is to locate all protein-coding genes encoded in the genomic sequence to able then to characterize the regulatory content of the genome (**?**).

Genes are switches regulated by cellular mechanisms which turn them on or off according to different situations and circumstances. The identification of the promoter elements required for the correct expression of genes is crucial to understand why many genetic diseases are caused and perhaps, how to prevent or stop them.

Computational gene-finding and promoter characterization have been traditionally strongly related. Both methods process the genomic sequence using similar techniques in order to extract the information that is used by the cells to control the production of genes. However, the elaboration of catalogues of genes in eukaryotes have shown to be more feasible in practice than the construction of regulatory maps because of the specific nature of each problem. Nonetheless, promoters are still very interesting for gene-finding because their detection will help to improve the accuracy of current gene predictions. Therefore, the complete annotation of a gene should include both the protein-coding regions and the promoter elements that govern its expression (**?**).

Eukaryotic gene structure

The identification of genes is difficult, specially because of their fragmented nature and the large spacers found between them. Only 2% of the 3,000 million nucleotides in the human genome are estimated to code for proteins (**?**).

As explained in Chapter 2, the splicing machinery removes from the transcript those regions that are not coding for proteins (introns), joining the coding fragments (exons). The mRNA is constituted of the coding sequence (CDS) and the untranslated region (UTR). For further details about the general structure of an eukaryotic gene see Figure 4.1.

Most gene computational tools can only predict the location of the coding exons of a gene. Essentially, the splicing and translation signals are first located in order to construct then the possible reading frames that form the exons. Typically, there are four types of exon-defining signals:

① Start codons: the first amino acid of a protein is usually the Methionine, coded with the codon ATG. It represents the beginning of a translation.

② Stop codons: there are three codons (TAA, TAG and TGA) that end the translation of a mRNA.

③ Acceptor splice site: the right part (3') of a removed intron contains this signal. It represents the nucleotides immediately before the beginning of an exon.

④ Donor splice site: the left part (5') of a removed intron contains this signal. It represents the nucleotides immediately after the end of an exon.

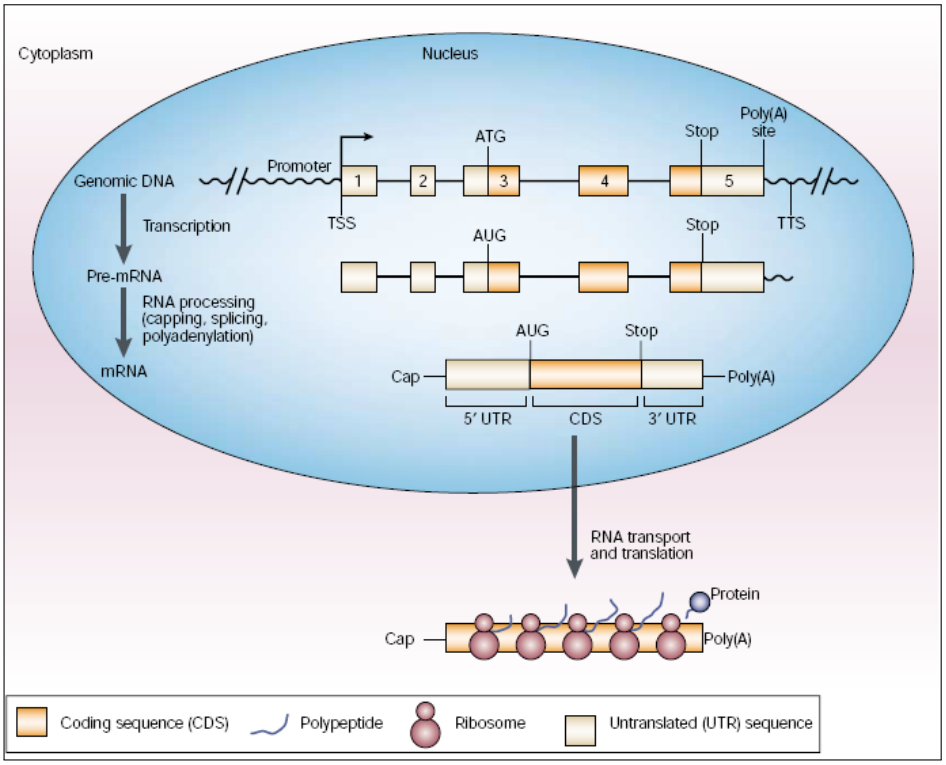


Figure 4.1 The typical gene structure. TSS is the transcription start site. TTS is the transcription termination site. ATG/AUG is the translation start codon. Adapted from **?**.

With such signals, the following types of exons can be defined:

① Initial exons (Start codon - Donor site): the first coding exon of a gene

② Internal exons (Acceptor site - Donor site): the set of coding exons between the initial and the terminal ones

③ Terminal exons (Acceptor site - Stop codon): the last coding exon of a gene

It is important to mention that, due to the existence of exons completely or partially constituting the UTR region at both ends of a gene, the initial and terminal coding exons predicted by a computational approach do not usually correspond to the authentical ends of the transcript.

Other forms of gene structures

Gene identification is not an easy problem. Nowadays, there are still serious discussions to establish the exact number of genes in an organism. One of the reasons for this controversy is the definition of what a gene is. Exceeding the classical definition “one gene for one protein”, biological reality has shown how

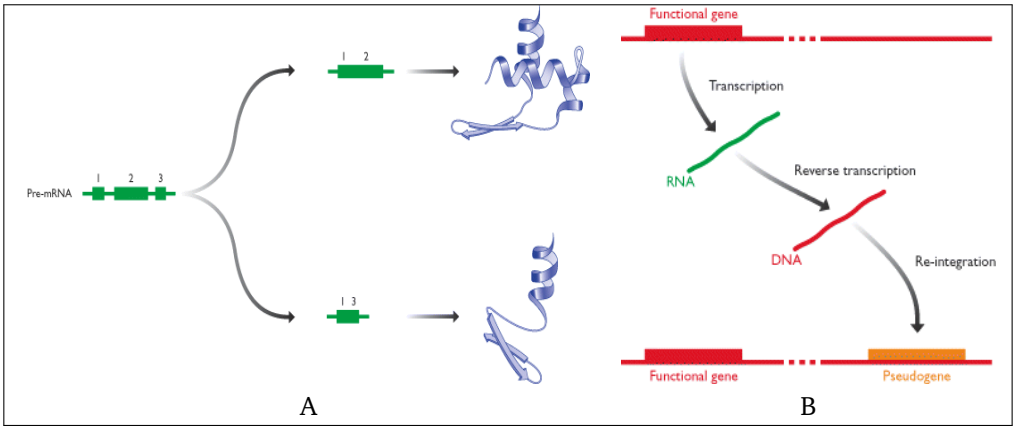


Figure 4.2 Other forms of gene structures. (A) Alternative splicing results in different combinations of exons from the same pre-mRNA. (B) The origin of a processed pseudogene. Adapted from **?**.

things are more complex. A better biological understanding of these facts will help to to obtain in the future more accurate gene predictions (**?**).

These are other forms of gene structures that exceed the classical definition of a gene:

- Alternative spliced genes: 60 % of human genes can be spliced following different patterns of exons and introns, omitting some exons or altering the length of others to produce different proteins (**?**). See Figure 4.2 (A) for an example of alternative splicing.
- Pseudogenes: due to the continually changing nature of the genomes, some genes have been inactivated by excess of mutations (conventional pseudogene). Processed pseudogenes are the result of the insertion in the genome of a reversed-transcribed mRNA copy of a gene. See Figure 4.2 (B) for an example.
- Intronless genes: genes without introns (prokaryotic origin).
- Non-coding genes: some genes correspond to specific RNA molecules playing crucial roles in the cell that are not translated into a protein.
- Non-canonical spliced genes: splicing signals in most genes present certain dinucleotides as characteristic signatures. However, other types of splicing signals occurring in a minority of genes are recognized by a different splicing machinery (**?**).
- Genes-within-genes: some human genes have been found to be within long introns of others. These internal genes can be affected by the normal splicing process as well (**?**).
- Selenoproteins: some codons can be translated into different amino acids according to each situation (context-dependent codon reassignment). For instance, in presence of a secondary structure in the mRNA called SECIS, the codon TGA is translated into the novel amino acid Selenocysteine instead of stopping the process (**?**).

Eukaryotic promoter structure

The expression of a gene is the appearance of an observable feature or action caused by the effect of the protein encoded by this gene. Gene regulation is the mechanism which determines the amount of protein product that must be syntesized by switching the genes responsible for that protein on or off. Only a subset of genes in an eukaryotic cell are expressed at each instant, considerably changing this regulational composition during the life cycle.

But research about gene expression is not trivial: a human cell can be seen in terms of a black box with approximately 20,000 inputs, one per gene. Such box must work with $2^{20,000}$ states, since every gene would be either on or off. This number can be approached to $10^{6,000}$ while the number of particles in the universe is believed to be about 10^{80}. Moreover, the degrees of intensity and the large network of relationships among related genes are neglected in this estimation.

In fact, little is known about the relationship, for instance, between transcription and splicing. More and more evidences are being gathered to postulate that both processes are in fact performed simultaneously or at least in a very intimate manner (**?**).

Checkpoints in the pathway from DNA to protein

There are actually two levels of gene expression control along the pathway from DNA to RNA to protein (**?**). The primary level selects which genes have to be expressed and which not and belongs to the process of transcription (see Figure 4.3). The second level is necessary to modulate the expression of a gene by changing the rate of production or by modifiying the nature of the product (RNA, protein) using post-transcriptional methods.

Specifically, this control is implemented through different stages:

① Accessibility: What regions of a chromosome are visible for being transcribed

② Transcriptional control: When and how often a given gene is transcribed.

③ RNA processing control: How the primary transcript is spliced.

④ RNA transport control: Which mRNAs are exported to the cytoplasm.

⑤ RNA translational control: Which mRNAs are translated by ribosomes.

⑥ RNA degradation control: Which and when mRNAs have to be destroyed.

⑦ Protein activity control: (In)activating synthesized protein molecules.

Transcriptional regulation: promoters

Transcriptional regulation is a highly dynamic process. Most of genes are governed by variable temporal and spatial heterogeneous profiles. The promoter sequences are functional regions located immediately upstream the transcription start site of the gene (TSS). Many genes usually possess several alternative TSSs, having

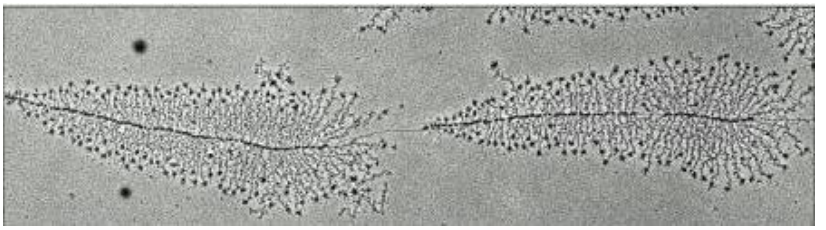


Figure 4.3 Transcription of two tandem genes as observed under the electron microscope. Each gene is being transcribed simultaneously by hundreds of RNA-polymerase II. Adapted from **?**.

therefore different promoter regions. The main function of a promoter is the integration of information about the status of the cell, to alter the rate of transcription of a single gene accordingly (**?**).

In Figure 4.4, a promoter prototype is represented as a gene specific container for the assembly of some special proteins called transcription factors (TFs). The TFs are responsible for recruiting the RNA-polymerase II that performs the transcription from DNA into RNA molecules. Every gene is regulated by a core of general TFs and a combination of gene-specific TFs located upstream the TSS. About 1,800 different TFs are estimated to be encoded in the human genome (**?**).

The TFs are attracted to the promoter region by very specific motifs imprinted in the DNA called TF binding sites (TFBSs). From the study of a well-characterized set of eukaryotic promoters, the occupation of a promoter has been estimated to be about 10 to 50 TFBSs for 5 to 15 different TFs (**?**). TFs are usually arranged along the promoter region following very restrictive rules such as minimum/maximum distance or neighbourhood constraints (**??**).

The problem of finding regulatory elements is extremely difficult due to many reasons (**?**):

- There are thousands of differents TFs.
- TFBSs are short: tipically 5-15 nucleotides long.
- Each TF can connect to more than one different binding site.
- Each TFBS can recruit different TFs.
- The core promoter is not universal, presenting high diversity as well.
- TFBSs can form clusters of regulatory modules or composites.
- The poor knowledge about the biological interactions between different TFs.

Eventually, some regulatory regions called enhancers are located within intergenic segments, being able to affect several loci in other parts of the genome. First exons and introns are also known to contain some regulatory signals as well. In addition, other promoter regions control the coordinate expression of two bidirectional genes, that is, gene pairs that are arranged head-to-head on opposite strands with less than 1,000 nucleotides separating the TSSs (**?**).

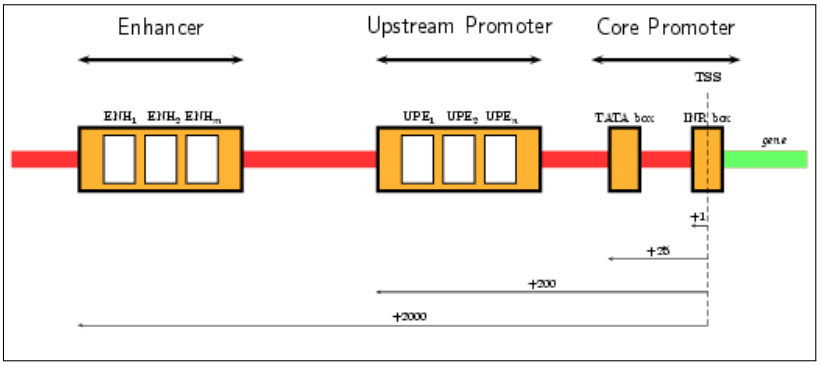


Figure 4.4 A schematic representation of a promoter.

Chromatine structure and gene expression

In Eukaryotes the chromatin is packaged into a compact structure with the aid of a class of proteins called histones. The nucleosomes, the fundamental packaging units, are histones with DNA wrapping around (**?**). Chromatin packagement plays an important function of regulation before the beginning of the transcription. To be transcribed, a promoter must be physically accessible to the RNA polymerase for starting the copy (see Figure 4.5).

If a region containing a gene is not momentaneously accessible, that gene is said to be silenced. RNA polymerases can transcribe a region containing attached nucleosomes when they are moved slightly by thermal effects. This process allows the polymerase to copy short regions of DNA while the nucleosome shifts to a position near the end of the transcription. Thus, nucleosome positioning and distribution of genes into visible and not visible regions of chromatin are some types of pre-transcriptional control (**?**).

Methylation and CpG islands

In eukaryotes, Cytosine bases in CpG dinucleotides from chromosomal DNA molecules are sometimes modified with the addition of methyl groups by special enzimes which maintain this feature through the offspring of a cell. Such process is named methylation. The inheritance of methylation patterns is a feasible explanation to the cell memory event and is also associated with repression of gene activity.

Some correlation between the degree of methylation and the level of transcription of genes has been observed. Methylation is thought to be relationed with the way histones move and stand along the DNA molecules of chromatin and therefore with the silencing of genes as well (**?**).

CpG islands are regions of several hundreds of nucleotides in which the frequency of the dinucleotide CpG and the G+C content are higher than the average for the rest of genome (**?**). Most of the CpG islands in the human genome are methylated. However, the CpG islands that are adjacent to housekeeping genes[1] are unmethylated, being the genes potentially active.

[1] Genes that are expressed generally in every phase of the cell cycle.

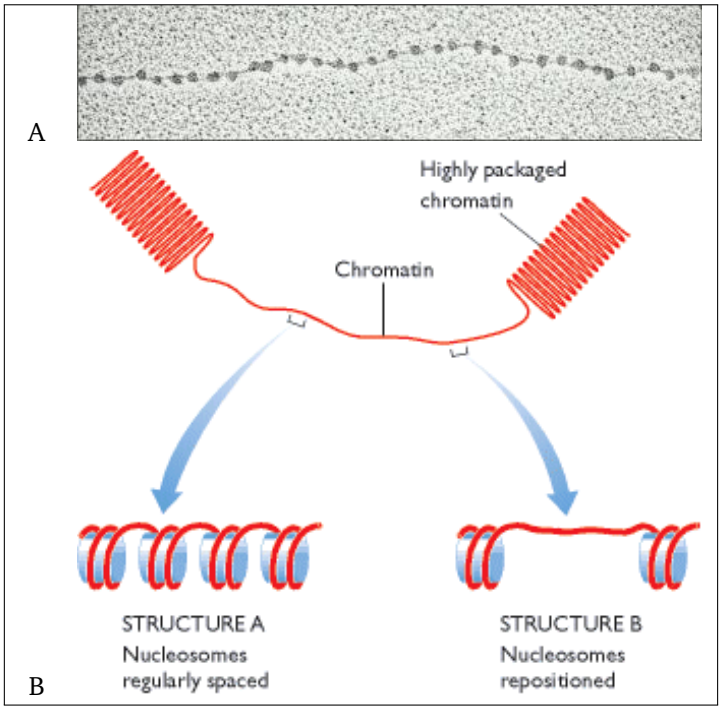


Figure 4.5 Nucleosomes and chromatin structure can influence gene expression. (A) Nucleosomes as seen in the electron microscope. Adapted from (**?**). (B) A region of unpackaged chromatin in which the genes are accessible is flanked by two more compact segments. On the left, the nucleosomes have regular spacing structure. On the right, the nucleosome positioning has changed and a short stretch of DNA is exposed for transcription. Adapted from **?**.

4.2 Computational approaches

Gene identification and promoter characterization methods essentially process similar input sequences with many common algorithmic approaches. However, the underlying biological problem is slightly different. The genes are regular structures formed by exon-defining signals with several exon features usually well conserved. The promoter regions instead are more flexible arrangements of TFBSs which, in addition, present a higher variability in their motifs.

Gene-finding methods normally use three different types of information to build a prediction: splice sites and translational signals, protein-coding potential measures, and similarity searches. Ab initio methods only rely on the investigation of the statistical properties of annotated coding sequences: signals and coding statistics. As shown in Figure 4.6, the combination of signals and content measures with an assembly algorithm of exons, typically based on dynamic programming, produces a predicted gene (**??**). Homology methods compare directly the sequence of interest to known coding sequences or even orthologous regions of other genomes using alignment programs.

Promoter characterization methods are often based on the detection of the motifs specifiying a family of TFBSs. A combinatorial set of rules can be designed to propose arrangements of sites in groups of few

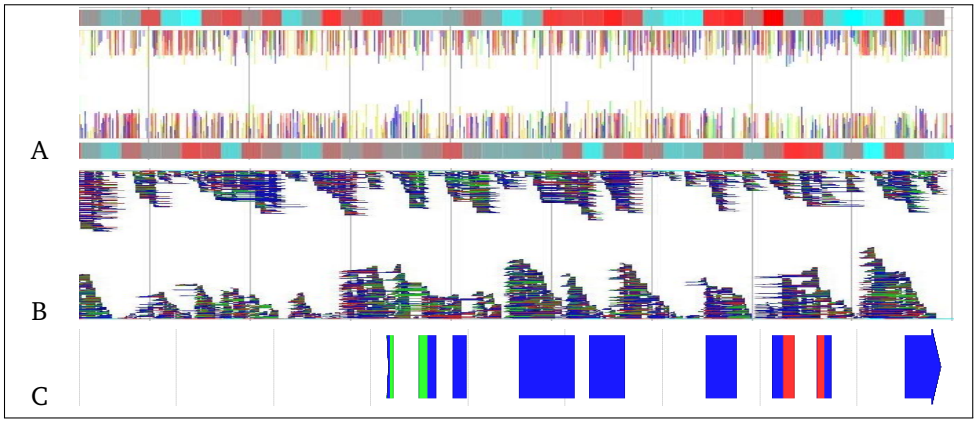


Figure 4.6 Sources of information in the ab-initio gene-finding process (in both strands). (A) Signal and content information: vertical bars are predicted splicing signals; the red-blue code measures the coding potential of the sequence. (B) Predicted set of coding exons. (C) Optimal gene structure assembled from the set of predicted exons with a dynamic programming algorithm.

elements (composites or modules). There is a severe lack of biological knowledge about the promoter structures (**??**). Despite this, promising advances have been obtained using homology methods based on the phylogenetic conservation of regulatory elements and the introduction of high-throughput expression data (**?**).

4.3 Detection of signals

Sequence signals or sites are defined as short, functional DNA elements involved in gene specification or transcriptional regulation. There is not a typical unique sequence of nucleotides that can be associated to each class of signal. Nonetheless, certain trends in the conservation of some base pairs in these motifs are usually detected, being statistically measured.

Because of the importance of these signals to characterize genes and promoter regions, an important family of techniques based on the use of an external catalogue of known examples have been designed for their detection: the pattern-driven algorithms (**?**), also called the search by signal approaches (**?**).

A naive procedure for scanning a genomic sequence suspicious to contain a functional element will always produce an enormous list of false positives due to the short length of most genomic signals and the high probability to find the same subsequence by chance in other region. To circumvent this problem, the pattern-driven algorithms usually rely on three steps:

① The construction of a catalogue of experimentally annotated sites of a given class

② The representation of this set of examples to mask their variability without losing information

③ The detection of new sites in other sequences using those representations of real examples, as in the algorithm shown in Figure 4.7.

```
Pre ≡ S: sequence; M: signal model; L, STEP, T: integer;

    i ← 1;
    j ← i + L;
    (* Apply the model on each window of length L *)
 5: while i ≤ |S| − L + 1 do
      (* Evaluate the current candidate with this model *)
      score ← M(S_{i,j});
      (* Report the candidates above a quality threshold *)
      if score ≥ T then
10:     ReportCandidate(S_{i,j},score);
      end if
      i ← i + STEP;
    end while
```

Figure 4.7 Pattern-driven algorithms.

Construction of a catalogue

Pattern-driven methods need an input set of real (annotated) elements to build a profile that represents such a family of signals. These samples are usually extracted from public databases of annotated gene and promoter regions.

A high-quality collection of exons extracted from the genome browsers annotations must be used to compile a set of real splicing and translation signals. Typically, the real signals are extracted from the boundaries of the exons, while a set of false signals is built from any similar sequence detected in the introns (see **??** for an example of construction of evaluation sets).

Due to the lack of experimental high-throughput methods to verificate and annotate regulatory functions, the amount of real regulatory signals is very small in comparison to the exon-defining ones. Despite this, several regulatory catalogues are available such as the databases TRANSFAC (**?** , see Web Glossary, page **??**), JASPAR (**?** , see Web Glossary, page **??**) or PROMO (**?** , see Web Glossary, page **??**). New regulatory databases specifically oriented to the training of computational tools are emerging now, such as the Cold Spring Harbor Laboratory Mammalian promoter database (**?**, see Web Glossary, page **??**) or the ABS database of orthologous TFBSs (**?**, see Web Glossary, page **??**).

A correct annotation of the TSS is also crucial for the correct extraction of the promoters. However, such a signal has been poorly characterized so far, being in practice useless to predict its location by computational means. The EPD (**?**, see Web Glossary, page **??**) and the DBTSS (**?**) databases maintain collections of experimentally determined TSSs.

Representation of functional sites

Representing a biological signal site as a unique string is very unrealistic. A large number of sequences containing the same signal (exon-defining or regulatory) represents a good statistical sample of the sequences that are likely to exist in the genome with the same function. However, the alignment of them will probably show differences in the context or even in the apparently best conserved positions of the core (see the

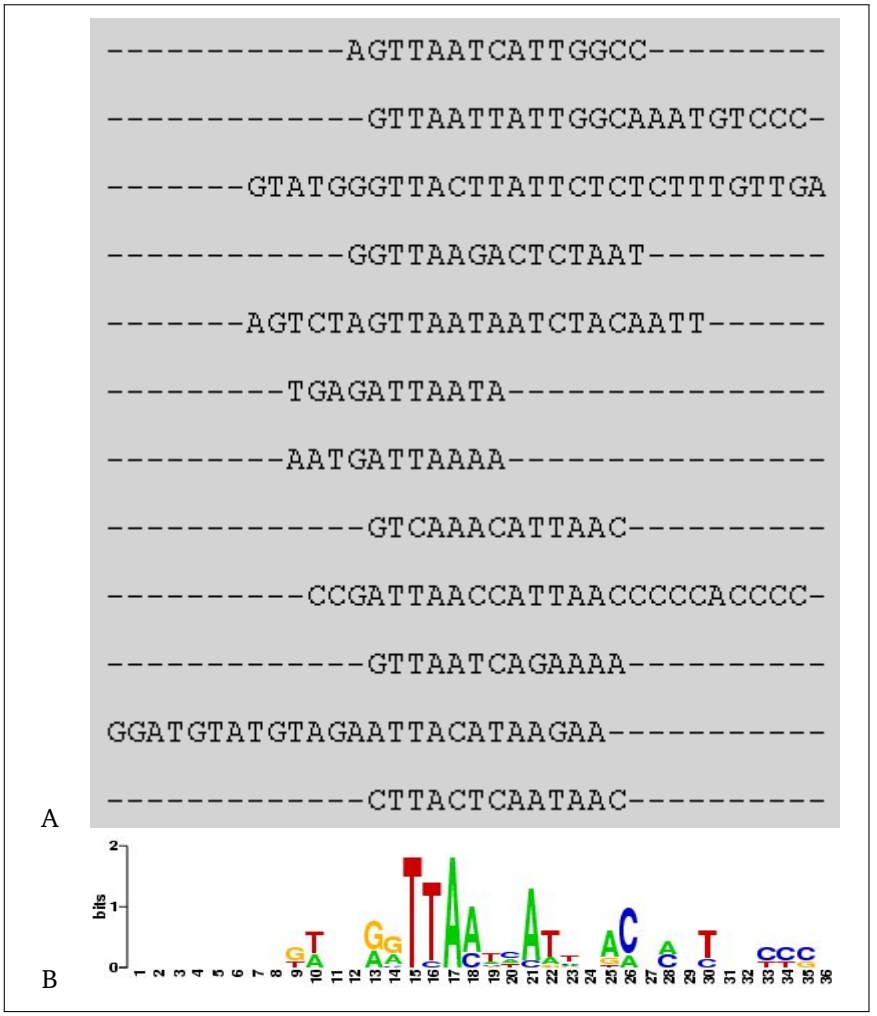


Figure 4.8 Alignment and representation of a set of TFBSs. (A) Global alignment of 12 human sites of HNF-1 α. (B) Sequence logo constructed from the multiple alignment.

example in Figure 4.8).

This limitation leads to a simple question: given a collection of biological signals, how to develop a representation or model to characterize them. Several data structures have been designed to retrieve enough information from the input sequences to be able to recognize putative sites in other sequences (see **??** for a review).

- Deterministic patterns:
  - Consensus sequences: sequences constructed by selecting the nucleotide appearing more often at each position of the motif in the examples.
- Probabilistic patterns:

- Position weight matrices: a numerical representation that registers the frequency of each nucleotide at each position of the motif in the examples.
- Hidden Markov models: a stochastic procedure that registers the dependencies between each nucleotide and the previous group of k nucleotides at each position of the motif in the examples.

- Non-symbolic representations:
  - Neural networks: machine-learning methods that represent the stronger dependencies found in the examples with stronger conectivities in an artificial network.

Example: position weight matrices (PWMs)

Once a collection of real binding sites is aligned, a more sophisticate treatment of the information than a simple consensus sequence can be performed. PWMs[2] are two dimensional arrays of values that represent the score for finding each of the possible sequence characters at each position in the signal that is being analyzed (**?**).

Such a score is derived from the frequency of each nucleotide observed in a set of real functional sites (see Figure 4.9 for an example of PWM). Because some positions are more conserved than others, this is a flexible method to represent sites, under the hypothesis that different positions within the site make independent contributions to the total score. As the most conserved positions are supposed to be relevant for the biological activity of the site, any sequence that differs from the consensus will have a lower score proportional to the significance of the mismatching positions in the motif (**?**).

PWMs are used to score new sequences that could contain a signal of the same family (e.g. splice sites in **?** or promoter elements in **?**). Each position of the matrix is a weight. Weights are employed to score every position of a candidate signal. The sum of these weights according to the content of such a sequence is the score of the candidate (see Figure 4.9).

There are several types of PWMs (**?**):

- Frequency matrices contain the absolute frequency of a nucleotide at each motif position
- Weight matrices contain the relative frequency of a nucleotide at a motif position as an estimation of the probability of this fact
- Log-likelihood ratio or log-odds matrices contain at each position the log of the quotient between the probability of finding a particular nucleotide at such a position position in sequences containing the real motif and the background frequency of the letter at the same position (usually computed from DNA random sequences). To eliminate null values, pseudocounts are usually added to every weight in the matrix.

PWM main drawbacks are two: first, the need for a threshold to filter candidates once the matrix has been used to search for putative sites in new sequences; second, the difficulty to estimate the length of the matrix depending on the interesting positions that show a stronger bias or conservation in comparison with the context (**?**).

In the case of the promoter regulation, an additional serious inconvenient has been detected. Because of the high degree of ambiguity for a TF to select a binding site, the majority of the PWMs representing

[2] PWMs are sometimes called Position-Specific Scoring Matrices (PSSMs).

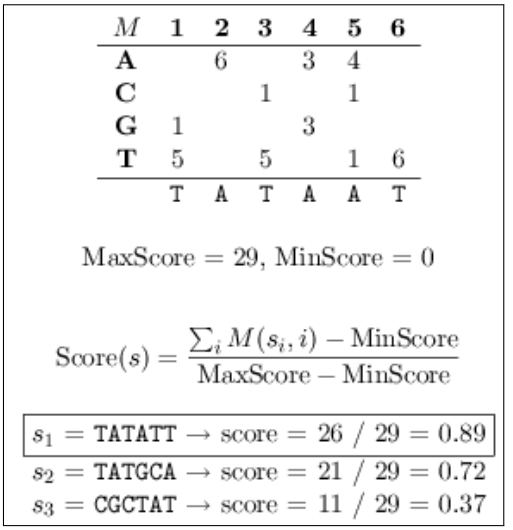


Figure 4.9 A Position Weight Matrix. A naive scoring system is also presented. Three candidates are scored. Only the first one would be over a reasonable threshold of 85% of similarity to the original matrix.

classes of TFBSs are very unspecific. Recently, **?** measured the similarity between the matrices of several popular collections, reporting the existence of classes of equivalences between PWMs of different TFs. This unexpected result is probably produced by the small number of cases employed to construct such models (**?**).

PWMs and information content

The quality and quantity of information provided by the PWMs is different for each column in the motif and can be explained in terms of entropy or amount of uncertainty, expressed in bits per symbol for each position in a PWM (see **?** for a review of the topic).

Given i, a position in a PWM, and (p_A, p_C, p_G, p_T), the relative frequencies of the four possible nucleotides in that column, the information content of this position is defined as (**?**):

$$H(X) = -\sum_{x=A,C,G,T} p_x \log(p_x). \tag{4.1}$$

According to H, the maximum uncertainty is reached when $p_A = p_C = p_G = p_T = 0.25$. In this situation, no additional information can be assumed to guess what nucleotide will be found over there. Obviously this is not the preferred situation because no particular trend or bias is observed. The opposite situation happens when one of the nucleotides dominates the rest of them: $p_A = 1, p_C = p_G = p_T = 0$. The absence of uncertainty in that position reflects a high degree of conservation that might be explained in biological terms. In general, some nucleotides tend to dominate the distribution in a subset of consecutive positions in the signal (the footprint or core). Instead, the context around usually shows a weaker conservation although discontinuities may happen along the matrix.

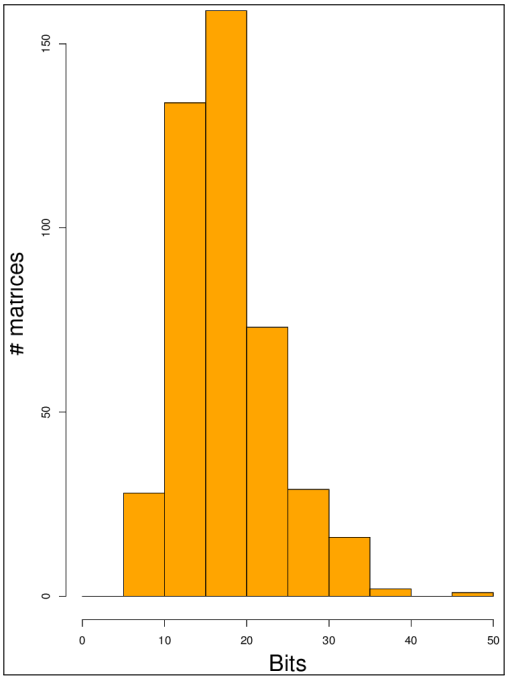


Figure 4.10 Information content of TRANSFAC 6.3 matrices.

The amount of uncertainty of a PWM can be depicted in a sequence logo as in Figure 4.8 with the most conserved positions clearly highlighted (**?**). Motif positions are represented along the horizontal axis while the height of every column corresponds to the lack of uncertainty, that is, maximum entropy (2 bits in DNA) minus entropy computed for that position. The higher the column, the more conserved that position is.

The distribution of TRANSFAC matrices (**?**) according to their information content, calculated as shown in Equation 4.1, is presented in Figure 4.10.

4.4 Content recognition

The analysis of word counts has been very relevant in the detection of interesting regions in sequences of DNA. Historically, this analysis has been applied to locate functional sequences whose statistical content was significantly different from the values expected in non-functional regions.

Once a method to count oligo-nucleotides has been implemented, two approaches are possible. On the one hand, the search can be devoted to detect those regions richer in words that are statistically similar to the type of words observed in functional regions. On the other hand, the search can be directed to locate over-representations that are a priori unkown, reporting then such words in a set of related sequences.

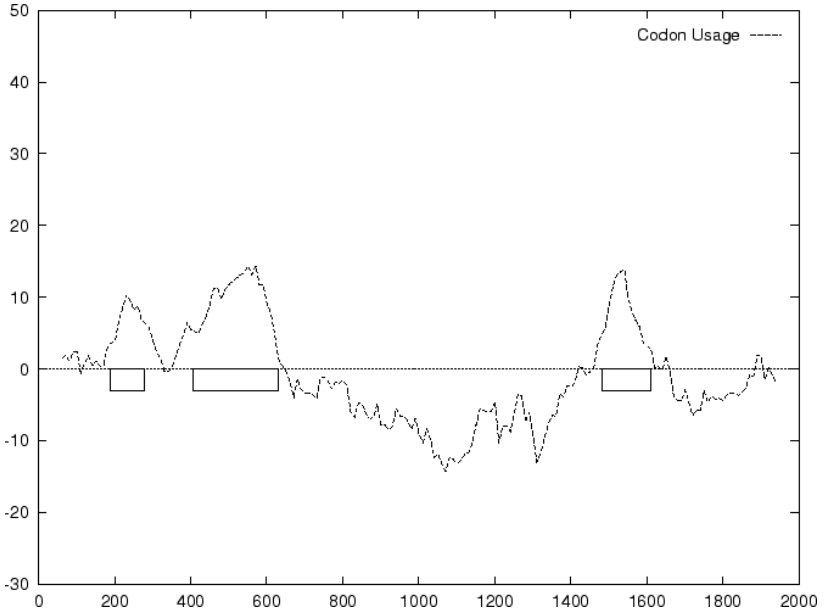


Figure 4.11 An example of coding statistic. The coding Vs non-coding model based on the codon usage along 2,000 bp of the human β-globin gene sequence (3 exons), computed on a sliding window of length 120 with step 10. Adapted from **?**.

Protein-coding regions

The distribution of amino acids in the known families of proteins is not uniform: for each species some amino acids are more common than others. Additionally, not all the sinonymous codons of the genetic code that represent the same amino acid are used in the same proportion. Both facts produce a bias in the codon usage that can be statistically measured in the known genes of each species. Obviously, such a biased distribution is not observed in intronic and intergenic regions, improving the discrimination power. At the core of most gene-finding methods are one or more coding measures that evaluate the codingness of a sequence based on the codon bias (see **?** for a review).

A coding statistic is a function that given a DNA sequence computes a real number measuring the likelihood that the sequence is coding for a protein (see Figure 4.11). The most popular coding statistic is the count of the frequency of each hexamer (two codons) in a sequence, to compare it afterwards to the frequencies observed in real protein-coding regions and non-coding regions (introns or intergenic sequences). If the content of such a region is similar to the oligomers that are more present in exons than in introns then it is reported as a predicted coding exon (**?**). Markov models are a natural form of counting these oligonucleotides to detect the dependencies between a group of consecutive nucleotides and the current one (**?**).

Other type of statistical regularities are independent of a coding model. These statistics only capture the universal features of coding DNA, not requiring a sample of real protein-coding regions. For instance, periodicities or asymmetries are typical deviations from randomess (see **?** for a review on DNA composition and codon usage).

Promoter regions

Gene promoter regions consist of clusters of binding sites, with some TFBSs oftenly occurring more than once to favour a higher rate of success in the transcription. Promoters can be therefore detected by taking advantage of this biased composition. However, there is not a general composition present in the majority of promoters, and the bias is not as strong as in the case of the coding regions.

The exact location annotation of the beginning of a transcript (the TSS) is usually very difficult. Basically, oligonucleotide counts are used in combination with other techniques to locate the TSS, as well as the upstream promoter region and the first exon (**?**). Such a region is supposed to contain a significant concentration of words representing binding site motifs. The enumerative methods to characterize promoter regions count all possible DNA words of a certain length in promoter sequences, and then evaluate statistically the results to report a list of over-represented words that could reflect the regulatory content of the sequences (**?**).

Simulating the coding and non-coding models constructed for gene prediction, similar methods have been attempted in the case of the promoter prediction. For instance, a model for promoter sequences and a model for coding exons can be used to discriminate promoters from other genic regions (**?**).

4.5 Sequence comparison

A region of DNA that is significantly similar to a known sequence is suspicious to possess a similar function. This information may be used to guide or validate the prediction process. When a genomic sequence encodes a protein with a known homolog, methods that are based on the comparison with annotated sequences are preferable (positive evidence). Conversely, a region that matches well to repetitive sequence is unlikely to contain coding regions (negative evidence). Obviously, the main drawback of such methods is the impossibility to find genes and regulatory elements that are completely different from the products in the databases.

Different sources of information can be used to establish the comparison:

- Comparison to databases of expressed sequence tags (ESTs) or complete transcripts (cDNAs), to identify regions of a contig that could correspond to a processed mRNA.
- Translation of the input genomic sequence in the six reading frames and alignment to protein databases.
- Comparison of the predicted peptide in a genomic sequence to protein databases.
- Comparative analysis with homologous genomic sequences from other organisms to identify conservations of functional elements (binding sites, exons, . . .).

Comparative genomics

The complete genomic sequence of a number of eukaryotes is already available. Therefore, it is natural to expect to extract practical results from this data. The rationale behind comparative genomic methods is that

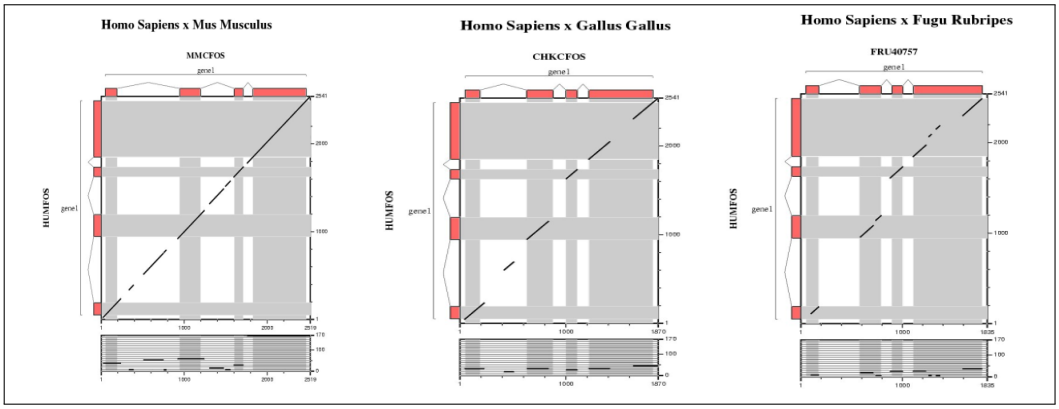


Figure 4.12 Comparative analysis of the mouse, chicken and fugu orthologs for the human *FOS* gene. The boxes in red are the coding exons in both species. The diagonal lines are conserved segments in the pairwise alignment of the genomic sequences. Notice the better discrimination of the exons in more distant species.

functional sequences (e.g. protein-coding regions, regulatory elements) tend to be more conserved than non-functional sequences in other species.

There is a lot of controversy in the scientific community about the use of the terms synteny, orthology/paralogy, homology or similarity. A syntenic region is defined to be a set of gene loci that stay together on the same chromosomic location in two or more species (**?**). As explained in Chapter 3, two sequences are homologous if both share a common ancestor (**?**). In addition, two sequences are similar when an alignment procedure reports a high degree of identity/similarity, not necessarily reflecting an evolutionary relationship (**?**).

Comparative gene prediction

When two genomes have only recently diverged, the order of many genes, gene numbers, gene positions and even gene structures (exon-intron organization, splice site usage) remain highly conserved (see Figure 4.12). Thus, gene prediction accuracy can be improved by using comparisons between two closely related genomes (**?**).

Typically, comparative gene-finding combines sequence alignment and gene prediction. In a first step, the syntenic sequences of both genomes are located by the alignment of both genomes. Due to the importance of a good detection of such sequences, the choice of the genomes to align, the programs, and their parameters is crucial (**???**).

In a second step, the gene-finding engines predict genes on these hypothetically homologous regions, enhancing the score of the predicted exons overlapping the conserved parts of both genomes (**??**).

Phylogenetic footprinting

Transcription regulation and animal diversity are intimately associated. For example, despite the number of genes in common between two different species as human and mouse is extremely high, both animals

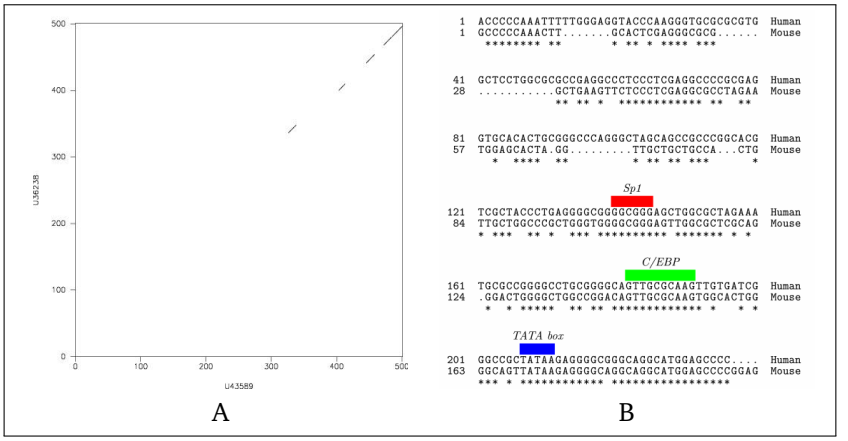


Figure 4.13 Phylogenetic footprinting (A) Dotplot of the promoter regions of the human and mouse *Leptin* gene. (B) Comparative analysis of both promoters.

present different organismal complexity. Emerging evidence suggests that a more sophisticate elaboration of the regulatory mechanisms can be the responsible of this great variability (**?**).

Comparative promoter prediction is based on the hypothesis that patterns of gene regulation are often conserved across species. Interspecies comparisons would help to identify common regulatory sequences (see Figure 4.13).

? proposed the term 'phylogenetic footprinting' to describe the phylogenetic comparisons that reveal evolutionary conserved functional elements in homologous genes. However, this promising technique also presents some caveats, such as the difficulty to select the proper pair of species to perform the comparisons as every region of the genome evolves at a different speed (**?**), the detection of specific elements of a given genome that are not present in the other one (**?**) or the existence of ultraconserved elements in the genomes of several species whose function must be determined (**?**).

Despite their limitations, phylogenetic footprinting has become very popular, being widely extended as an interesting method to locate regulatory elements (see **??** for a review).

Microarray data

The advent of the genome projects have favored the development of revolutionary techniques to process such a huge volume of information. High-throughput transcriptional profiling is definitely among these substantial improvements. DNA microarrays are the best representative of this new class of data-driven research paradigm. Microarray data measure the expression of a set of genes in two different cellular samples (knock-out vs. wild type) or after inoculation of some substance during a period of time divided into several stages.

The main principle of the method is the hybridization between unique oligonucleotides that represent a gene: one of which is immobilized on a matrix and the other is the actual RNA that is being transcribed in the sample. By fluorescently tagging each sample with different colours, the amount of transcript present in each sample can be quantified with a posterior image scanning of the hybridized microarray (see an example

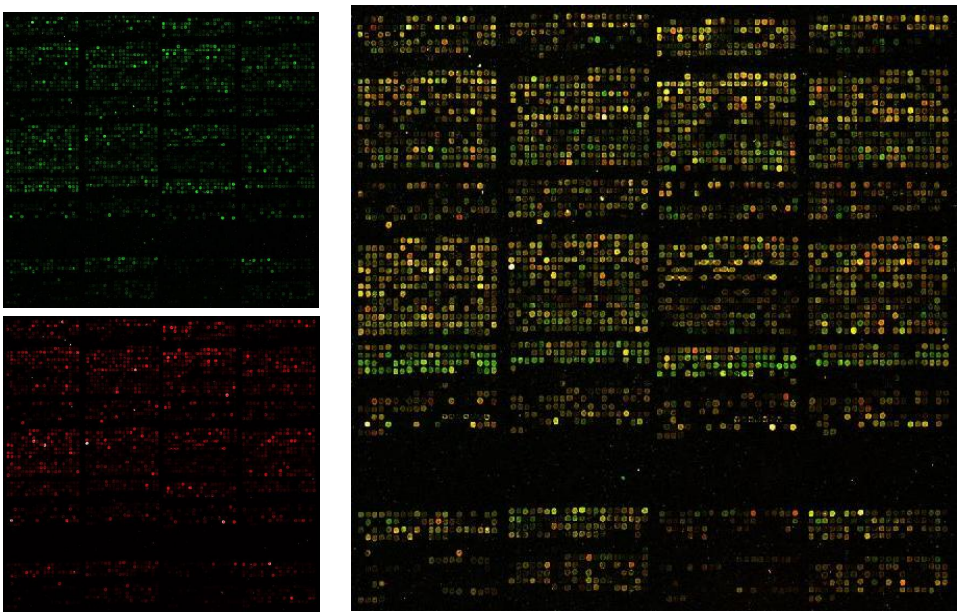


Figure 4.14 A microarray experiment. (Left) Expressed genes in a cell after a specific treatment in green and expressed genes in a normal cell in red. (Right) The ratio between both sets to detect the coexpressed genes.

in Figure 4.14).

Many different implementations of the general microarray concept have been developed. Despite the ambiguity inherent to the high volume of output information, the procedure to elaborate and perform a microarray experiment usually consists of these steps (for further details see **?**):

① Selection of the platform to construct the array

② Experiment design: choose a set of genes adequate to answer a biological question

③ Perform the experiment in the microarray (replications)

④ Image processing and estimation of the expression

⑤ Data collection and management of the gene expression data

⑥ Normalization of the expression data

⑦ Data analysis to find significant genes

⑧ Clustering the genes according to the pattern of expression

⑨ Analysis of the interesting groups (function, promoter elements, . . .)

The final result of a microarray experiment is usually a list of genes that are over-expressed or under-expressed according to the state of the cells or the tissues from which the samples where extracted. Each group of genes presenting a similar temporal pattern of expression is said to be co-regulated or co-expressed.

The guilty by association strategy states that genes exhibiting a similar pattern of expression probably possess in common a similar transcriptional regulatory mechanism or play a similar function in such a cell. Thus, co-expressed genes are mainly the target of promoter detection analysis, being also functionally characterized using some catalogue of known biological functions such as the Gene Ontology (**?**).

Since their creation, microarray technology has shown to be extremely useful to produce an enormous amount of large scale expression information. Microarrays have been applied at a genome-wide scale to build a regulatory map of *Saccharomyces cerevisiae* (**?**), to classify and discover different types of acute leukemia (**?**), to annotate the human genome (**?**), to reconstruct the transcriptional network controlled by a TF in *Drosophila melanogaster* (**?**), to study alternative splicing (**?**) or to experimentally annotate the genes controlled by a family of TFs in human (**?**). Several outstanding reviews on the topic of microarrays have been published (**??**).

Pattern discovery

Opposite to pattern matching or pattern-driven methods reviewed in Section 4.3, a new family of algorithms called sequence-driven methods appeared for searching novel motifs in a set of sequences that are hypothetically regulated in a similar manner (**?**).

Sequence-driven methods, also called pattern discovery, do not rely on the use of any external dictionary or catalogue of elements that must be searched in the sequences. Instead, this approach attempts to detect novel patterns that are conserved in the input sequences. These motifs are not expected to be exact matches so that some mismatches are allowed and positional conservation is somehow neglected during the process.

The procedure described in Figure 4.15 is based on the definition of a fitness function and the implementation of an iterative procedure to distinguish the occurrences of the novel motifs that stops when no improvement is observed. Sequence-driven algorithms have been mainly used to analyze the promoters of co-regulated genes according to microarray expression experiments. Examples are the programs AlignAce (**?**), MEME (**?**) and Gibbs sampling (**?**).

4.6 The state of the art in gene identification

In the early nineties, the first computational gene-finding programs were designed to integrate both signal and content sensors, modeled during the eighties using either linguistic methods, machine learning procedures or purely statistical approaches. These programs used to be applied on single sequences. The seminal works in this field were presented by **?** and **?**. Other members of this first generation of gene finders were: `fgeneh` (**?**), `geneid` (**?**), `genelang` (**?**), `genemark` (**?**) and `grail` (**?**).

The first exhaustive evaluation of the accuracy of those methods on a large set of vertebrate sequences with simple gene structure was published by **?**. The results indicated that the predictive accuracy of the programs analyzed was lower than originally expected (the average percentage of exons exactly identified was less than 50%). This low accuracy level was in part explained because of the limited number of sequences used in the training process. Some of the basic accuracy measures used in the field are described in Table 4.1.

At the end of the last decade, a second generation of programs appeared simultaneously with the completion of the first genome sequencing projects. Some of them were even used in the earlier stages of the annotation pipelines. As new data and more powerful computers became accessible, the gene finders were

```
Pre ≡ S_1, S_2, ..., S_n: sequence; M: motif model; F: scoring function;

    (* Select a random pool of motifs in the sequences to create M *)
    M ← CreateInitialModel();
    (* Evaluate the fitness of the current model M *)
 5: score_0 ← EvaluateModel(M, F);
    score ← score_0;
    (* Repeat until convergence in the model M *)
    while score ≥ score_0 do
      score_0 ← score;
10:   (* Alter the model, trying to locate the motifs in each sequence *)
      UpdateModel(M);
      score ← EvaluateModel(M, F);
    end while
    (* Use the new model M to search the best motifs on each sequence *)
15: for i ← 1 to n do
      PatternDriven(S_i,M);
    end for
```

Figure 4.15 Sequence-driven algorithms.

SHORT	NAME	DESCRIPTION
TP	True positives	Number of real positive examples correctly predicted
TN	True negatives	Number of real negative examples correctly predicted
FN	False negatives	Number of real positive examples not correctly predicted
FP	False positives	Number of real negative examples not correctly predicted
SN	Sensitivity	Proportion of real examples corresponding to any prediction: $\frac{TP}{TP+FN}$
SP	Specificity	Proportion of predictions supported by any real example: $\frac{TP}{TP+FP}$
CC	Correlation coefficient	Correlation between SN and SP: $\frac{(TP \times TN)-(FN \times FP)}{\sqrt{(TP+FN)\times(TN+FP)\times(TP+FP)\times(TN+FN)}}$

Table 4.1 The common accuracy measures in sequence analysis.

able to deal with sequences containing more than one gene. Examples of programs in this second generation of gene prediction tools include: `geneid` (**?**), `genie` (**?**), `genscan` (**?**), `hmmgene` (**?**) and `mzef` (**?**).

Moreover, it was evident that sequence similarity to external databases containing known examples (search by homology) should be incorporated into the scoring schema of the programs in order to reinforce the predictions. This paradigm was developed in programs such as `genewise` (**?**), `grail-exp` (**?**) or `procrustes` (**?**). Some of these approaches were evaluated by **?** and **?**. Although the gain in accuracy was significant in short sequences containing one gene, the performance was still insufficient in long semi-artificial sequences constructed from annotated examples.

Nowadays, after the completion of the first draft of the human genome we are completely immersed in a context of genomic research. The current generation of gene finders is devoted to the automatic reannotation of genomes by using the increasing amount of new information. Comparisons between genomes have proven to be very helpful in the discovery of novel genes (**?**). Some representatives of the current generation of gene prediction programs are `fgenesh+` (**?**), `geneid` (**?**) and `genomescan` (**?**), or the comparative analysis systems `doublescan` (**?**), `rosetta` (**?**), `slam` (**?**), `sgp1` (**?**), `sgp-2` (**?**) and `twinscan` (**?**).

The latest achievements in the sequencing of other higher eukaryotes have allowed the advent of comparative predictors that consider the alignment of multiple genomes in the prediction model, such as `N-scan` that simultaneously combines the genomes of human, mouse, rat and chicken (**?**). Moreover, new tools such as `jigsaw` (**?**) and `gaze` (**?**) for the assembly of data obtained from external sources of prediction and experimental evidence have been recently developed.

geneid

The current version of `geneid` (**?**) is a program that predicts genes in anonymous genomic sequences designed following a simple hierarchical structure (see Figure 4.16 (A)). First, splice sites and start and stop codons are predicted and scored along the sequence. Next, potential exons are constructed from these sites and scored as the sum of the defining sites plus the score of a Markov model for coding DNA. Finally, from the set of predicted exons, the gene structure maximizing the sum of the score of its exons is assembled using a dynamic programming algorithm (**?**).

`geneid` offers two features to integrate external information into the ab initio predictions: (1) sequence homology information can be used to reinforce the predictions that are supported by the alignment and (2) partial or complete genes obtained from other sources can be incorporated before the exon assembly.

As a consequence of its simple design, `geneid` has been also parallelized. Parallelism of data (distribution of data among processors with shared memory) was finally implemented because it was the best solution for distributing the overload in the system. Following the divide and conquer strategy, the best gene structures computed in different processors are assembled introducing some overlap between sequence fragments (see Figure 4.16 (B)).

The simplicity of the architecture of `geneid` is appropriate to deal with problems different from the canonical ones. Taking advantage of the implemented facilities to reannotate sequences, `geneid` has been the main component of two recent genome annotation pipelines:

① Identification of novel selenoproteins in eukaryotes. The presence of a secondary structure (SECIS element) in the 3' UTR of the mRNA induces the UGA codon, usually a termination signal, to be translated as Selenocysteine. `geneid` was modified to permit the dual meaning of the UGA triplet, being succesfully applied to describe the *Drosophila melanogaster*, human and *Takifugu rubripes* selenoproteomes (**???**). In addition, `geneid` was used to reannotate selenoproteins in the *Tetraodon nigroviridis* genome (**?**), being the first eukaryotic genome project to integrate the identification of this particular family into the gene annotation pipeline.

② Comparative gene prediction. `sgp2` is a method to predict genes in a target genome sequence using the sequence of a second informant or reference genome (**?**). Essentially, `sgp2` is a framework to integrate the search program `tblastx` results with `geneid` predictions. The result of the `tblastx` alignment of two sequences is used by `geneid` to rescore the exons supported by the alignment, penalizing the score of the others. `sgp2` was successfully used in cooperation with another similar

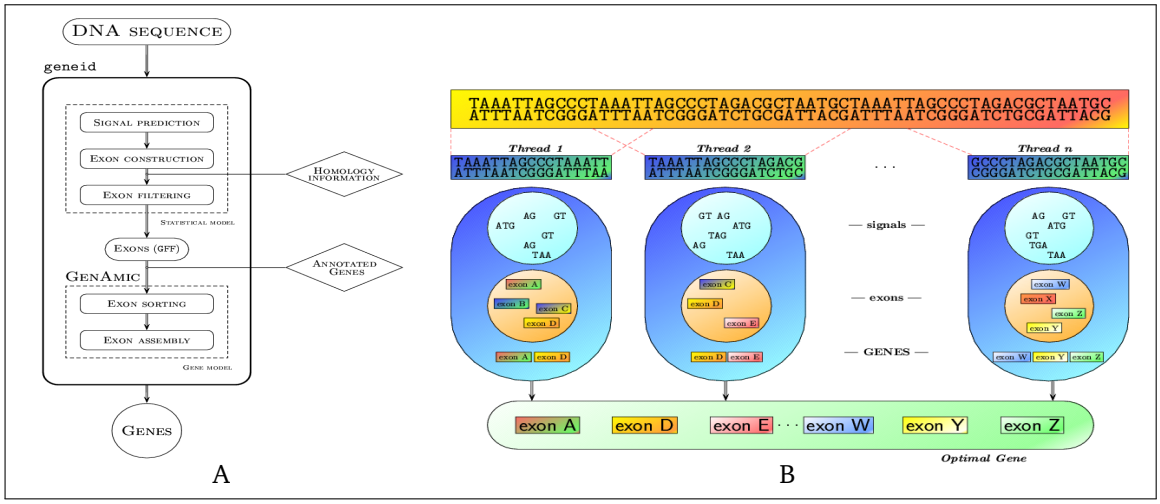


Figure 4.16 `geneid` dataflow. (A) The serial dataflow. (B) The parallel dataflow.

program called TWINSCAN (**?**) to discover a set of novel human and mouse genes. A subset of them was then experimentally validated in a subsequent stage of the genome comparison protocol (**?**). The same protocol was used to annotate the genomes of human and chicken (**?**).

4.7 The state of the art in promoter characterization

The first algorithms of sequence alignment were enterely written to analyze proteins (**?**). However, it was soon noticed that the same procedures could be applied over any type of biological sequence, including transcription regulatory regions. For instance, **?** used consensus and similarity searches to locate some general promoter elements in a set of vertebrate sequences. In (**?**), two algorithms to detect a common motif that can be known or unknown a priori in a set of sequences were presented. Later, these algorithms were used to characterize the core promoter of several *Escherichia coli* genes (**?**).

Consensus are a rudimentary form for representing regulatory sites so that new proposals to overcome their limitations were published. **?** suggested the use of weight matrices. These PWMs were constructed from previous alignments of different types of biological sites. **?** systematically refined and tested the PWMs for detecting different regulatory signals such as the TATA box, the CAAT-box or the GC-box. At the same time, theoretical studies to relate the information content and the quality of anchored alignments were already published (**?**). Posterior studies have shown the low specificity of the PWMs when the set of initial examples is small (**?**).

Soon, several databases to store the experimental examples and the constructed matrices were published, such as TRANSFAC (**?**). At the same time, efficient programs to scan promoter sequences based on the pattern matching technique (pattern-driven approaches) were designed to use these matrices, being `MatInspector` the most popular one (**??**). However, methods to identify TFBSs in a single sequence demonstrated a very poor performance with an excess of false positives. Certain improvements were observed when using additional information. New heuristic methods to discover unkown patterns in a set of

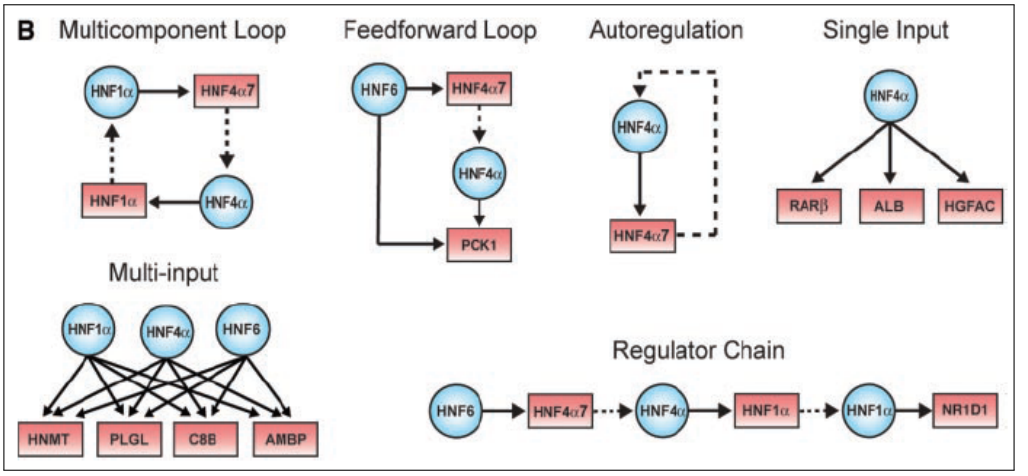


Figure 4.17 Transcriptional regulatory module architectures. Regulatory proteins and their gene targets are represented as blue circles and red boxes, respectively. Solid arrows indicate protein-DNA interactions, and genes encoding regulators are linked to their protein products by dashed lines. Adapted from (**?**).

regulatory sequences appeared (sequence-driven approaches): the application of the Gibbs sampling (**?**) and the expectation-maximization method (**?**) are good examples.

In general, however, the experimental investigation of a single promoter in all cell types where it can be active, under all conceivable conditions, at all possible developmental and cell-cyle stages, is in practice impossible. With this limitation in mind, the predictions obtained by any method must be always very carefully evaluated to avoid the rejection of predicted functional sites that have not been experimentally annotated yet.

The identification of the core promoter regions and the annotation of the TSSs have also been two problems associated to the problem of the TFBSs prediction. The presence of significantly over-expressed words or an unusual high percentage of CpG dinucleotides have traditionally been two measures of promoterness. For instance, **?** combined these two sensors with splicing detection to locate the first exon of a gene, predicting therefore the TSS position. Neural networks and genetic algorithms were used in (**?**) to discriminate between promoter and non-promoter sequences. **?** reviewed the topic, showing the poor accuracy of most methods in the detection of the TSS. Word over-representations have been also used to study the association of adjacent TFBSs to form regulational modules or clusters with interesting results although the deciphering of a regulatory code seems still too complex (**????**). An example of such architectures is shown in Figure 4.17.

A new revolution in the study of gene regulation began with the availability of genomic information and the possibility to work with abundant expression data. Phylogenetic footprinting, for instance, is a new form of leaving a great fraction of false positives out (**??**). Promising results have been obtained in several investigations (**???**). A review on phylogenetic footprinting can be found in (**?**). Gene expression data from microarrays is the other great hope in the field to elaborate a regulatory map of human. Despite at the beginning, there was a boom of analysis of such data in different biological problems (**???**), the difficulty to analyze and understand such an amount of data has been underscored in many occasions, though. The new generation of arrays based on chromatin immunoprecipitation promise to be an interesting method of prediction validation (**?**). The combination of comparative genomics and expression data will become in a

few years the standard way to study a group of genes as in (**?**).

Due to the poor results obtained when analyzing sequences to find pure binding motifs, intensive research has been performed in other areas to understand better the gene regulation problem. For instance, the association between CpG islands and promoters (**?**), DNA structure (**?**), nucleosome positioning (**?**) or protein-DNA physical interactions (**?**).

Similarly to the gene-finding accuracy tests, several assessments have been performed about the quality of promoter characterization tools, always with discouraging results. The lack of stable data sets of regulation sites, and the surprising difficulty to deal sometimes with orthologous sequences are two causes that suggests the need for further improvement (**??**).

4.8 Looking forward

Despite the numerous advances in the basic algorithms of gene and promoter prediction and the unceasing flow of new data, the way to determine the exact number of genes in the human genome remains unclear (**?**) and the elaboration of a regulatory map of the human genome seems today an objective too ambitious (**?**).

In the discipline of gene prediction, the same concepts have been applied since more than 20 years ago. While the basic gene models have been improved to support comparative research, the definition of a gene predicted by a gene-finder is still the same. It is true that some non-canonical gene structures are being slowly incorporated into the programs such as prediction of UTRs, alternative splicing forms or selenoproteins (**?**). Right now, the gene identification problem is still open and many efforts are engaged in the creation of a solid catalogue of human genes (**?**), in which large-scale experimental methods of validation will be crucial (**?**).

Moreover, gene prediction and promoter recognition should be performed simultaneously. Unfortunately, we are far from reaching such an achievement due to the poor performance in the detection of regulatory elements despite the new and promising research that is currently being done in that direction (**?**). The enormous volume of high-throughput expression data has provided new opportunities in the investigation of the biology of the systems (**?**). Phylogenetic footprinting is also demonstrating their capability to unveil regulatory blocks conserved in several species (**?**). In addition, more accurate catalogues of annotated regulatory elements are appearing, making the training of new pattern discovery methods easier. All together will be part of a future pipeline to automatically identify and annotate the eukaryotic promoter regions. However, much effort must be still invested in understanding better other aspects of the same biological problem such as chromatin effect, methylation, or nucleosome movement (**?**).

Perhaps a new line of thought should be established in both fields (**?**). So far, we have been only focusing on the sequence and many successful advances have been possible following such an approach. However, it is assumed that the cell machinery works in many levels with uncountable number of interactions that we have not incorporated in our systems yet. Once we have reached the limit with the current methods, and that moment is not too far, it will be essential to move from the current analytical systems to more constructive and dynamic applications, emulating the mechanisms of the cell.

PART III

Meta-Alignment of Sequences

Chapter 5

Multiple Non-Collinear TF-map Alignment

Summary

The generalization of the pairwise TF-map alignment is presented here. First, the formal definition of a multiple map alignment and how to compute the optimal score is provided. Next, we use a progressive approach to build up a multiple alignment in a stepwise manner. Then, we have studied how to break the non-collinearity property inherent to the alignments produced by dynamic programming techniques. Results on biological data indicate that multiple TF-map alignments are able to locate regulatory elements in several promoters that are not conserved at sequence level.

5.1 The need for multiple TF-map alignment

SEQUENCE COMPARISONS ARE ONE OF THE MOST IMPORTANT COMPUTATIONAL TOOLS in molecular biology. Sequences are good symbolic representations of biological molecules that encode relevant information about their structure, function and history. From the analysis of several related sequences, biologically significant facts can be inferred. For instance, genomic sequence comparisons are performed in order to identificate genes or regulatory sites across different genomes, as these functional elements tend to exhibit conservational patterns different from those observed in regions that are not functional.

In attempt to allow for multiple sequence comparisons, the basic dynamic programming recurrences introduced in the 1970s to align efficiently two sequences of n symbols in $O(n^2)$ (**??**), can be naturally extended for k sequences, with an exponential cost $O(n^k)$ (**?**). As this cost is unaffordable in practice, many heuristics have appeared to provide acceptable solutions with a minor cost. The most popular of them is the hierarchical or clustering method (**??**).

This procedure, also called progressive alignment, is a greedy algorithm that runs in $O(k^2n^2)$ time. In a first step, this method performs all of the pairwise alignments to build an evolutionary tree. In a second step, an initial alignment is constructed from the two closest sequences, incorporating then the rest to the profile following the guide tree. Such a procedure does not guarantee to find the optimal solution in mathematical terms. However, the results are generally in good agreement with the biological problem of aligning correctly bases of homologous functional elements. See Chapter 3 Section 3.5 for a comprehensive review of this topic.

Progressive alignment has also commonly used in the genome-wide alignment methods that perform rapid multiple genomic alignments to identify conserved biological features between distant species. Basically, these algorithms identify local similarities between two genomes that are then used as anchors to align the interleaving regions (**?**). The progressive technique is then combined with these genome pairwise aligners to build up the multiple genome alignment (**??**).

These comparisons at the sequence level have limitations however. Although similar sequences do tend to play similar biological functions, the opposite is not necessarily true. Often similar functions are encoded in higher order sequence elements that are not necessarily conserved at the sequence level. As a result, similar functions are frequently encoded by diverse sequences which are undetectable by conventional sequence alignment methods.

Gene promoter regions are a good example. The information that governs the RNA synthesis is mostly encoded in the gene promoter, a region normally 200 to 2,000 nucleotides long upstream of the transcription start site of the gene (TSS). Transcription factors (TFs) bind to sequence specific motifs (the TF binding sites, TFBSs) within the promoters. TFBSs are 5−8 nucleotides long and one promoter region contains on the order of 10 to 50 of them (**?**). Such motifs appear to be arranged in specific configurations that define the temporal and spatial transcriptional pattern program of each gene. Genes presenting similar expression patterns are assumed to share similar configurations of TFBSs in their promoters. However, TFBSs associated to the same TF are known to contain sequence substitutions, being in many cases completely different. Promoter regions of genes with similar expression pattern may not be similar at the sequence level, even though they may be co-regulated.

In the previous chapter (**?**), we suggested the existence of regulatory information conserved between related promoters that could not be detected at the sequence level. Let Σ_{TF} be the alphabet of TFs denoting symbols. We initially defined the process of mapping a nucleotide sequence into a sequence in Σ_{TF} (the TF-maps). Then, we developed an efficient algorithm to obtain the global pairwise alignment between two TF-maps (**?**). Finally, we showed the TF-map alignments were more accurate than conventional sequence

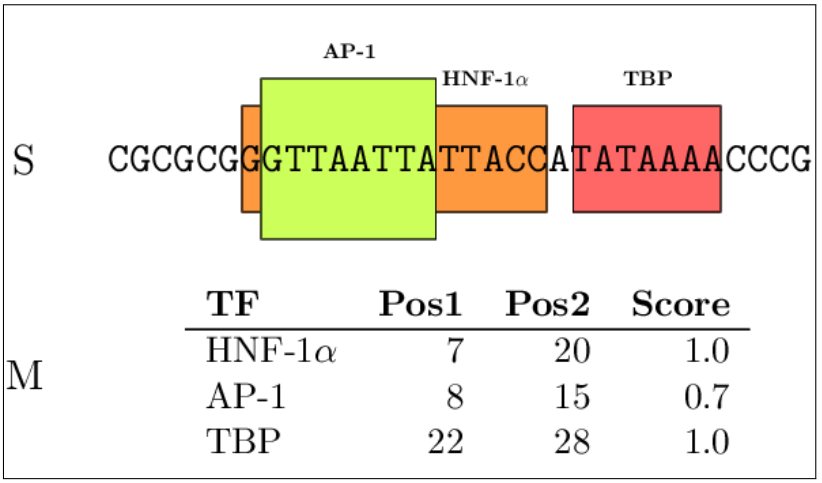


Figure 5.1 TF-mapping in a simple example.

alignment to distinguish pairwise gene co-expression in a collection of microarray results (**?**).

In this chapter, we present an efficient implementation of the multiple TF-map alignment based in the progressive alignment paradigm. We have introduced some modifications in the pairwise global TF-map alignment algorithm to align two clusters of TF-maps, eventually allowing non-collinear arrangements of TFBSs in the results without additional cost. Most dynamic programming global alignments rarely cope with the presence of rearrangements observed in the DNA, being only partially identified by combining global and local alignment strategies (**??**). This problem is particularly relevant in the case of the regulatory regions, where non-collinear configurations of TFBSs are prone to be conserved (**?**).

The structure of the chapter is the following: first, we briefly reviewed the concept of mapping functions and provide the formal definition of a multiple TF-map alignment. Then, we introduce the main algorithm that performs the progressive alignment of multiple TF-maps. Next, we detail the algorithm to compute the optimal pairwise alignment of two clusters of maps. Later, we define formally a non-collinear alignment, introducing some modifications in the pairwise algorithm to allow the detection of these cases. Finally, we systematically estimate the optimal parameters of the alignment to distinguish promoters from other gene regions in a set of well characterized human-rodent gene pairs and their corresponding orthologs in chicken and zebrafish. These results are compared to those obtained by conventional sequence alignment methods, showing the validness of our approach. Several particular examples are presented in which multiple TF-map alignments characterize conserved regulatory elements that are otherwise imperceptible in sequence-level comparisons.

5.2 Basic definitions

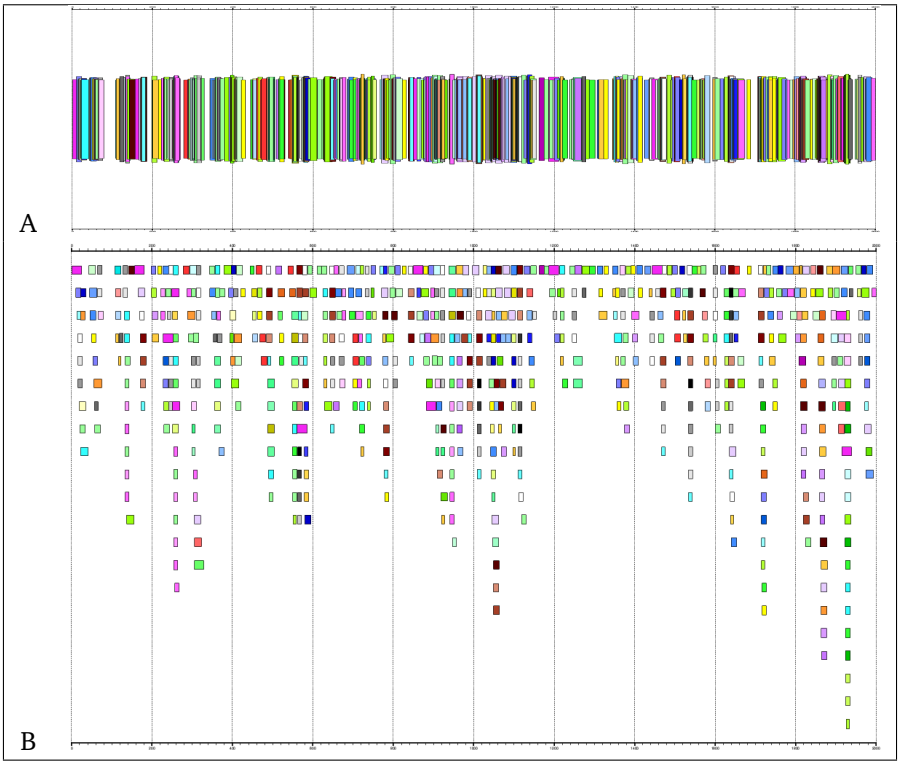


Figure 5.2 TF-mapping of the human promoter NM_015900 (500 nucleotides). (A) Condensed representation of the TRANSFAC predictions. (B) The same set of predictions displayed in a non-overlapping format.

Mapping a promoter sequence into a TF-map

Let Σ_{DNA} be the alphabet of four nucleotides. Let Σ_{TF} be the alphabet of TFs denoting symbols. In a previous work (**?**), we defined a mapping function as a procedure to translate a promoter region $S = s_1 s_2 \dots s_k$ where each nucleotide $s_i \in \Sigma_{DNA}$, into a sequence of TF-tuples $M = m_1 m_2 \dots m_n$ where each TF-tuple $m_i =< m_i^f, m_i^{p1}, m_i^{p2}, m_i^s >$ denotes the match of a binding site for the TF $m_i^f \in \Sigma_{TF}$ occurring between the position m_i^{p1} and the position m_i^{p2} over the sequence S with score m_i^s. Different mapping functions can be used to obtain the translation from S to M such as a collection of weight matrices representing TFBSs (JASPAR (**?**), PROMO (**?**) or TRANSFAC (**?**)). For each match over a given threshold, we register a new TF-tuple in M defined by the label (m_i^f) of the TF associated to the PWM, the positions (m_i^{p1}, m_i^{p2}) and the score (m_i^s) of the match (see Figure 5.1, for an example). Other mapping functions can used instead, such as pattern discovery programs that identify a set of unknown motifs conserved in several promoters (e.g. MEME (**?**)).

Matches are annotated at a given location irrespective of their orientation in which they occur. This translation preserves the order of S in M, that is if $i < j$ in M then $(m_i^{p1} < m_j^{p1})$. Matches to different TFs may possibly occur at the same position, being false positives in most cases (see a real example in Figure 5.2). We refer to the resulting sequence of TF-tuples M as a Transcription Factor Map, or simply a TF-map.

Multiple alignment of TF-maps

Let $M_1, M_2, \ldots, M_k$ be a set of TF-maps where each map is denoted as $M_i = m_{i,1} m_{i,2} \ldots m_{i,|M_i|}$ and each TFBS is denoted as $m^f_{i,j} \in \Sigma_{TF}$. Let $M^*_1, M^*_2, \ldots, M^*_k$ be the extended set of TF-maps where each map is denoted as $M^*_i = m^*_{i,1} m^*_{i,2} \ldots m^*_{i,|M^*_i|}$, and each TFBS is denoted as $m^{*f}_{i,j} \in \Sigma_{TF} \cup \{-\}$. The symbol $'-'$ indicates a gap, which can be considered as a particular TF-tuple $<' -', \cdot, \cdot, \gamma>$ where the value $\cdot$ is a null value and γ is the penalty for introducing a gap in a column of the alignment.

The alignment of k maps $M_1, M_2, \ldots, M_k$ is then a correspondence T, maybe empty, among the extended maps $M^*_1, M^*_2, \ldots, M^*_k$ such that:

1. The extended maps have the same length.
2. If the gaps are removed from each M^*_i, we recover M_i.
3. At least one element in a column is different from a gap.
4. The elements that are aligned in a column correspond to the same TF.
5. No overlap in the primary sequence is permitted between adjacent sites in the alignment.

Note that the first three conditions define the classical multiple alignment of sequences. Last two conditions, however, introduce two new constrains that are related to the match state and the non-overlapping property, according to the notion of pairwise TF-map alignment provided in (**?**).

The score of a multiple alignment of TF-maps

A multiple TF-map alignment –or simply, a multiple map alignment (MMA), in contrast to a multiple sequence alignment (MSA)– can be also represented as a rectangular array:

$$T = \begin{pmatrix} m^*_{1,1} & m^*_{1,2} & \ldots & m^*_{1,t} \\ m^*_{2,1} & m^*_{2,2} & \ldots & m^*_{2,t} \\ \ldots & & & \ldots \\ m^*_{k,1} & m^*_{k,2} & \ldots & m^*_{k,t} \end{pmatrix}, \tag{5.1}$$

where each column $T(i) = (m^*_{1,i}, m^*_{2,i}, \ldots, m^*_{k,i})$ is the multiple match among the TF-tuples in position i from $M^*_1, M^*_2 \ldots M^*_k$. Given the multiple alignment T, we compute the score $s(T)$ of the MMA as:

$$\begin{aligned} s(T) = & \quad \alpha \textstyle\sum_{i=1}^{t} \sum_{j=1}^{k} m^{*s}_{j,i} \\ & - \quad \lambda(g) \\ & - \quad \mu \textstyle\sum_{\forall i,i'} f(m^{*p1}_{1,i} - m^{*p1}_{1,i'}, m^{*p1}_{2,i} - m^{*p1}_{2,i'}, \ldots, m^{*p1}_{k,i} - m^{*p1}_{k,i'}) \end{aligned} \tag{5.2}$$

where $\alpha, \lambda, \mu > 0$, g is the number of columns with only one element different from a gap in the MMA (unaligned elements), and f is a function that measures the conservation of distance between the sites of every map in two consecutive columns (i, i') with more than one aligned element in the MMA. That is, the score of the alignment increases with the score of the aligned elements and the penalty of the gaps (α), and decreases with the number of unaligned elements (λ), and with the difference in the distance between adjacent aligned elements (μ). See the previous chapter and **?** for further details about the TF-map alignment parameters.

5.3 The algorithms

There are many possible alignments between a group of TF-maps. The optimal alignment is the one scoring the maximum among all possible alignments. In a previous work (**?**), we implemented a dynamic programming algorithm to obtain such an alignment efficiently for the case of two TF-maps. The optimal multiple sequence alignment problem (and therefore also the multiple alignment of maps) is, however, much more difficult, being formally a NP-complete problem (**?**).

Here, we propose to adapt the popular progressive alignment strategy to the TF-map alignment. The solutions obtained by this method are not guaranteed to be optimal. However, multiple progressive alignments usually have an underlying biological explanation (**?**). We have also introduced some changes in the basic pairwise TF-map alignment algorithm developed in (**?**), in order to deal now with two clusters of MMAs instead of two single TF-maps.

Progressive MMA algorithm

Let $(G_1 \dots G_k)$ be the initial list of k TF-map groups, where each group contains a single TF-map. Let S be the similarity matrix where $S(G_i, G_j)$ denotes the similarity between the TF-map groups G_i and G_j.

The progressive MMA algorithm shown in Figure 5.3 builds up a multiple TF-map alignment in a stepwise manner. In a first step, all pairwise TF-map alignments are performed. The initial multiple alignment is created with the two most similar ones. Both maps are substituted for a new group that contains their alignment. The similarity between this new cluster and the rest of the TF-maps is then estimated, updating tha S matrix (see Implementation).

In a second step, an iterative procedure selects at each round the pair of clusters that are more similar from the pool of available groups. These two groups are aligned and merged again into a new TF-map cluster, estimating the similarity to the remaining ones. At the end of the process, there is only one group that contains the progressive alignment of the input TF-maps.

The cost of the progressive MMA can be expressed in terms of the number of pairwise TF-map alignments that must be computed. Let k be the number of maps to be aligned and n be the length of each map. The initial round performs $O(k^2)$ pairwise alignments. Next, the progressive round performs $O(k)$ alignments involving two groups. Let $P(n)$ be the cost of each pairwise operation, then the cost of the progressive alignment algorithm is $O(k^2 \cdot P(n))$. The expected value of $P(n)$ is calculated in the next section.

Implementation

In the progressive MMA algorithm shown in Figure 5.3, the variable *maxSim* saves the maximum score so far computed at each round. The group identifiers of such a score can easily be retrieved using a supplementary pair of variables *iSim, jSim*.

The pairwise TF-map alignment algorithm called *ComputePairwiseSimilarity* (**?**) has been slightly modified to accomodate the alignment of two TF-maps groups, as explained in the next section. The optimal pairwise alignments between the input TF-maps in the initial round are saved, as they could be required during the iterative procedure.

```
Pre ≡ G: list of TF-map groups (G_1 ... G_k)

    (* Initial Step: pairwise alignment all Vs all *)
    maxSim ← −∞
    for i = 1 to k do
 5:   for j = i + 1 to k do
        S(G_i, G_j) ← ComputePairwiseSimilarity(G_i, G_j);
        (* Select the pair with maximum similarity *)
        maxSim ← max(maxSim,S(G_i, G_j));
      end for
10: end for
    (* Create a new group: estimate the similarity to others *)
    G_iSim−jSim ← MergeGroups(G_iSim, G_jSim);

    (* Progressive Step: cluster the two most similar groups *)
15: while |G| > 1 do
      maxSim ← −∞
      for i = 1 to |G| do
        for j = i + 1 to |G| do
          (* Select the pair with maximum similarity *)
20:       maxSim ← max(maxSim,S(G_i, G_j));
        end for
      end for
      (* Create a new group: estimate the similarity to others *)
      G_iSim−jSim ← MergeGroups(G_iSim, G_jSim);
25: end while
```

Figure 5.3 Progressive multiple map alignment algorithm.

Once a new TF-map group is created from the two most similar ones, their binding sites must be merged (function *MergeGroups*). The order of the TFBSs in the new group must take into account the position of the binding sites in their primary promoter sequences. In the approach here, we do not create a profile of each MMA. Instead, all of the TFBSs of each group are always available for subsequent TF-map alignments.

The alignments between this new TF-map group and each one of the rest of the groups are not explicitly computed. The similarity among them is instead estimated with the WPGMA method (Weighted Pair Group Method with Arithmetic Mean) according to the previous similarity between the groups G_{iSim} and G_{jSim} to the others. If an alignment between two groups whose similarity was estimated before is identified as the most similar during the progressive step, the MMA must be explicitly computed before merging both TF-map groups.

The alignment of two clusters of MMAs

Let $G_x = m_{x,1} m_{x,2} \dots m_{x,|G_x|}$ and $G_y = m_{y,1} m_{y,2} \dots m_{y,|G_y|}$ be the two most similar groups of TF-maps in the current round of the progressive alignment. Let S be the scoring dynamic programming matrix where $S(i,j) = S(m_{x,i}, m_{y,j})$ denotes the similarity of the best TF-map alignment of the groups $G_x = m_{x,1} \dots m_{x,i}$

and $G_y = m_{y,1} \dots m_{y,j}$, according to the scoring function in Equation 5.2. The *ComputePairwiseSimilarity* algorithm explained here is a generalization of that developed in (**?**) to align two TF-maps that computes the optimal pairwise TF-map alignment between G_x and G_y.

This algorithm basically searches the the maps of both groups to find matches between one site in one group and one site in the other. Once a new match is identified, the previous matches must be evaluated in order to construct the optimal alignment ending at this one (see Figure 5.4). Because this class of scoring matrices are highly sparse, we register the coordinates in S of the matches computed previously. Thus, to compute the optimal score at the cell $S(i,j)$, only the non-empty cells in S that are visible for the current match need to be accessed. In addition, we maintain the list sorted by optimal score, so that the cell scoring the maximum value is at the beginning of the list and, in most cases, only a few nodes will need to be accessed before a critical node is reached beyond which the optimal score can not be improved (**?**).

The number of computations $P(n)$ in this algorithm is very similar to that obtained in the conventional pairwise TF-map alignment algorithm (**?**). The exact complexity of this algorithm is difficult to be studied –depending mostly on the size of the input maps and the sparsity of the resulting marix S. An expected time cost analysis reveals that the cost function can be explained in terms of (a) a first quadratic term derived from the obligatory comparison between all of the TFBSs of both maps to detect the match cells and (b) a second quadratic term necessary to search for each match the best adjacent previous pair in the optimal TF-map alignment. In (**?**), we studied the contribution of using a list of non-empty cells in S that reduces the second component to an expected cost of $O(p \cdot n^2)$, where p is the percentage of the matrix that is occupied. This value was estimated to be below 5% of occupancy for the pairwise TF-map promoter comparisons.

Implementation

In the pseudocode in Figure 5.5, the groups G_x and G_y are represented as two arrays of sites sorted by the position in their promoters, where each site corresponds to an input TFBS. The multiple TF-map alignment of a cluster is internally encoded with pointers among the sites that form each match. Gaps here are not explicitly represented.

Each site $m_{x,i}$ is a structure as described above with the functions *factor, pos1, pos2 and score* returning the values of the corresponding fields. The variable *maxSim* stores the optimal score so far computed. The sites in the optimal TF-map alignment can be easily retrieved using a supplementary structure *path(i,j)* that points to the previous cell in the optimal path leading to cell $S(i,j)$. In addition, the function *ComputeInitialSimilarity* calculates for each match $S(i,j)$ the initial score of a hypothetical alignment that includes only the sites $m_{x,i}$ and $m_{y,j}$.

Once the match between two sites $m_{x,i}$ and $m_{y,j}$ has been identified, the best previous match between two other sites $m_{x,i'}$ and $m_{y,j'}$ is used to construct the new alignment (see the matches A and B in Figure 5.4). The list L is used to locate the non empty positions in S. Each node of the list L is represented as structures p and n with the functions *abscissa* and *ordinate* returning the corresponding coordinates in S of each previous match.

The score of the new match between $m_{x,i}$ and $m_{y,j}$ is the sum of the scores of the columns in which both elements were aligned in their respective MMAs. Unaligned sites are scored with the gap penalty γ. The function *ComputeLambda* counts the number of sites in each group that are not included in the alignment, taking into account the size of each group. The function *ComputeOverlap* calculates the average distances D1 and D2 between any pair of consecutive matches in the maps of both groups, verifying the absence of physical overlap in their promoters. The function $|D1 - D2|$ scores the conservation of distance between the sites of every map in two consecutive columns in the MMA (function f, see Equation 5.2).

5.4 Non-colinear TF-map alignments

The existence of regulatory elements that are conserved in different order between related promoter regions is documented, specially in enhancers (**?**). Even at the sequence level, the identification of these DNA rearrangements is very difficult. We have here introduced some subtle changes in the pairwise TF-map alignment algorithm shown before to deal with non-collinear alignments. The aligned TFBSs in such MMAs are therefore not necessarily located in the same relative order in every map.

Definition

Let T be an alignment between two TF-maps M_1 and M_2 formally defined as a correspondence $T = \{(m_{1,I_1}, m_{2,J_1}), \ldots, (m_{1,I_t}, m_{2,J_t})\}$. Let $(m_{1,i}, m_{2,j})$ and $(m_{1,k}, m_{2,l})$ two matches in T ,not necessarily contigous, with $i < k$. Then, we define the collinearity or non-collinearity of T in terms of the ordering between j and l, for all the match pairs of T as:

1. If $j < l$ then T is a collinear alignment
2. If $j > l$ then T is a non-collinear alignment (see example shown in Figure 5.6 (Left).

The generalization of this definition for $k > 2$ TF-maps is immediate (see the example of a non-collinear MMA for $k = 3$ TF-maps in Figure 5.6 (Right).

The algorithm

The non-collinear matches shown in Figure 5.6 can not be detected in the basic pairwise TF-map alignment algorithm. Let A and B be two TF-maps in which two matches could form a non collinear alignment (represented as a circle and a square in Figure 5.7). The normal implementation fills in the matrix row by row, from top to bottom (or column by column, from left to right). According to this, when the first match is being processed (red square), the second one (red circle) is not still available (green area). On the contrary, when the second match is processed, the first one is not accessible as the basic algorithm only allows the search for best previous aligned elements in the list of computed values that are in the area delimited by the current match.

To overcome such a limitation, we propose to compute the optimal values of the matrix S following a different order, to allow the visibility of one of these elements (circle) by the other (square). For instance, the top-bottom diagonal filling of the matrix depicted in Figure 5.7 may process in first position the element that was not visible before (circle) for the other element (square) that will computed later in the next diagonal (square). While this strategy still produces the same aligments obtained with the ordinary implementation, non-collinear alignments produced by new combinations of matches can also be formed.

Adjusting the non-collinearity

Non-collinear conservation of regulatory elements is documented in very specific cases (**?**). Most upstream promoter regions, however, are constituted of collinear arrangements of TFBSs. Because of the poor speci-

ficity of the collections of PWMs (**?**), many non-collinear alignments produced with the algorithm described above are simply artifacts.

Thus, we have designed a simple mechanism to adjust the frequency of non-collinear aligned sites in the output. As the function *ComputeOverlap* in the algorithm above needed to be redefined in order to detect non-overlap between non-collinear matches as well, we have introduced an additional parameter c to weight those alignments involving non-colinearity.

The following example is graphically presented in Figure 5.8 (Left). Let A and B be two TF-maps in which a previous match has been identified (represented as a circle). Then, a second match between an element in A and another in B is being processed (the squares). The dotted lines indicate that such a site in B can be located either on the left or on the right of the circle site in the same map. In the first case, a non-collinear alignment is produced; in the second case, a normal collinear alignment is constructed.

The algorithm to align two clusters of TF-maps must be slightly modified to accomodate the non-collinearity parameter c (the case in which the non-collinear match occurs in A can be similarly defined):

$$z = \begin{cases} \text{if } (D_2 < 0) & \rightarrow \mu|D_1 - c \cdot D_2|, c \geq 1 \\ \text{if } (D_2 \geq 0) & \rightarrow \mu|D_1 - D_2| \end{cases} . \tag{5.3}$$

The optimal positional conservation between both matches occurs when $d_1 = d_2$. However, the parameter c is used into the μ penalty to punish only those matches that do not respect the collinearity of the current alignment (the square site is on the left of the circle site in B, see Figure 5.8).

Informally, if $c = 1$ then both collinear and non-colinear matches are indistinctly combined into the resulting MMA. High values of c, however, produce a higher amount of collinear matches into the results. In order to establish formally the behaviour of this parameter, we have count the number of non-collinear matches in the TF-map alignment of the human and mouse promoters (500 nucleotides) of the MMP13 gene (REFSEQ entries NM_002427 and NM_008607). In Figure 5.8, there is a clear correspondence between the amount of inversions in the MMA and the value of c. No inversions are produced for large values of c.

Identification of non-collinear configurations of TFBSs in regulatory regions is poorly known. We recommend, therefore, to use this option very carefully. In addition, we also suggest the use of a small set of matrices to perform the mapping, which can reduce the number of artifacts in the resulting non-collinear MMA.

5.5 Biological results

The optimal MMA of a set of TF-maps is obviously dependant on the values of the $\alpha, \lambda, \mu, \gamma$ and c parameters. In addition, the optimal parameter configuration is likely to depend on the particular problem to be addressed (orthologous genes or co-regulated genes in microarray experiments), and the particular protocol to map the TFBSs on the sequences.

Results in the previous chapter (**?**), indicated that TF-maps alignments are able to characterize promoter regions of co-regulated genes in absence of sequence similarity. Thus, TF-map alignments were shown to detect high-order regulatory signals conserved in a collection of related promoters that were undetectable

HRCD SET	Multiple TF-map alignment		CLUSTALW	
	TOP1	Avg.Score	TOP1	Avg. score
CODING	9	18.61	28	3706.72
5'UTR	2	11.80	4	2671.78
PROMOTER	21	27.81	4	2005.67
INTRONIC	3	9.75	0	1359.19
DOWNSTREAM	1	10.53	0	1174.28
INTERGENIC	0	7.84	0	1052.92

Table 5.1 Results when distinguishing promoters with MMAs.

for current sequence alignment methods. It is important to mention that two different TFBSs can be aligned if they correspond to the same TF, irrespectively of their sequence motifs.

Here we have conducted a similar systematic training over an extended set of orthologous promoters for obtaining the optima configuration. In order to verify the ability of MMA to identify regulatory elements that are rarely detected in conventional comparisons, we have compared the results to those obtained by global sequence alignment methods. In addition, we have focused on three specific examples to show the abilities of MMA in the characterization of co-regulated gene promoters. In all of the cases, we have only constructed collinear map alignments as non-collinear regulatory rearrangements have not been reported on them.

Multiple TF-map training

For the pairwise TF-map alignment, we estimated the optimal parameters in a set of experimentally characterized human and rodent gene promoters (**?**). Here we have extended such a dataset by searching the corresponding orthologs in chicken and zebrafish as well. Using the REFSEQ (**?**) gene set as mapped into the UCSC genome browser, we have correctly identified the ortholog in both species, if available. We refer to the resulting set of human-mouse-chicken-zebrafish homologous genes as the HRCZ SET. This dataset contains 18 human-rodent-chicken-zebrafish orthologs, 7 human-rodent-chicken orthologs, 4 human-rodent-zebrafish orthologs, and 7 human-rodent orthologs.

The lack of available collections of experimentally verified TFBSs is an important limitation for the evaluation and the training of phylogenetic footprinting systems. Despite several databases of annotations and promoter sequences have recently appeared (**??**), there is not a minimum amount of regulatory information conserved among species other than human and mouse to train the MMA on them.

Thus, we can not repeat the training procedure used in (**?**) to evaluate the ability of MMA to detect conserved regulatory elements at larger evolutionary distances –at which the degree of conservation may be negligible. However, we can use another method, also presented in (**?**), to show that MMAs are much more informative than primary multiple sequence alignments.

We first have mapped the TFBSs occurrences in the promoter sequences using the collection of 50 most informatives matrices in JASPAR 1.0 (**?**), to which we refer as $\text{JASPAR}_{\text{TOP50}}$ (**?**).

Then, we have compared the MMAs obtained in the 200 nucleotides of the promoter region of the 36 gene pairs from the HRCZ SET, with the MMAs obtained in fragments of 200 nucleotides from intergenic (2,000 nucleotides upstream of the TSS), 5'UTR (downstream of the TSS), coding (downstream of the translation start site and considering only coding DNA), intronic (downstream of the first intron junction), and downstream (downstream of the transcription termination site) sequences (see Figure 5.9 for a graphical representation of the test). We have computed the average score of the MMA on each one of the genomic re-

gions and have identified, for each orthologous set, the genome regions in which the alignment produces the highest score. We have performed the same exercise using global pairwise sequence alignments (obtained with CLUSTALW, (**?**)).

We have repeated this test using different combinations of parameters. Systematically, the parameters α, λ and μ were allowed to independently take values between 0.0 and 1.0, in incremental steps of 0.1. At the same time, the parameter γ (gap penalty) was tested between 0 and -10. The optimal parameter configuration is considered to be that set of parameter values that better discriminate between promoters and the rest of genomic regions.

Results appear in Table 5.1. As expected, nucleotide sequence alignments score the highest in the coding regions (in 28 out of 36 cases), followed by the alignments in the 5′ UTR regions (4 out of 36) and in the promoters (4 out of 36). The scores of the sequence alignments show that promoter regions are less conserved than coding regions, and 5'UTRs. Despite this, the optimal MMA configuration in the collinear configuration $(\alpha = 1, \lambda = 0.1, \mu = 0.1, \gamma = -2)$ scores the highest in the promoter regions (in 21 out of 36, see Table 5.1). In addition, the average score of map alignments is notably higher than that of the coding regions. Only in 9 out of 36 cases the TF-map alignments score the highest in coding regions. Interestingly, while intron sequences in the human-mouse-chicken-zebrafish orthologs are much less conserved than 5'UTRs, MMAs score the highest in intronic regions in 3 cases whereas they only score the best in 5'UTRs in 2 cases. This is consistent with the fact that first introns are known to often contain regulatory motifs.

Finally, we have also performed a complementary test to measure the specificity of the TF-map alignments. As a negative control, we have shuffled the orthologous associations in the HRCZ SET to construct a pool of unrelated human-mouse-chicken-zebrafish 36 gene entries. Then, the corresponding multiple TF-map alignments of these non-orthologous paired promoters were obtained using the parameters previously optimized. The TF-map alignments of the unrelated promoters of each entry were significantly worse with an average score more than 50% smaller than TF-map alignments that involved "bona fide" orthologous promoters. For instance, the average score of the TF-map alignments among orthologous promoters when using the $\text{JASPAR}_{\text{TOP50}}$ collection was 27.81. In contrast, the score of the TF-map alignments between non-related promoters was 12.51. The sites in the alignments involving non-orthologous gene promoters may hypothetically correspond to general regulatory elements present in most core promoters. An alternative, more probable, hypothesis is that they reflect the poor specificity of most PWMs representing TFBSs.

Promoter characterization

We have selected three examples to show the ability of MMAs to characterize promoter regions in the absence of sequence conservation. In the three cases, we have compared the multiple TF-map aligment against the corresponding multiple sequence alignment produced by CLUSTALW, as in the section above.

All of the cases are graphically represented as pictures in which the input TF-maps are displayed on the upper part of the picture and the resulting MMAs are displayed on the lower part of the picture, using the `gff2ps` program (**?**).

As it is possible to see, the main effect of the MMA is the dramatic reduction in the number of predicted TFBSs that typically result after a PWM-based search (see Figure 5.10 and Figure 5.11). For instance, we aligned 157 human sites to 197 mouse sites, 229 chicken sites and 167 zebrafish sites mapped in the respective Actin α-cardiac gene promoter orthologs (see next section). The resulting multiple TF-map alignment only contained 14 TFBSs, which approximately represents a 13-fold reduction. Graphically, this reduction is noticed in the smaller density of aligned sites in the resulting MMAs picture.

In addition to this, most aligned sites in the MMAs are concentrated in the proximal promoter region of

each gene (200 nucleotides upstream of the TSS). This gain in specificity is not simply due to the selection of an arbitrary set of non-overlapping TFBSs, as many experimentally annotated TFBSs on these promoters are successfully covered by the MMAs.

Actin α-cardiac gene

Actins are highly conserved proteins that are involved in various types of cell motility. The alpha actins are found in muscle tissues and are a major constituent of the contractile apparatus. The *Actin α-cardiac* gene has been identified in many kinds of cells including muscle, where it is a major constituent of the thin filament, and platelets.

The promoter of the human and mouse *Actin α-cardiac* genes (ACTC, GENBANK entries M13483 and M26773) have been extensively characterized by experimental means (**?**). In the ABS database (**?**), the entry *A0028* informs about the known orthologous binding sites in the respective human and mouse promoters (500 nucleotides, the position +501 is the TSS). The human ACTC promoter is constituted of three SRF sites $(+301, +352, +392)$, a SP1 site $(+418)$, a MYOD site $(+445)$ and a TATA box $(+469)$. Using the REFSEQ gene annotations, we have also identified the corresponding orthologous promoters in chicken and zebrafish (REFSEQ entries NM_001031229 and NM_214784).

We have then aligned the four promoters and compared the resulting MMA with the functional annotations detailed above. In general terms, the multiple TF-map alignment of the four orthologous promoters of ACTC contains many of the functional sites in human and mouse, detecting as well the corresponding orthologs in the other species. The output coverage is, however, smaller than 50% of the promoter nucleotides.

The MMA of the ACTC promoters is shown in Figure 5.10 (Top). While the region proximal to the TSS is not more dense in predicted TFBSs than other regions, most of the aligned elements cluster near to the TSS. In addition, the alignment agrees well with the functional annotation available in human and mouse, providing novel orthologous sites in chicken and zebrafish:

1. The second SRF binding site is correctly identified in human, mouse and also in zebrafish.
2. A RREB-1 site that overlaps the SP-1 active site is identified in the MMA. RREB-1 and SP-1 are both members of the zinc finger protein families (**?**).
3. A SQUA site that overlaps the third SRF active site is identified in the MMA. SQUA and SRF are both members of the MADS family (**?**).
4. A novel forth SRF binding site is located immediately upstream of the experimental first one at the four species.
5. The TATA box is correctly detected in human, mouse and zebrafish as well.

No significant conservation among the sequences was, however, detected in the CLUSTALW multiple alignment of the four ACTC promoters (data not shown).

Myoglobin gene

The *Myoglobin* gene is a member of the globin superfamily and is expressed in skeletal and cardiac muscles. The encoded protein is a haemoprotein contributing to intracellular oxygen storage and transcellular facilitated diffusion of oxygen.

The promoter of the *Myoglobin* gene in human (MB, GENBANK entry X00371) and in mouse (REFSEQ entry NM_013593) have been experimentally characterized (**??**). In the ABS database (**?**), the entry A0037 informs about the known orthologous binding sites in the respective human and mouse promoters (500 nucleotides, the position +501 is the TSS). The human MB promoter is constituted of a CCAC box (+272), a MEF-2 site (+335) with two surrounding E-boxes (+326, +348) and a TATA box (+469). Using the REFSEQ gene annotations, we have also identified the corresponding orthologous promoters in chicken and zebrafish (REFSEQentries NM_203377 and NM_200586).

We have then aligned the four promoters and compared the resulting MMA with the functional annotations detailed above. The multiple TF-map alignment of the four orthologous promoters of MB contains several of the functional sites in human and mouse, detecting some of the orthologs in the other two species. The output coverage is again very small.

The MMA of the MB promoters is shown in Figure 5.10 (Bottom). Most of the aligned elements are present near to the TSS, while this spatial trend is not observable at the predictions at each promoter. The alignment also contains several of the functional human and mouse sites, providing their counterparts in chicken and zebrafish:

1. A RREB-1 site that overlaps the functional CCAC box is identified in the MMA. In fact, the RREB-1 matrix consensus in JASPAR represents an A/C rich area that contains the CCAC motif (**?**).
2. The TATA box is correctly detected in the four species.

The CLUSTALW multiple alignment of the four MP promoters did not reveal any significant conservation (data not shown).

Collagenase-3 gene (MMP13)

The two previous examples have been extracted from the HRCZ SET. We have now focused on another gene with a more complete set of identified orthologous promoters to test the ability of the MMAs to elucidate high-level conservation even at more phylogenetically distant sequences.

The *Collagenase-3* (MMP13) gene is a member of the matrix metalloproteinase family. MMP13 plays a major role in normal tissue remodeling processes, being abnormally expressed in breast carcinomas and in cartilage from arthritic patients (**?**). Many experimental studies have confirmed the presence of several functional binding sites for known TFs in human and mice (**???????**).

Here, we have analized the proximal promoter regions of MMP13 in human, chimp, mouse, rat, cow, dog, chicken, zebrafish and *Xenopus* (Ortín *et al.*, personal communication). As the 5'UTR of this gene is very small in most cases, we have considered the region 500 bps immediately upstream the ATG (Translation Start Codon) as the proximal promoter.

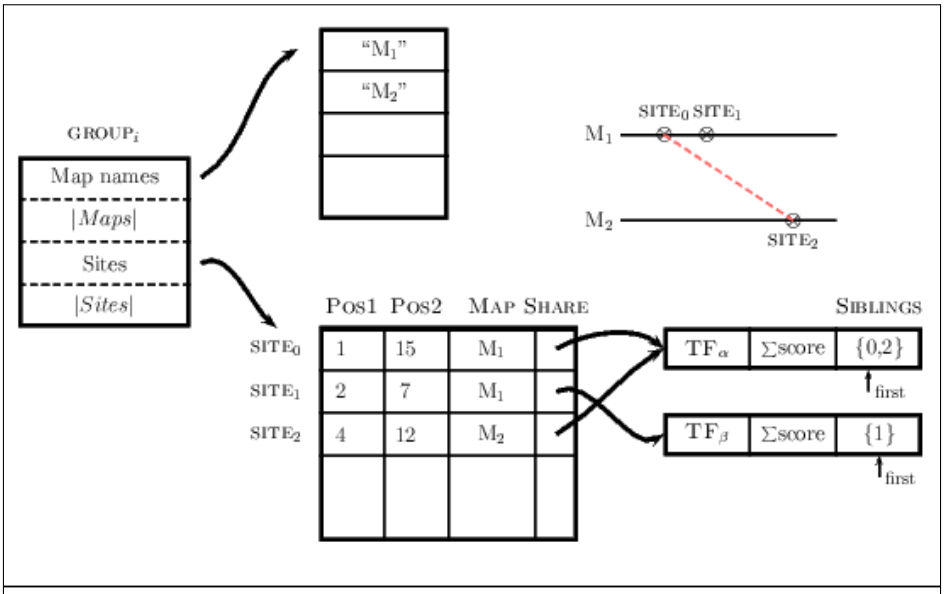
We performed the multiple TF-map alignment of the nine MMP13 promoters with the optimal configuration calculated in the previous section for four species, increasing the μ parameter to 0.75 to highlight only those regulatory elements that can be aligned in similar positions in most promoters. We also performed the multiple sequence alignment of the nine promoters with the program CLUSTALW. The MMA and the CLUSTALW alignments are both shown in Figure 5.11.

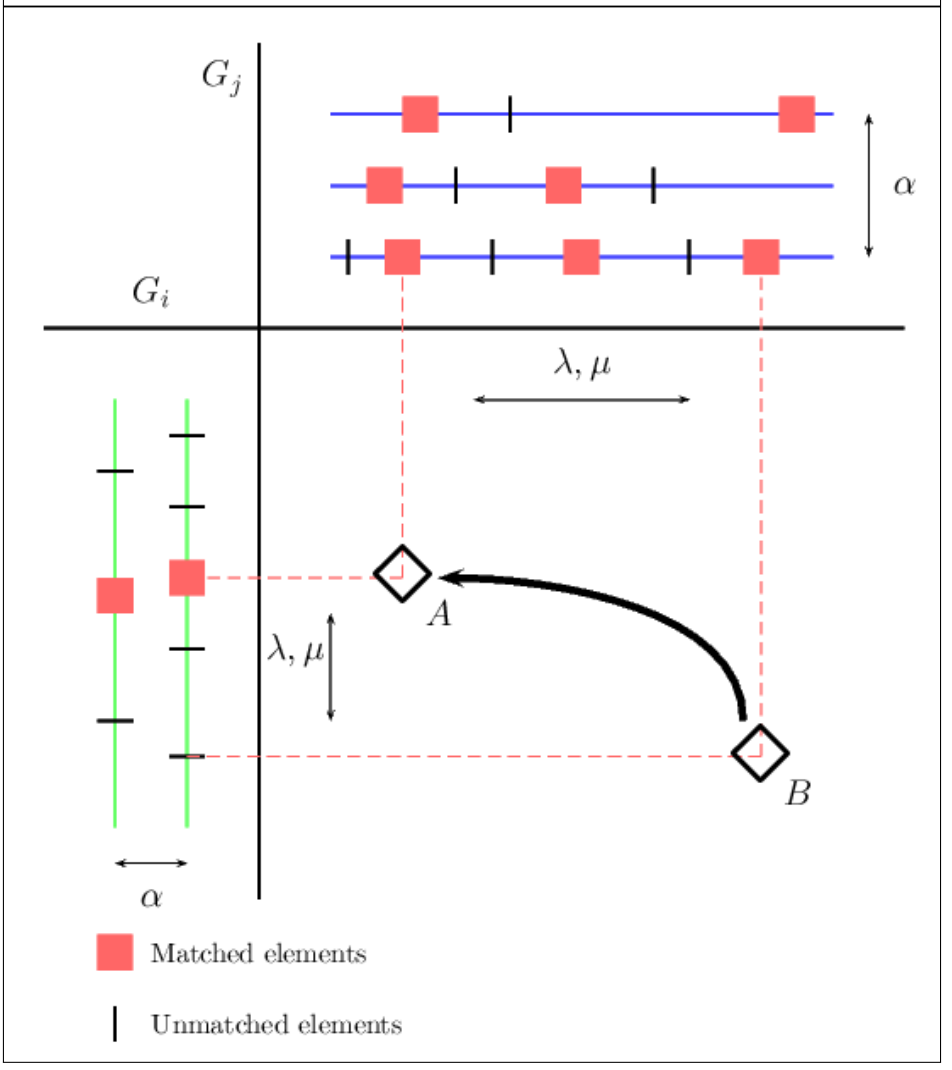
The comparison between the the resulting MMA shown in Figure 5.11 (Top) and experimental annotations on MMP13 gene promoter reveals interesting results. Up to four TFBSs that have been experimentally reported to be functional in human and mouse are remarkably included in such a MMA:

1. The AML-1 binding site included in the resulting MMA (position 330 in human promoter; alternative names: CBFA-1, OSE-2, OSF-2) (**???**).
2. The FREAC-4 binding site (position 370 in human promoter; alternative names: FREAC, p53) (**?**).
3. The SPI-1 binding site (position 391 in human promoter; alternative names: AP-1, ETS, PEA-3) (**???**). The SPI-1 transcription factors are distant related members of the Ets family (**?**).
4. The TCF11-MafG binding site (position 420 in human promoter, alternative names: AP-1) (**???**). The human transcription factor TCF11 is known to bind to a subclass of AP1-sites (**?**).

We have not only detected the human and mouse experimental binding sites but we have also identified with the MMA the putative novel site of each TF in most orthologs of the other species, including the most distant ones. The first aligned TF in the MMA (FREAC-3), which has not been experimentally detected so far, presents a similar positional conservation in all of the orthologs. In addition, the resulting phylogenetic tree constructed from the progressive multiple TF-map alignment (shown in red, left) correlates well with the real phylogeny of these nine species.

Accurate inspection of the the global sequence alignment by CLUSTALW in Figure 5.11 (Bottom) only reveals some weak conservation blocks that could partially contain any of the functional TFBSs detected by the multiple TF-map alignment. We also tested several configurations of CLUSTALW (adjusting the gap open and gap extension penalties). However, we did not found any parameter combination that was able to clearly detect all of the four functional sites.


GROUP$_i$
Map names
|Maps|
Sites
|Sites|
"M$_1$"
"M$_2$"
SITE$_0$ SITE$_1$
M$_1$
M$_2$
SITE$_2$
Pos1 Pos2 Map Share
Siblings
SITE$_0$ 1 15 M$_1$
SITE$_1$ 2 7 M$_1$
SITE$_2$ 4 12 M$_2$
TF$_\alpha$ Σscore {0,2}
first
TF$_\beta$ Σscore {1}
first


G_j
G_i
α
λ, μ
A
B
λ, μ
α
Matched elements
Unmatched elements

```
Pre ≡ G_x, G_y: TF-map groups, L: list of <abscissa,ordinate>, L = ∅
    (* Calculating the element i, j in S *)
    for i = 0 to |G_x| − 1 do
      for j = 0 to |G_y| − 1 do
        if factor(m_{x,i}) = factor(m_{y,j}) then
 5:       S(i, j) ← ComputeInitialSimilarity(m_{x,i}, m_{y,j});
          x ← α (score(m_i) + score(m_j));
          (* Searching the best previous match in L *)
          p ← first(L);
          i′ ← abscissa(p);
10:       j′ ← ordinate(p);
          while end(L) = FALSE and S(i′, j′) + x > S(i, j) do
            (* Compute the μ value and check overlap *)
            (D_1, D_2,overlap) ← ComputeOverlap(i, i, j, j′, G_x, G_y);
            if overlap = FALSE then
15:           y ← λ (ComputeLambda(i, i, j, j′));
              z ← μ(|D_1 − D_2|);
              maxSim ← S(i′, j′) + x − y − z;
              if maxSim > S(i, j) then
                S(i, j) ← maxSim;
20:           end if
            end if
            p ← next(L);
            i′ ← abscissa(p);
            j′ ← ordinate(p);
25:       end while
          n ← CreateNewNode(i, j);
          InsertNode(n, L);
        end if
      end for
30: end for
```

Figure 5.5 Pairwise alignment of two clusters of TF-maps.

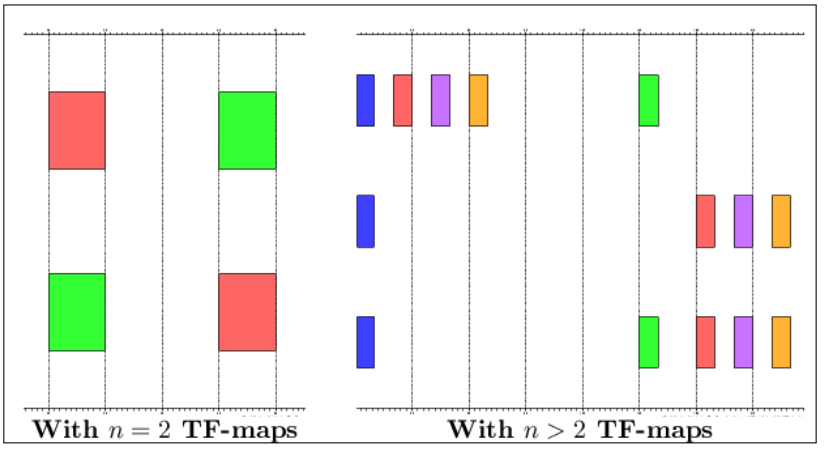


Figure 5.6 Two examples of non-collinear MMAs. (Left) A pairwise non-collinear TF-map alignment. (Right) A non-collinear MMA.

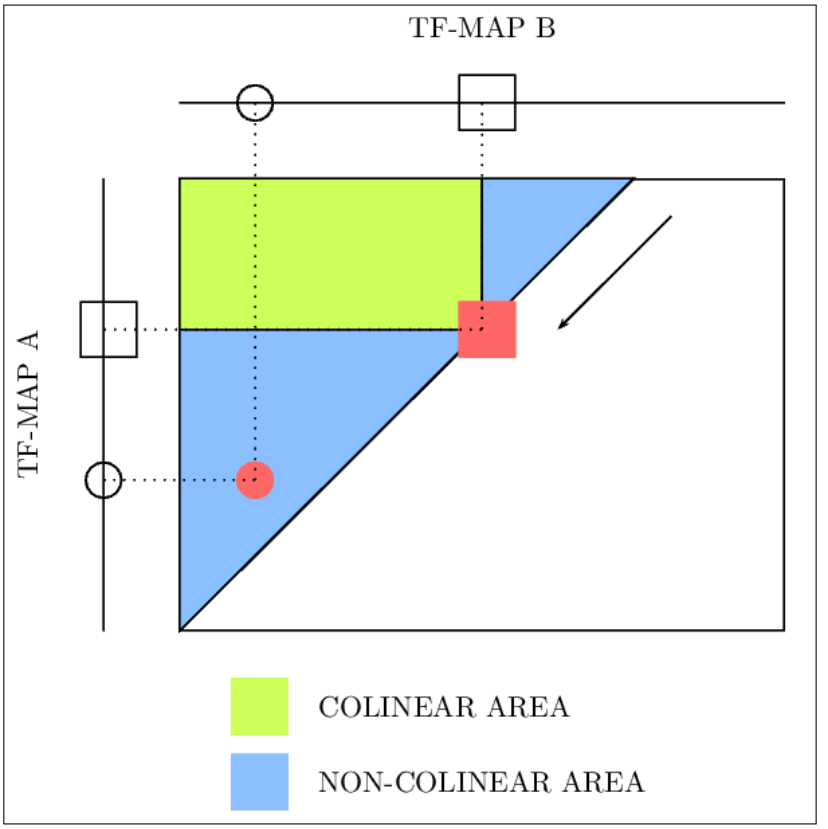


Figure 5.7 Diagonal filling of the alignment matrix.

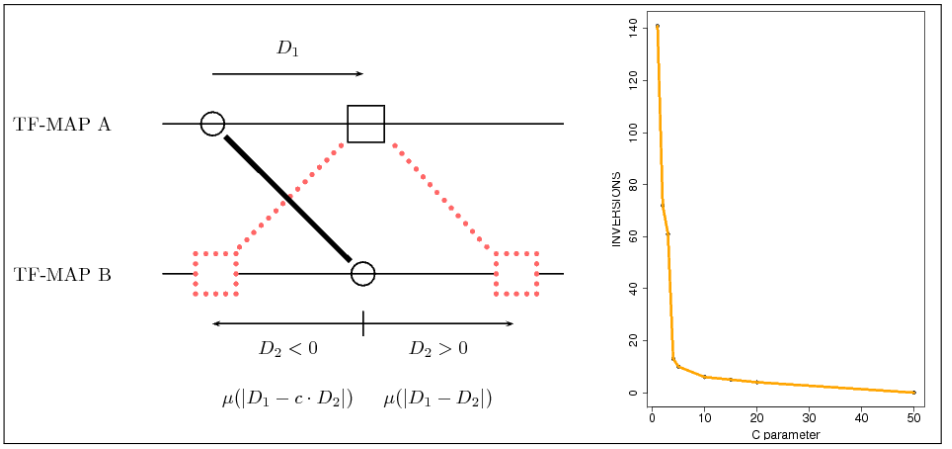


Figure 5.8 **The non-collinearity parameter.**

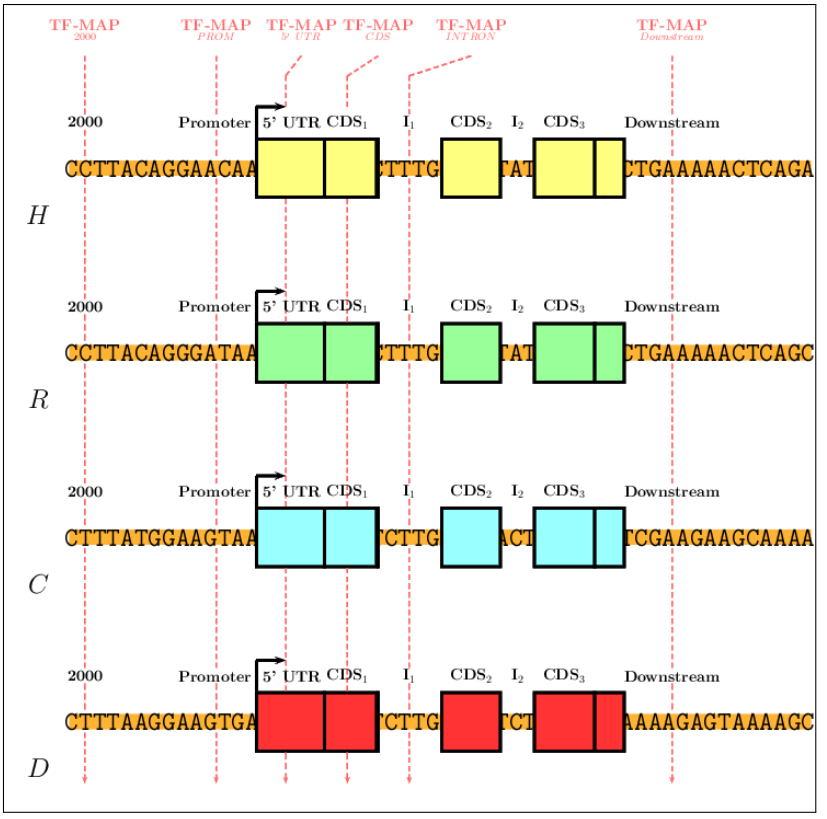


Figure 5.9 **Distinguishing promoters from other genomic regions.**

Figure 5.10 Multiple promoter characterization. (Top) JASPAR predictions and the MMA among the *Actin α-cardiac* gene promoters. (Bottom) JASPAR predictions and the MMA among the *Myoglobin* gene promoters.

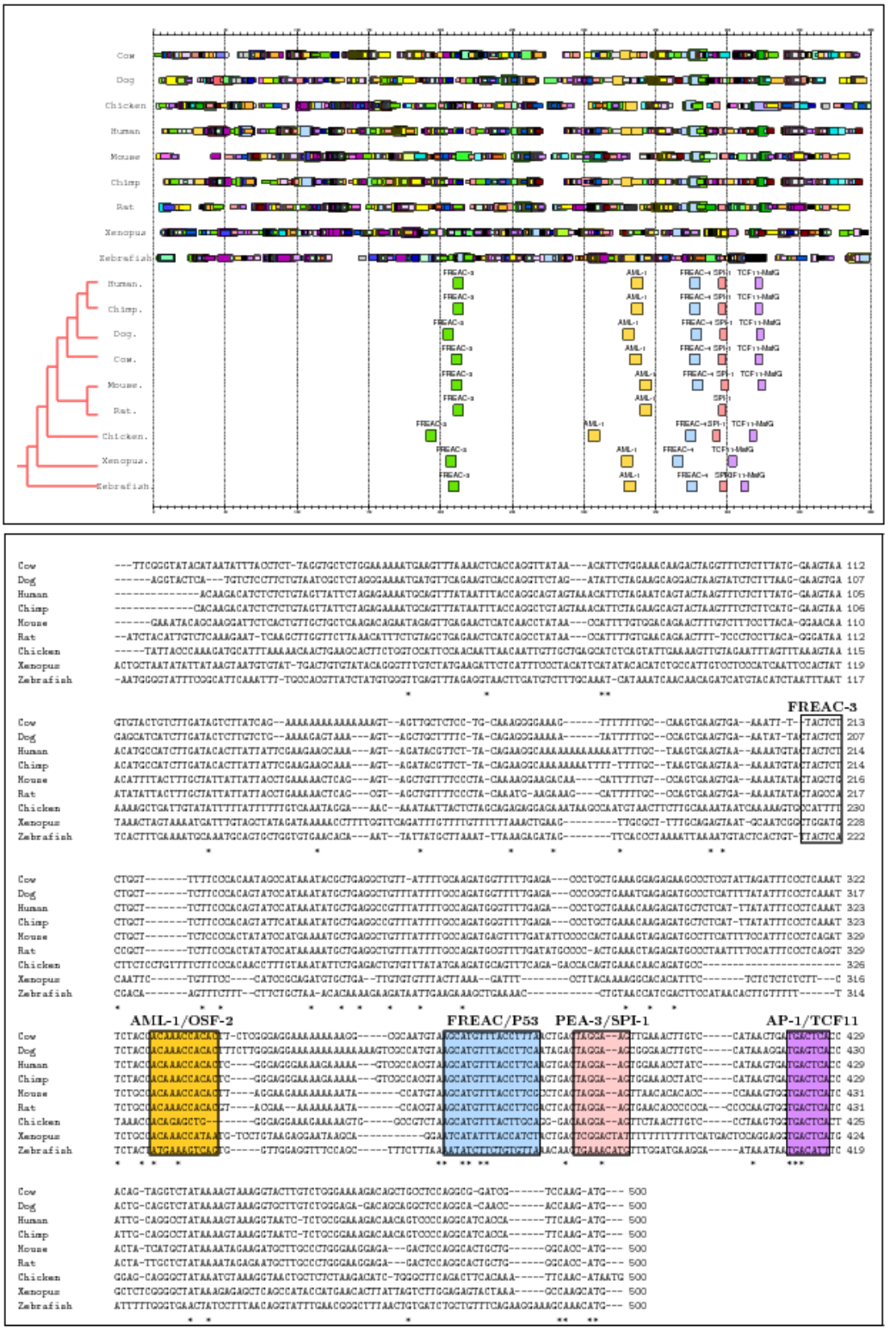


Figure 5.11 MMA of the MMP13 promoter in 9 species. (Top) JASPAR predictions and the resulting multiple TF-map alignment. (Bottom) The CLUSTALW multiple sequence alignment of the 9 promoters.

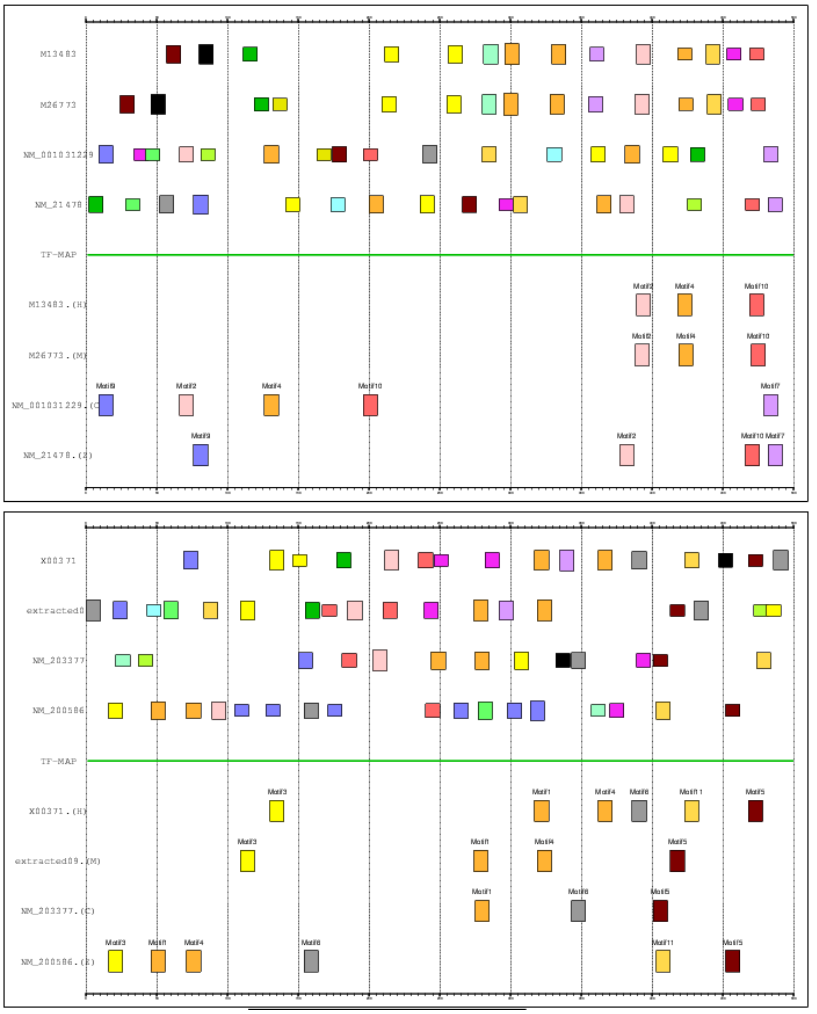


Figure 5.12 Using MEME as a mapping function. (Top) The MEME motifs and the resulting MMA in the Actin α-cardiac orthologous promoters. (Bottom) The MEME motifs and the resulting MMA in the Myoglobin orthologous promoters.

Chapter 6

Conclusions

THE TF-MAP ALIGNMENTS CAN BE VERY USEFUL to efficiently perform searches of promoter elements that might be conserved in different species. In short, the research presented here has contributed to improve the computational characterization of gene transcription regulatory regions in the following aspects:

① We have designed a new family of algorithms, which are named TF-map alignments or simply meta-alignments, to detect conserved high-order configurations of functional elements that do not show discernible sequence conservation. The meta-alignment algorithm does not directly compare the primary sequences. Instead, the algorithm aligns the map of high-level elements obtained with an external mapping function over the original sequences, taking into account their position, the element class and the mapping score.

② We have generalized the pairwise meta-alignment algorithm to deal with multiple maps. We followed a progressive approach in which the multiple meta-alignment is build up in a stepwise manner: a first multiple alignment is created with the two most similar maps, and the rest of maps or groups of maps are then aligned to this initial multiple meta-alignment following a guide tree.

③ We have investigated the structure and the shape of the resulting meta-alignments. We have incorporated some modifications in the basic algorithm in order to detect non-collinear configurations in the alignments without additional computational cost.

④ We have successfully applied the meta-alignment algorithms on the biological problem of eukaryotical promoter characterization. First, we have manually curated a collection of orthologous transcription factor binding sites from the literature, that are experimentally verified in human, mouse, rat or chicken. Next, we have trained the meta-alignment program on a subset of well characterized human-mouse promoters, extracted from this collection. Then, we have shown the TF-map alignments are more accurate than conventional sequence alignment to distinguish pairwise gene co-expression in a large collection of microarray results.

⑤ We have also used the meta-alignment approach to distinguish promoters from other gene regions in a set of well characterized human-rodent gene pairs and their corresponding orthologs in chicken and zebrafish. In this particular problem, the multiple meta-alignment identified correctly most orthologous

promoter regions, even when comparing to protein coding regions that presented a stronger sequence conservation.

⑥ We have comprehensively reviewed the topic of sequence alignment, specially focusing on the pioneering algorithms that have mostly contributed to the field. In addition, we have also contributed to extend our expertise in the areas of computational gene finding and promoter characterization, within the field of bioinformatics.

PART IV

Appendices

Curriculum Vitae

PERSONAL DATA

Name:	Enrique Blanco García
Birthplace and birthdate:	Barcelona, January 12th. 1976
Working Address:	Centre de Regulació Genòmica Passeig de la Barceloneta 37-49 Barcelona
Telephone number:	+34 93 224 08 91
E-mail:	eblanco@imim.es
Web page:	http://genome.imim.es/~eblanco

ACADEMIC CURRICULUM

- ENGINEER IN COMPUTER SCIENCE (*Ingeniero superior en Informática*). Facultat d`informàtica de Barcelona. Universitat Politècnica de Catalunya, Spain (June 2000). [Mark: 7.40/10, PFC: MH]
- DEA IN ALGORITHMICS (*Diploma de Estudios Avanzados, Research Sufficiency*). Departament de Llenguatges i Sistemes Informatics. Facultat d`informàtica de Barcelona. Universitat Politècnica de Catalunya , Spain (June 2002).
- AQU CERTIFICATE: Professorat Col.laborador (teaching staff), 25 November 2005.

Language Skills

- English : ADVANCED LEVEL (CERTIFICAT D' APTITUD) (LEVEL C), Official School of Languages, Barcelona (EOIBD), Spain.
- Italian : ELEMENTARY LEVEL (CERTIFICAT ELEMENTAL) (LEVEL B), Official School of Languages, Barcelona (EOIBD), Spain.

- Catalan and Spanish : mother tongues.

RESEARCH CURRICULUM

- 2001 - 2006. PhD student (Software program, Universitat Politècnica de Catalunya) at Genome Informatics Research Lab, IMIM, Barcelona.
  PhD supervisors:
  - Dr. Xavier Messeguer - peypoch@lsi.upc.edu
    (Facultat d`informàtica de Barcelona. Universitat Politècnica de Catalunya)
  - Dr. Roderic Guigó - rguigo@imim.es
    (Genome Informatics Research Lab, Research Group of Medical Informatics. IMIM-UPF-CRG).
- 1999 - 2000. Programmer in Genome Informatics Research Lab, Research Group of Medical Informatics, at IMIM, Barcelona.

Research areas

1. Bioinformatics (algorithmics)
   - Sequence analysis
   - Sequence and map alignments
   - Multiple alignments
   - Representation of biological signals
2. Bioinformatics (computational biology)
   - Characterization of gene regulatory regions
   - Gene expression
   - Comparative genomics
   - Microarray analysis
   - Computational gene prediction
3. Computer Science
   - Algorithmics
   - Artificial intelligence
   - Parallelism and supercomputation
   - Internet aplications

Computer Skills

- Programming languages: Perl, C, C++, Java, LISP, Pascal, Modula, Ada, PVM, Prolog, GAWK
- Document edition: LaTeX, `pdflatex`
- Web design: XML, HTML, JavaScript, CGI-scripts (web servers), Macromedia Flash, CSSs
- Operating systems: Linux, MAC OS X, Irix, Solaris, Windows 95/98/00/XP
- Office: Word, PowerPoint, Excel, Access

Publications

- **E. Blanco**, X. Messeguer, T.F. Smith and R. Guigó. Transcription Factor Map Alignment of Promoter Regions. *PLOS Computational Biology*, 2(5):e49(2006).
- **E. Blanco**, D. Farre, M. Albà, X. Messeguer, and R. Guigó. ABS: a database of Annotated regulatory Binding Sites from orthologous promoters. *Nucleic Acids Research*, 34:D63-D67 (2006).
- **E. Blanco** and R. Guigó. Predictive Methods Using DNA Sequences. In A. D. Baxevanis and B. F. Francis Ouellette, chief editors: *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Third Edition*. John Wiley & Sons Inc., New York (2005). ISBN: 0-471-47878-4.
- S. Castellano, S.V. Novoselov, G.V. Kryukov, A. Lescure, **E. Blanco**, A. Krol. V.N. Gladyshev and R. Guigó. Reconsidering the evolution of eukaryotic selenoproteins: a novel non-mammalian family with scattered phylogenetic distribution. *EMBO reports*, 5(1):71-77 (2004).
- S. Beltran, **E. Blanco**, F. Serras, B. Perez-Villamil, R. Guigó, S. Artavanis-Tsakonas and M. Corominas. Microarray analysis of the transcriptional network controlled by the trithorax group gene ash2 in Drosophila melanogaster, *PNAS*, 100: 3293-3298, (2003).
- **E. Blanco**, G. Parra and R. Guigó. Using geneid to Identify Genes. In A. Baxevanis and D.B. Davidson, chief editors: *Current Protocols in Bioinformatics*. Volume 1, Unit 4.3 (1-26). John Wiley & Sons Inc., New York, (2002). ISBN: 0-471-25093-7.
- G. Parra, **E. Blanco**, and R. Guigó. geneid in Drosophila. *Genome Research*, 10: 511-515, (2000).

Posters

- **E. Blanco**, M. Pignatelli, X. Messeguer and R. Guigó. "Deconstructing the position weight matrices to detect regulatory elements. Systems Biology meeting: global regulation of gene expression". *Cold Spring Harbor: global regulation of gene expression*. (March 2005, New York, USA).
- **E. Blanco**, X. Messeguer and R. Guigó. "Novel computational methods to chracterize regulatory regions. Systems Biology meeting: genomic approaches to transcriptional regulation". *Cold Spring Harbor: genomic approaches to transcriptional regulation*. (March 2004, New York, USA).
- **E. Blanco**, X. Messeguer and R. Guigó. "Alignment of Promoter Regions by Mapping Nucleotide Sequences into Arrays of Transcription Factor Binding Motifs". *Seventh annual internation conference on computational biology-RECOMB*. (April 2003, Berlin, Germany).

- **E. Blanco**, G. Parra, S. Castellano, J.F. Abril, M. Burset, X. Fustero, X. Messeguer and R. Guigó. "Gene prediction in the post-genomic era". *9-th international conference on Intelligent Systems in Molecular Biology*. (July 2001, Copenhaguen, Denmark).
- J.F. Abril, **E. Blanco**, M. Burset, S. Castellano, X. Fustero, G. Parra and R. Guigó; "Genome Informatics Research Laboratory: Main Research Topics."*I Jornadas de Bioinformática* (June 2000, Cartagena, Spain).

Grants

- Predoctoral fellowship. Formacion de Personal Investigador (FPI). Ministerio de Educacion y Ciencia (Spain), 2001-2004.
- Predoctoral fellowship. Institut Municipal d'Investigacio Medica (Spain), 2005-2006.

Participation in Research Projects

- Plan Nacional I+D (2003-2006), ref. BIO2003-05073, Ministerio de Ciencia y Tecnologia (Spain). Principal investigator: Dr. R. Guigó i Serra.
- Plan Nacional I+D (2000-2003), ref. BIO2000-1358-C02-02 Ministerio de Ciencia y Tecnologia (Spain). Principal investigator: Dr. R. Guigó i Serra.

TEACHING CURRICULUM

Topics

- Sequence alignment
- Dynamic programming
- Data structures
- Bioinformatics
- Weight matrices
- Likelihood ratios
- Pattern discovery (EM)
- Computational gene prediction
- Promoter characterization
- Genome browsers on internet

- Artificial neural nets
- Markov models
- Hidden Markov models
- The Human Genome Project
- DNA computing
- Introduction to UNIX

Teaching Activities

2006

- Participation in the master *Tecnologie bioinformatiche applicate alla medicina personalizzata* (Genefinding: a primer). Consorzio21/Polaris - parco scientifico e tecnologico della Sardegna. Pula (Italy). [Master, 20h]
- January-March. Participation in the course *Bioinformatica* at Facultat de Ciencies de la Salut i de la Vida. Universitat Pompeu Fabra. Barcelona (Spain). [University degree, 60h]

2005

- Participation in the course *Bioinformatica* at Facultat de Ciencies de la Salut i de la Vida. Universitat Pompeu Fabra. Barcelona (Spain). [University degree, 60h]
- Participation in the Phd course *Eines informatiques per a genetica molecular* (Computational Gene Prediction). PhD program in Genetics. Facultat de Biologia. Universitat de Barcelona. Barcelona (Spain). [PhD program, 5h]
- Participation in the summer course *Bioinformatica per a tothom* (Genome analysis). Universitat d'Estiu de la Universitat Rovira i Virgili. Reus (Spain). [Summer course, 10h]
- Participation in the summer course *Bioinformatica* (Computational Gene Prediction). Universidad Complutense de Madrid. Madrid (Spain). [Summer course, 6h]
- Participation in the master *Bioinformatics for health sciences* (Introduction to the UNIX environment). Universitat Pompeu Fabra. Barcelona (Spain). [Master, 10h]

2004

- Participation in the course *Bioinformatica* at Facultat de Ciencies de la Salut i de la Vida. Universitat Pompeu Fabra. Barcelona (Spain). [University degree, 60h]
- Participation in the Phd course *Eines informatiques per a genetica molecular* (Computational Gene Prediction). PhD program in Genetics. Facultat de Biologia. Universitat de Barcelona. Barcelona (Spain). [PhD program, 5h]

- Participation in the summer course *Bioinformatica* (Computational Gene Prediction). Universidad Complutense de Madrid. Madrid (Spain). [Summer course, 5h]
- Participation in the master *Bioinformatics for health sciences* (Introduction to the UNIX environment). Universitat Pompeu Fabra. Barcelona (Spain). [Master, 10h]
- Participation in the workshop on *Computational genome analysis* at Cosmocaixa, Fundació La Caixa. Barcelona (Spain). [Workshop, 4h]
- Participation in the *Postgraduate programme in Bioinformatics* (Computational Gene Prediction). Universidade de Lisboa / Gulbenkian Institute. Lisbon (Portugal). [Master, 40h]

2003

- Participation in the course *Bioinformatica* at Facultat de Ciencies de la Salut i de la Vida. Universitat Pompeu Fabra. Barcelona (Spain). [University degree, 60h]
- Participation in the Phd course *Eines informatiques per a genetica molecular* (Computational Gene Prediction). PhD program in Genetics. Facultat de Biologia. Universitat de Barcelona. Barcelona (Spain). [PhD program, 5h]
- Participation in the master *Bioinformatica y biologia computacional* (Computational Gene Prediction). Universidad Complutense de Madrid. Madrid (Spain). [Master, 4h]

2002

- Participation in the course *Bioinformatica* at Facultat de Ciencies de la Salut i de la Vida. Universitat Pompeu Fabra. Barcelona (Spain). [University degree, 60h]
- Participation in the course *Bioinformatica* (Genome analysis) at ALMA bioinformatics. Madrid (Spain). [Course, 8h]

2001

- Participation in the EMBL course *Bioinformatics for comparative and functional genomics* (Computational analysis of promoter regions). Universitat Pompeu Fabra. Barcelona (Spain). [Course, 2h]

2000

- Participation in the EMB-net course *Bioinformatics* (Computational gene identification). Gulbenkian Institute. Lisbon (Portugal). [Course, 20h]

Attended conferences

- Cold Spring Harbor Labs: global regulation of gene expression. (March 2005, New York, USA).
- Cold Spring Harbor Labs: genomic approaches to transcriptional regulation. (March 2004, New York, USA).
- IV Jornadas de Bioinformática Españolas (September 2003, A Coruña, Spain).
- Seventh annual internation conference on computational biology-RECOMB. (April 2003, Berlin, Germany).
- Workshop sobre bioinformatica y biologia computacional. Fundacion BBVA. (April 2002, Madrid, Spain).
- 9-th international conference on Intelligent Systems in Molecular Biology. (July 2001, Copenhaguen, Denmark).
- I Jornadas de Bioinformática Españolas (June 2000, Cartagena, Spain).
- Jornada Catalana de Supercomputación. Parque tecnológico de la Universidad de Barcelona (October 1999, Barcelona).
- Segunda jornada científica sobre análisis computacional de biomoléculas. IMIM-UPF (October 1999, Barcelona).

Software

TF-map alignments

- **Programs:** http://genome.imim.es/software/meta/index.html
- **Web server:** http://genome.imim.es/software/meta/meta.html
- **Datasets:** http://genome.imim.es/datasets/meta2005/index.html

Multiple TF-map alignments

- **Programs:** http://genome.imim.es/software/mmeta/index.html
- **Web server:** http://genome.imim.es/software/mmeta/mmeta.html
- **Datasets:** http://genome.imim.es/datasets/mmeta2006/index.html

The ABS database of annotated promoters

- **Data:** http://genome.imim.es/datasets/abs2005/index.html
- **Constructor:** http://genome.imim.es/datasets/abs2005/constructor.html
- **Evaluator:** http://genome.imim.es/datasets/abs2005/evaluator.html

The geneid program

- **Program:** http://genome.imim.es/software/geneid/index.html
- **Web server:** http://genome.imim.es/software/geneid/geneid.html
- **Annotations:** http://genome.imim.es/genepredictions/index.html

Posters

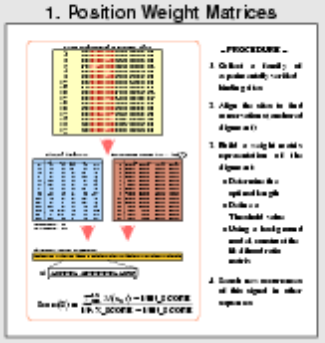
Blanco *et al.*, Cold Spring Harbor, 2005

Deconstructing the Position Weight Matrices to Detect Regulatory Elements

BLANCO, E.[1,2,*] PIGNATELLI, M.[1,3,*] MESSEGUER, X.[2] GUIGÓ, R.[1]

1. Grup d'Anàlisi de la Informació Biomèdica (GRIB) - Universitat Pompeu Fabra - Centre de Regulació Genòmica Barcelona (SPAIN)
2. Grup d'Algorísmia i Genòmica Departament de Llenguatges i Sistemes Informàtics Universitat Politècnica de Catalunya Barcelona (SPAIN)
3. Grup de Genòmica i Biologia del Desenvolupament Departament de Genètica Universitat de Barcelona Barcelona (SPAIN)

1. Position Weight Matrices



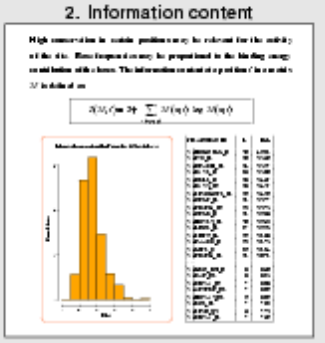
Summary

Position Weight Matrices also called Position-Specific Scoring Matrices have been traditionally used to characterize families of Transcription Factor Binding Sites (TFBSs). New sequences can be analyzed with this model in order to locate putative occurrences of the represented regulatory element. However, the ambiguous nature and the short length of the binding sites usually induce an overwhelming amount of false positive predictions in the search by process.

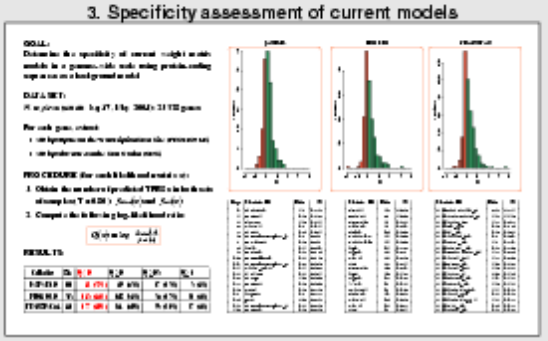
We present a human genome-wide study to assess the specificity of three popular collections of weight matrices -JASPAR, PROMO and TRANSFAC- using coding sequences as a simple background model. Next, we repeat the same evaluation in the mouse genome, to obtain several subsets of very discriminant matrices in both species. Then, we use these smaller collections of matrices to better distinguish two orthologous promoters from other homologous gene components in a set of human and mouse genes.

Finally, we propose an iterative procedure to avoid the potential pitfalls of using the complete collection of weight matrices when a set of co-regulated or homologous gene promoters must be characterized. This simple but effective method detects the statistically over-represented matrices in the input set of sequences that can then be used to annotate the putative regulatory elements.

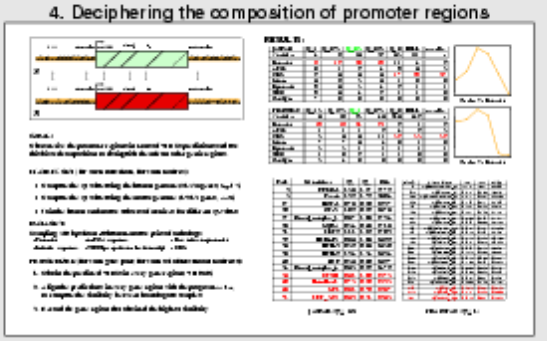
2. Information content



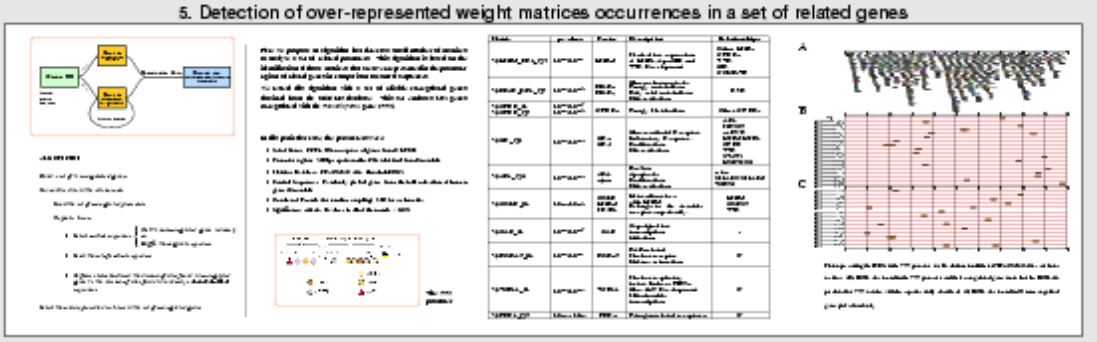
3. Specificity assessment of current models



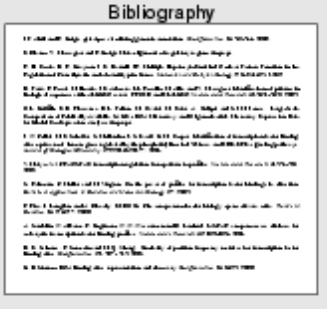
4. Deciphering the composition of promoter regions



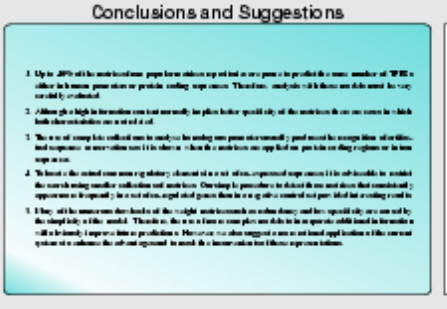
5. Detection of over-represented weight matrices occurrences in a set of related genes



Bibliography

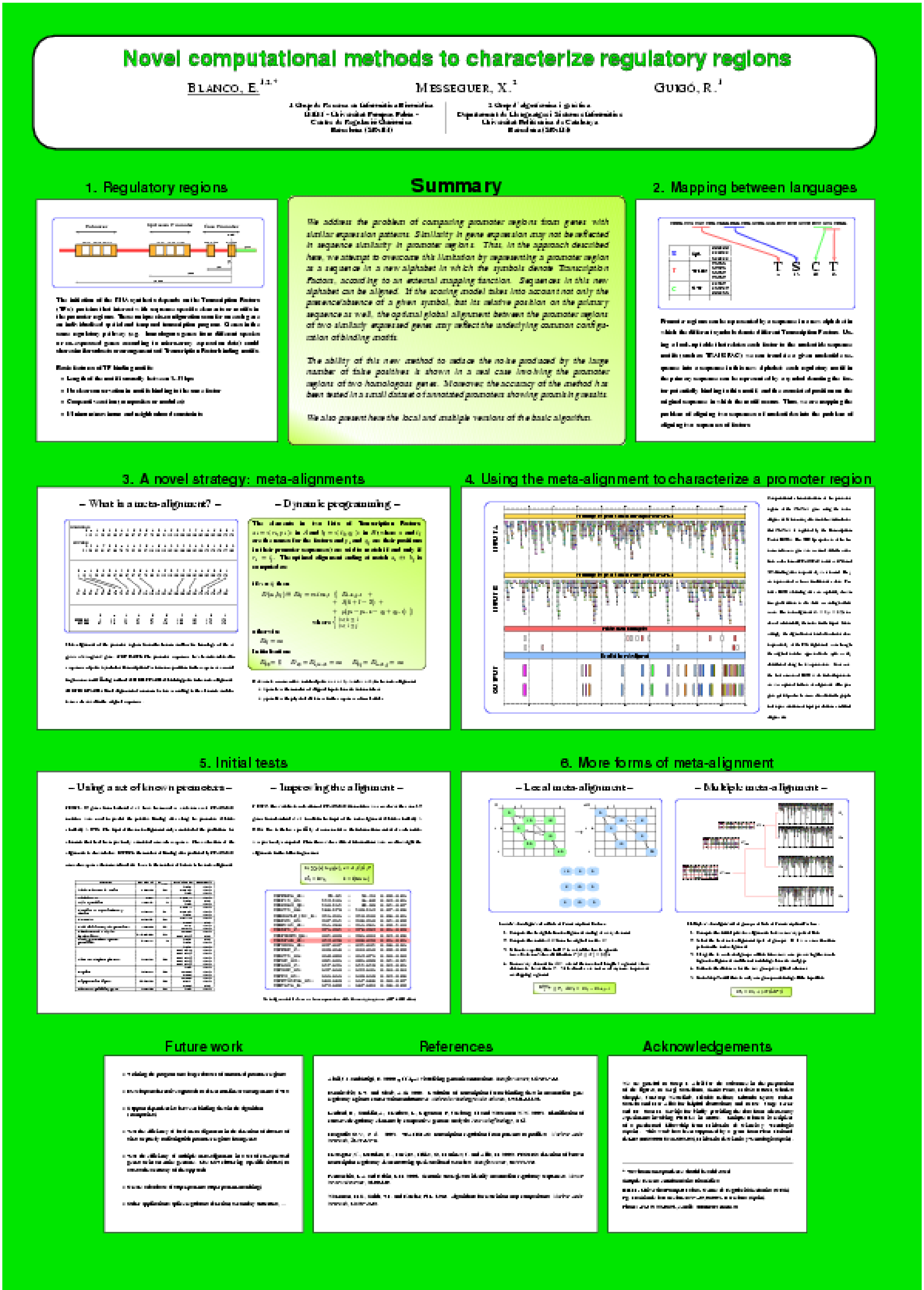


Conclusions and Suggestions



Acknowledgements

Blanco *et al.*, Cold Spring Harbor, 2004


Novel computational methods to characterize regulatory regions
BLANCO, E.
MESSEGUER, X.
GUIGÓ, R.
1. Regulatory regions
Summary
We address the problem of comparing promoter regions from genes with similar expression patterns. Similarity in gene expression may not be reflected in sequence similarity in promoter regions. Thus, in the approach described here, we attempt to overcome this limitation by representing a promoter region as a sequence in a new alphabet in which the symbols denote Transcription Factors, according to an external mapping function. Sequences in this new alphabet can be aligned. If the scoring model takes into account not only the presence/absence of a given symbol, but its relative position on the primary sequence as well, the optimal global alignment between the promoter regions of two similarly expressed genes may reflect the underlying common configuration of binding motifs.
The ability of this new method to reduce the noise produced by the large number of false positives is shown in a real case involving the promoter regions of two homologous genes. Moreover, the accuracy of the method has been tested in a small dataset of annotated promoters showing promising results.
We also present here the local and multiple versions of the basic algorithm.
2. Mapping between languages
3. A novel strategy: meta-alignments
– What is a meta-alignment? –
– Dynamic programming –
4. Using the meta-alignment to characterize a promoter region
5. Initial tests
– Using a set of known promoters –
– Improving the alignment –
6. More forms of meta-alignment
– Local meta-alignment –
– Multiple meta-alignment –
Future work
References
Acknowledgements

Blanco *et al.*, RECOMB, 2003

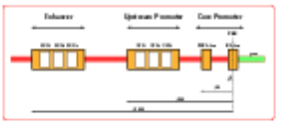
Alignment of Promoter Regions by Mapping Nucleotide Sequences into Arrays of Transcription Factor Binding Motifs

BLANCO, E.[1,2] MESSEGUER, X.[2] GUIGÓ, R.[1]

[1] Grup de Recerca en Informàtica Biomèdica
IMIM - Universitat Pompeu Fabra -
Centre de Regulació Genòmica
Barcelona (Spain)

[2] Grup d'algorísmia i genòmica
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Barcelona (Spain)

1. Promoter regions



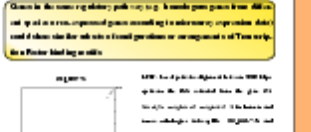
Summary

We address the problem of comparing promoter regions from genes with similar expression patterns (e.g. homologous genes). Similarity in gene expression may not be reflected in sequence similarity in promoter regions which partially explains the limited success of currently available computational methods for promoter prediction.

In the approach described here (based on an existent algorithm to align restriction maps), we attempt to overcome such a limitation representing a promoter region as a sequence in a new alphabet in which the different symbols denote different Transcription Factors. Sequences in this new alphabet can be aligned. If the scoring model takes into account not only the presence/absence of a given symbol, but its relative position on the primary sequence as well, the optimal global alignment between the promoter regions of two similarly expressed genes may reflect the underlying common configuration of binding motifs.

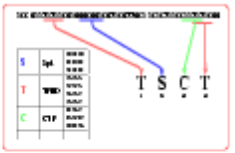
The ability of this new method to reduce the noise produced by the large number of false positives is shown in a real case involving the promoter regions of two homologous genes.

2. Comparative approaches

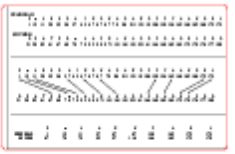


3. A new strategy: meta-alignments

– Mapping of languages –



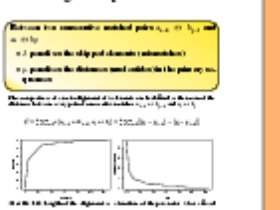
– A simple example –



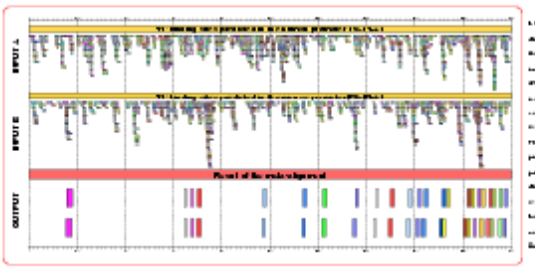
4. Algorithm and parameters

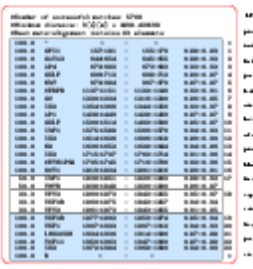
– Dynamic programming –

– Alignment parameters –

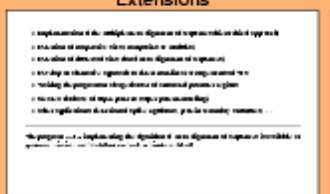


5. Using the meta-alignments to reduce the noise in a real example

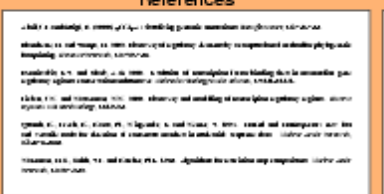




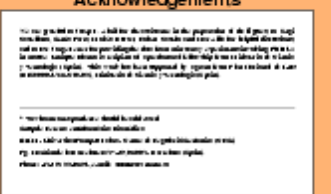
Extensions



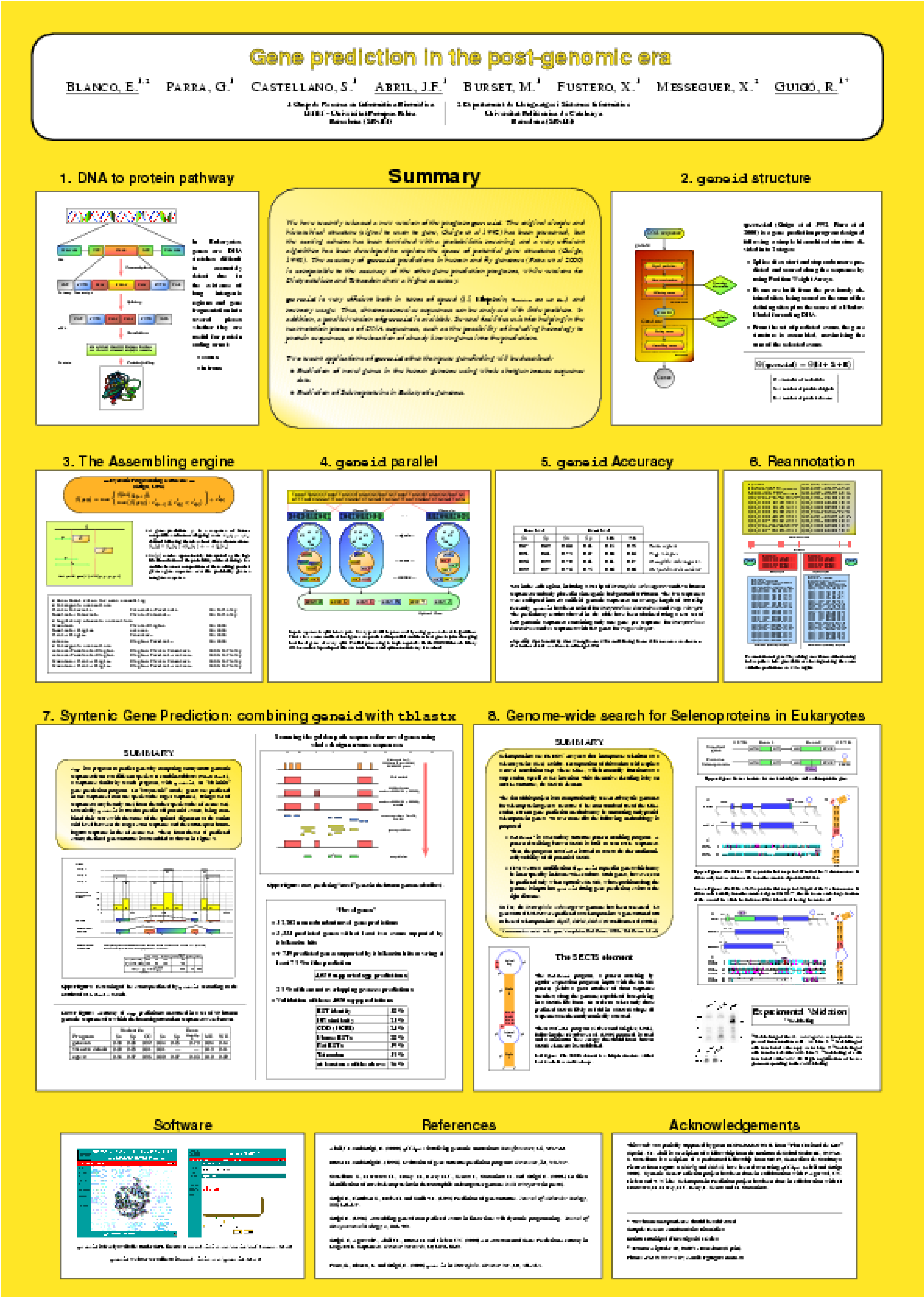
References



Acknowledgements



Blanco *et al.*, ISMB, 2001


Gene prediction in the post-genomic era
BLANCO, E.[1,2] PARRA, G.[1] CASTELLANO, S.[1] ABRIL, J.F.[1] BURSET, M.[1] FUSTERO, X.[1] MESSEGUER, X.[2] GUIGÓ, R.[1]
1. DNA to protein pathway
Summary
2. geneid structure
3. The Assembling engine
4. geneid parallel
5. geneid Accuracy
6. Reannotation
7. Syntenic Gene Prediction: combining geneid with tblastx
8. Genome-wide search for Selenoproteins in Eukaryotes
SUMMARY
The SECIS element
Experimental Validation
Software
References
Acknowledgements

Miscellanea

This thesis layout is largely derived from the LaTeX template created by Robert Castelo in 2002[1]. His templates were extended by Sergi Castellano and Genís Parra for their theses. Josep Francesc Abril substantially improved those files, creating an excellent automatical framework that produces a variety of different formats and layouts. Here, I provide some comments on his version and the modifications I incorporated to, and the source code for download.

Technical comments

This book was typeset with GNU `emacs` 21.3.1 in LaTeX mode and converted to PDF with `pdflatex` 3.14159-1.10b (Web2C 7.4.5). All running on a linux box with Red Hat Fedora Core 2 and kernel 2.6.9-1.6. LaTeX is a document preparation system, powerful, robust and able to achieve professional results (**?**). However, the learning curve may be stiff.

The main document, `thesis.tex`, depends on several LaTeX files—including each chapter, the tables and few POSTSCRIPT figures—, but it also depends on other files—such as style files, hacked LaTeX packages, several bitmaps and the PDF files for the attached papers. Furthermore, `pdflatex` had to be run several times, together with BIBTEX (to produce the bibliography chapter), `makeindex` (to build the index and the web glossary), `thumbpdf` (to generate the main PDF document thumbnails), and few `perl` scripts. A `Makefile` was written to automatize the compilation process of the whole document. In fact, the `Makefile` was extended to produce four versions of the main document. The "*draft*" version does not include figures and the PDF files for the papers, displaying crop marks and boxes around several elements (such as the area reserved for the pictures). The "*proofs*", where everything is included but crop marks and boxes are kept, and different hyperlink types use different colors. The "*pdf*" version is the electronic version in which all the hyperlinks are marked in blue color, crop marks are disabled. Finally, the "*press*" version is very similar to the "*pdf*" one, currently the only difference is that all the hyperlinks are black. The `Makefile` also includes a rule to build the final book "*cover*", which recycles the `abstract.tex` file and takes some customization from the same style file as the main `thesis.tex` file.

The compilation of a complete version of this document takes about 600 seconds—of course, the "*draft*" version takes much less—with an AMD Athlon 64 processor 3200+, with 512 KB of RAM. This is mainly due

[1] R. Castelo, April 2002.
"The Discrete Acyclic Digraph Markov Model in Data Mining"
Faculteit Wiskunde en Informatica, Universiteit Utrecht

to the several steps required to ensure that every reference, index and so on, is in place. The basic build series of commands is the following: an initial `pdflatex`, a BIBTEX run to produce the bibliography, a second run of `pdflatex` to include it, one call to `makeindex` (for the Web Glossary), a third run of `pdflatex` to include the glossary, another call to `makeindex` (to generate the final index) and to `pdflatex`, then `makeindex` and `pdflatex` are run again, an extra run of `pdflatex` is followed by `thumbpdf`, and a final `pdflatex` to obtain the finished document. If any problem was found, like missing references, an extra round of `pdflatex`, BIBTEX and `pdflatex` is performed by the `Makefile`.

Here you can find the version of some of the programs refereed above: BIBTEX version 0.99c (Web2C 7.4.5), `thumbpdf` version 3.2 (2002/05/26), and `makeindex` version 2.14 (2002/10/02).

LATEX Packages

As there are four versions of the document, the `ifthen` package was used to define version specific parameters, as well as to include different files. The package `geometry` facilitates the definition of the page layout. The current document original dimensions for both, the electronic and printed versions, are 170 mm width by 240 mm height. The "*cover*" requires `calc` to calculate automatically the total width for the page layout, which includes the front and the back covers and the spine width. The main document basic font size is the default value for the "`book`" document class, 10 pt.

The `crop` package is usefull to define the trimming marks for the "*draft*" and "*proofs*" versions of this document. It distinguishes between the logical page, the page sizes defined by the user, and the physical page, the page size for the hardcopy. The `layout` package is used in the "*draft*" version to show on the first page the LATEX variable settings controlling the page layout. Another useful package has been `nextpage`, which provides additional "`clear...page`" commands that ensure to get empty even pages at the end of chapters—and of course, to ensure that all chapters begin at odd pages—, even with automatically generated sections like the Bibliography and the Index.

The `babel` package provides a set of options that allow the user to choose the language(s) in which the document will be typeset, for instance language-specific hyphenation patterns. The default language was set to "`english`", while "`catalan`" and "`spanish`" were also loaded for using them for the corresponding translations of the ABSTRACT.

When working with `pdflatex` there are three unvaluable packages: `pdfpages`, which makes it easy to embed external PDF documents, such as the attached publications; `thumbpdf`, it must be included in files for which a user wants to generate thumbnails (which are created by the `thumbpdf` program); and `hyperref`, which extends the functionality of all the LATEX cross-referencing commands to produce `special` commands which a driver can turn into hypertext links. To protect URL characters we must load the `url` package, unless we have already provided `hyperref`. This package has its own version of the `url` macro, enhanced to provide clickable URLs.

To include POSTSCRIPT figures one needs `graphics` and/or `graphicx`. Those packages are modified by `pdflatex` so that they are able to include bitmaps (PNGs, JPEGs, and so on) and PDF files into the document. `color` facilitates the specification of user-defined colors (such as the cover green shades). Figures generated with LATEX can use any of the following packages: `pstricks`, `pstcol`, `multido`.

The bibliography was produced with BIBTEX. The package `natbib` (NATural sciences BIBliography) provides both author-year and numerical citations; it makes possible to define different citation styles. We have set the following options: "`round`", to put citations within parenthesis; "`colon`", to separate multiple citations with colons; "`authoryear`" to show author and year citations (instead of numerical citations);

and the option "sectionbib" to use the package chapterbib. The style "plainnat" was then applied to format the bibliography. The package chapterbib allows to include a bibliography for each chapter. The package minitoc creates a mini table of contents for each chapter as well.

makeidx provides the macros required to make a subject index. To show the capital letter section headings, few variables were redefined on an auxiliary file (header.ist). One glossary was generated for this document: the web references. The package glossary allowed us to customize the format of this section.

We also defined a style file named mythesis.sty. It loads the following font packages: fontenc (with "T1" option), to set extended font encoding (accents and so on); textcomp, to include some extra symbols, such as the Euro symbol for instance; pifont, for SYMBOL and ZAPF DINGBATS fonts; charter, with which roman family is set to BITSTREAM-CHARTER; helvet, with which sans-serif family is set to HELVETICA; euler, with which formulas are set to EULER; and courier, to set typewriter family to COURIER. Other packages that were loaded are: fancyhdr, to produce nice headings; fancyvrb, to extend the verbatim environment; comment, to hide parts of the original LaTeX files; rotating, to rotate boxes of text; and multirow, to get multirow cells within the tabular environment.

Getting the template files

Notes

Titles in the GBL Dissertation Series

2002-01 Moisés Burset.
Estudi computacional de l'especificació dels llocs d'splicing.
[Computational analysis of the splice sites definition.]
Departament de Genètica, Universitat de Barcelona.

2004-01 Sergi Castellano.
Towards the characterization of the eukaryotic selenoproteome: a computational approach.
Departament de Ciències Experimentals i de la Salut, Universitat Pompeu Fabra.

2004-02 Genís Parra.
Computational identification of genes: "ab initio" and comparative approaches.
Departament de Ciències Experimentals i de la Salut, Universitat Pompeu Fabra.

2005-01 Josep F. Abril.
Comparative Analysis of Eukaryotic Gene Sequence Features.
Departament de Ciències Experimentals i de la Salut, Universitat Pompeu Fabra.

2006-01 Enrique Blanco.
.
Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.

Ejemplo de una tesis de Maestría en la Universidad del Atlántico

Ejemplo de Tesis

The sequences are very versatile data structures. In a straightforward manner, a sequence of symbols can store any type of information. Systematic analysis of sequences is a very rich area of algorithmics, with lots of successful applications. The comparison by sequence alignment is a very powerful analysis tool. Dynamic programming is one of the most popular and efficient approaches to align two sequences. However, despite their utility, alignments are not always the best option for characterizing the function of two sequences. Sequences often encode information in different levels of organization (meta-information). In these cases, direct sequence comparison is not able to unveil those higher-order structures that can actually explain the relationship between the sequences.

We have contributed with the work presented here to improve the way in which two sequences can be compared, developing a new family of algorithms that align high level information encoded in biological sequences (meta-alignment). Initially, we have redesigned an existent algorithm, based in dynamic programming, to align two sequences of meta-information, introducing later several improvements for a better performance. Next, we have developed a multiple meta-alignment algorithm, by combining the general algorithm with the progressive schema. In addition, we have studied the properties of the resulting meta-alignments, modifying the algorithm to identify non-collinear or permuted configurations.

Molecular life is a great example of the sequence versatility. Comparative genomics provide the identification of numerous biologically functional elements. The nucleotide sequence of many genes, for example, is relatively well conserved between different species. In contrast, the sequences that regulate the gene expression are shorter and weaker. Thus, the simultaneous activation of a set of genes only can be explained in terms of conservation between configurations of higher-order regulatory elements, that can not be detected at the sequence level. We, therefore, have trained our meta-alignment programs in several datasets of regulatory regions collected from the literature. Then, we have tested the accuracy of our approximation to successfully characterize the promoter regions of human genes and their orthologs in other species.

GBL Dissertation Series

Universitat Politècnica de Catalunya